Effective Decision Support for Semantic Web Service Selection

Dissertation

zur Erlangung des akademischen Grades
Doktor-Ingenieur (Dr.-Ing.)

vorgelegt dem Rat der Fakultät für Mathematik und Informatik
der Friedrich-Schiller-Universität Jena

von Diplom-Informatikerin Friederike Klan
geboren am 26.06.1978 in Jena

Gutachter

1. **Prof. Dr. Birgitta König-Ries**
   Friedrich-Schiller-Universität Jena, D-07743 Jena
2. **Prof. Dr. Sonja Buchegger**
   KTH Royal Institute of Technology Stockholm, SE-100 44 Stockholm
3. **Prof. Dr. York Sure-Vetter**
   Universität Koblenz-Landau, D-56070 Koblenz

Tag der öffentlichen Verteidigung: 29. August 2012

Ehrenwörtliche Erklärung

Hiermit erkläre ich,

- dass mir die Promotionsordnung der Fakultät bekannt ist,
- dass ich die Dissertation selbst angefertigt habe, keine Textabschnitte oder Ergebnisse eines Dritten oder eigenen Prüfungsarbeiten ohne Kennzeichnung übernommen und alle von mir benutzten Hilfsmittel, persönliche Mitteilungen und Quellen in meiner Arbeit angegeben habe,
- dass ich die Hilfe eines Promotionsberaters nicht in Anspruch genommen habe und dass Dritte weder unmittelbar noch mittelbar geldwerte Leistungen von mir für Arbeiten erhalten haben, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen,
- dass ich die Dissertation noch nicht als Prüfungsarbeit für eine staatliche oder andere wissenschaftliche Prüfung eingereicht habe.

Bei der Auswahl und Auswertung des Materials sowie bei der Herstellung des Manuskripts haben mich folgende Personen unterstützt:

- Prof. Dr. Birgitta König-Ries

Ich habe die gleiche, eine in wesentlichen Teilen ähnliche bzw. eine andere Abhandlung bereits bei einer anderen Hochschule als Dissertation eingereicht: Ja / [✓ Nein].

Jena, den 12. Juni 2012

[Friederike Klan]

To Hans and Luise

Acknowledgments

This thesis would have never been initiated nor completed without the encouragement, inspiration and support given by other people to whom I owe a debt of gratitude.

In the first place, this is Prof. Dr. Birgitta König-Ries, who supervised my work. I would like to thank her for believing in me and my abilities, for creating the environment and for providing the financial support that were required to accomplish this work. I would also like to thank her for her patience and for allowing me to approach things in my own way and time frame.

My thanks also go to my colleagues Aygul, Fedor, Mohamed, Torsten and Uli. They have always been a great source of joy and motivation and provided me with advice and help, if needed. I had a great time with all of you!

I'm also thankful to all the people that contributed to the advancement of this thesis. I would like to particularly thank Markus Frömerth for his commitment in improving this work, as well as Maciej, Ivonne, Ulrike and many others for giving critical comments and valuable insights. Special thanks also go to Christian Wozar, who assisted me with the implementation required for the simulative evaluation of part of my work, and to all those who volunteered as a test person for my user studies. I really appreciate their willingness to spent time assessing my work and for their enthusiasm.

My thanks also go to Prof. Dr. Sonja Buchegger and Prof. Dr. York Sure-Vetter, who served as external reviewers for my dissertation. I would like to thank them for their effort and for reviewing this work in time.

More than to everyone else, I'm in dept of gratitude to my parents and all my family, who continuously supported and encouraged me wherever they could and who worked and work hard to make all this possible. In particular, I would like to thank my mum, who wholeheartedly backs up everything I do and who bought me time, where it was missing. I'm also thankful to my sister Ines and to Andreas, who proofread this dissertation very thoroughly and quickly and who improved it by giving valuable feedback. Very special thanks go to my husband Daniel and to my children Josephine and Quentin. Thank you for your support and encouragement, for your honesty and all you give to me. Thank you for being there and making my life brighter.

I dedicate this work to my dad Dr. Hans Lauth and to my grandma Dr. Luise Lauth who I have always admired. The fact, that they cannot see this all happen fills me with deep sadness.

Deutsche Zusammenfassung

In den frühen 1990er Jahren veröffentlichte Tim Berners-Lee seine Vision des Semantischen Webs. Diese beinhaltet, dass im Internet verfügbare Informationen mit einer wohldefinierten und maschinenverständlichen Beschreibung ihrer Bedeutung versehen werden, um auf diese Weise die Kommunikation und Kooperation zwischen Mensch und Computer zu ermöglichen. Die Idee des Semantischen Webs beschränkt dabei nicht nur auf Informationen, sondern lässt sich auch auf über das Internet bereitgestellte Funktionalität in Form von Webdiensten übertragen. Letztere stellen die bekannteste und am weitesten verbreitete Realisierung einer sogenannten Dienstorientierten Architektur (engl. Service Oriented Architecture (SOA)) dar, welche flexible und adaptive verteilte Anwendungen basierend auf lose gekoppelten, gekapselten, mit einer wohl-definierten Schnittstelle versehenen und über das Internet zugänglichen Diensten realisiert. Webdienste, welche mit einer maschinenverständlichen, semantischen Beschreibung der durch sie bereitgestellten Funktionalität versehen sind, werden als Semantische Webdienste (engl. Semantic Web Services (SWSs)) bezeichnet. Basierend auf einer semantischen Beschreibung der Dienstanforderungen eines Dienstsuchenden können geeignete Semantische Webdienste daher automatisch und effektiv gefunden werden, indem die gegebene Dienstanfrage mit den verfügbaren semantischen Dienstangebotsbeschreibungen verglichen wird. Passende Dienste können dann automatisch konfiguriert, kombiniert und schließlich über das Internet genutzt werden.

Während Webdienste als Grundbaustein agiler Geschäftsprozesse in Unternehmen weit verbreitet sind, beschränkten sich die Aktivitäten um Semantische Webdienste lange Zeit ausschließlich auf Forschungsprojekte. In letzter Zeit ist jedoch zu beobachten, dass führende IT-Unternehmen wie Amazon.com oder Google.com ihre in Form von Diensten bereitgestellten Dienstleistungen über proprietäre Web-APIs oder klassische Webdienste öffentlich zugänglich machen. Dies beinhaltet beispielsweise Dienste, die zur Buchung von Flügen, zur Reservierung von Veranstaltungskarten oder zur Routenplanung für eine Reise benötigt werden. Führende Wissenschaftler und Fachleute erwarten, dass sich dieser Trend fortsetzt, sodass die Zahl der öffentlich verfügbaren Webdienste in den nächsten Jahren exponentiell steigen wird. Sie sagen eine Transformation des gegenwärtigen „Internets der Informationen“ hin zu einem „Internet der Dienste“, einer globalen, nutzerzentrierten SOA basierend auf einfach zugreifbaren Webdiensten, voraus. Semantischen Technologien kommt in diesem Zusammenhang eine bedeutende Rolle zu, da sie das automatisierte Auffinden und Kombinieren sowie die automatische Vermittlung zwischen Diensten ermöglichen.

In der vorliegenden Arbeit argumentieren wir, dass das entstehende „Internet der Dienste“ einen neuen Weg eröffnet benötigte Funktionalität in Form von Webdiensten zu finden und zu nutzen. Dieser bietet gegenüber den gegenwärtig vorherrschenden Such- und Nutzungsschemata für Endanwender, d.h. Dienstnutzer, einige entscheidende Vorteile.

- Durch die Verfügbarkeit von semantischen Dienstbeschreibungs- und Dienstvergleichstechniken verspricht Semantic-Webservice-Technologie das Problem der ineffektiven Dienstsuche durch stichwortbasierte Suchmaschinen zu lösen.

- Darüber hinaus stellen semantische Beschreibungen verfügbarer Dienste und deren Eigenschaften die Informationen zur Verfügung, welche benötigt werden, um Dienstnutzern umfassende Entscheidungsunterstützung bei der Auswahl eines geeigneten Dienstes aus der sehr großen Menge potentiell passender und über das Internet verfügbarer Dienste zu bieten.

- Die automatische oder halbautomatische Konfiguration und Kombination von Diensten, sowie deren automatischer Aufruf bei Bedarf ermöglichen eine effizientere, bequemere und damit benutzerfreundlichere Art der Nutzung von Funktionalität über das Internet.

Gegenwärtig stehen diesen Vorteilen jedoch eine Reihe von Unzulänglichkeiten existierender Ansätze aus dem Bereich der Semantischen Webdienste entgegen, welche die Realisierung der vorgestellten Vision bislang verhindern.

- Obwohl Semantic-Webservice-Technologie geeignete Mittel zur semantischen Beschreibung von angebotener und gewünschter Dienstfunktionalität zur Verfügung stellt, erfordert diese, dass Benutzer ihre Dienstanforderungen in einer formalen, Logik-basierten Sprache formulieren, welche für Endanwender nicht geeignet ist.

- Darüber hinaus berücksichtigen bestehende Ansätze aus dem Bereich der Semantischen Webdienste nicht, dass Benutzer zum Zeitpunkt des Aufkommens eines Dienstwunsches typischerweise keine genaue Vorstellungen davon haben, welche Funktionalität von ihnen benötigt wird. Genaue Dienstanforderungen werden vielmehr sukzessive im Zuge des Kennenlernens verfügbarer Dienste und deren Eigenschaften konstruiert. Dies macht Verfahren zur inkrementellen Anforderungserhebung sowie zum Dienstvergleich basierend auf unvollständigen und/oder ungenauen Dienstanforderungen als Teil einer umfassenden Entscheidungsunterstützung zu einer unumgänglichen Voraussetzung für benutzervermittelte Dienstauswahl.

- Da angesichts der Vielzahl der über das Internet verfügbaren Angebote sowie der breit gefächerten Anbietergemeinde Webdienste nicht notwendig von bekannten und vertrauenswürdigen Anbietern offeriert werden, muss davon ausgegangen werden, dass veröffentlichte Dienstangebotsbeschreibungen die von einem Dienst bereitgestellten Leistungen nicht immer korrekt beschreiben. Daher sind Dienstauswahlentscheidungen auf Basis der im Angebot offerierten Diensteigenschaften mit einem gewissen Risiko verknüpft. Um dennoch fundierte Auswahlentscheidungen zu ermöglichen, müssen Dienstnutzer über Art und Umfang dieses Risikos in Kenntnis gesetzt werden.

Ziel der vorliegenden Arbeit ist es die Realisierbarkeit der Vision des „Internets der Dienste“ auf Basis von Semantischen Webdiensten zu demonstrieren, indem ein Ansatz vorgestellt wird, welcher die identifizierten Unzulänglichkeiten bestehender Ansätze, vor allem hinsichtlich ihrer Eignung für den Gebrauch durch Endanwender, behebt. Im Detail, schlagen wir einen **Ansatz zur Auswahl von Semantischen Webdiensten vor, welcher Dienstkonsumenten, d.h. Endanwendern, effektive Entscheidungsunterstützung bei der**

Auswahl von Semantischen Webdiensten bietet. Der zu entwickelnde Ansatz soll Anwender insbesondere in die Lage versetzen sachkundige, ausgewogene und konsistente Dienstauswahlentscheidungen auf effiziente Weise zu treffen. Diese Art der Unterstützung soll trotz der aus ungenauem und unvollständigen Wissen über die tatsächlichen Dienstanforderungen eines Dienstkonsumenten und über die tatsächlichen Leistungen angebotener Dienste resultierenden Unsicherheit über die Eignung einzelner Dienstangebote erbracht werden. Der zu entwickelnde Ansatz hat die durch den Endanwender vermittelte Auswahl von Semantischen Webdiensten zum Ziel. Mögliche Anwendungsfälle finden sich im Bereich des E-Commerce, beschränken sich aber nicht auf diesen. Der Schwerpunkt der Arbeit liegt auf der Nutzbarmachung von Semantic-Webservice-Technologien für den Endanwender. Die Entwicklung effektiver und effizienter Techniken für die Internet-weite Suche von Semantischen Webdiensten ist orthogonal zu dieser Fragestellung und übersteigt den Rahmen der vorliegenden Arbeit.

Einen wesentlichen Beitrag der vorliegenden Arbeit bildet ein von Example-Critiquing-Empfehlungssystemen inspirierter Ansatz zur inkrementellen und interaktiven Erfassung von Dienstanforderungen sowie zur Auswahl von Diensten. Dieser wechselt zwischen Phasen in denen dem Benutzer potentiell geeignete Dienstangebote präsentiert werden und Phasen in denen der Anwender Dienstanforderungen informell, auf Basis der präsentierten Dienstangebote, oder vermittelt durch eine modifizierbare graphische Repräsentation der bereits erfassten Dienstanforderungen spezifizieren kann. Während dieses Prozesses entwickelt der Benutzer schrittweise seine Dienstanforderungen und Präferenzen und trifft schließlich eine Auswahlentscheidung. Als integralen Bestandteil dieses Systems beschreiben wir ein Modell, welches die Dienstanforderungen und Präferenzen eines Dienstkonsumenten sowie die aus fehlendem oder fehlerhaftem Wissen resultierende Unsicherheit über diese genau beschreiben kann. Dieses wird benötigt, um die Benutzer effektiv bei der Durchführung ihrer Aufgaben unterstützen und anleiten zu können. Darüber hinaus erläutern wir ausführlich wie das Anforderungsmodell eines Benutzers basierend auf dessen Interaktionen mit unserem System kontinuierlich aktualisiert wird, um hinzukommendes Wissen über die Dienstanforderungen und Präferenzen des Benutzers genau abzubilden. Wir erklären ferner, wie geeignete Dienstangebote basierend auf dem mit Unsicherheit behafteten Anforderungsmodell eines Dienstkonsumenten ermittelt werden können und wie diese in geeigneter Weise präsentiert werden können. Weiterhin legen wir dar, wie es dem Anwender ermöglicht wird wünschenswerte Diensteigenschaften basierend auf den präsentierten Dienstangeboten anzugeben. Wie wir in unserer Arbeit zeigen werden, kann ein um wenige Aspekte erweitertes, bereits existierendes Vergleichsverfahren für semantische Dienstbeschreibungen verwendet werden, um potentiell geeignete Dienstangebote basierend auf unvollständigem und/oder ungenauem, d.h. unsicherem, Wissen über die Dienstanforderungen eines Nutzers zu ermitteln. Schließlich zeigen wir, wie der Prozess der Anforderungserhebung und der Dienstauswahl gesteuert und fokussiert werden kann, um Modellunsicherheit effektiv zu reduzieren und damit zur Effizienz des Dienstauswahlprozesses beizutragen.

Um Informationen über das tatsächlich zu erwartende Ergebnis der Ausführung eines verfügbaren Webdienstes und damit über das mit dieser verbundenen Risikos bereitstellen

zu können, schlagen wir ein flexibles Feedback-System vor, welches sich die von Dienstnutzern bereitgestellten Rückmeldungen über das Ergebnis einer Dienstausführung zunutze macht. Der Schwerpunkt unserer Arbeit liegt hierbei auf der Frage, wie verfügbares Nutzerfeedback effektiv genutzt werden kann, um das zu erwartende Resultat einer Dienstausführung vorhersagen zu können. Im Detail stellen wir ein Feedback-Modell zur Verfügung, welches das Ergebnis der Ausführung eines Webdienstes hinsichtlich seiner zahlreichen Facetten detailliert beschreiben kann. Dieses wird um ein Verfahren ergänzt, welches Dienstkonsumenten effektiv bei der Bereitstellung dieser Art von Feedback unterstützt und sich dabei der Bereitschaft des Nutzers Informationen zur Verfügung zu stellen anpasst. Wir werden zeigen, dass auf diese Weise geeignete, umfassende und aussagekräftige Rückmeldungen über das Resultat einer Dienstausführung erfasst werden. Das vorgestellte Verfahren berücksichtigt darüber hinaus auch die Anforderungen eines Benutzers hinsichtlich seiner Privatsphäre, in dem es ihm gestattet den Detaillierungsgrad der bereitgestellten und mit anderen Nutzern ausgetauschten Informationen genau zu kontrollieren. Es verwendet eine Verschleierungstechnik, um die Übermittlung aussagekräftiger Feedbackinformationen zu ermöglichen und zugleich so wenig wie möglich persönliche Informationen preiszugeben. Schließlich zeigen wir, dass verfügbares Konsumentenfeedback effektiv genutzt werden kann, um den Grad und die Art des mit der Ausführung eines verfügbaren Dienstes verbundenen Risikos vorherzusagen. Ferner legen wir dar, wie diese Informationen bei der Sortierung der verfügbaren und passenden Dienstangebote geeignet berücksichtigt werden können. Die vorgeschlagene Lösung berücksichtigt, dass Konsumentenfeedback sowohl subjektiv als auch kontextabhängig ist und verwertet Konsumentenfeedback, welches in einem Nutzungskontext erhoben wurde, um Wissen über das Ergebnis einer Dienstausführung in einem anderen Nutzungskontext abzuleiten. Das vorgestellte Verfahren weist den Benutzer auf das mit der Ausführung eines gegebenen Dienstes verbundene Risiko hin, indem auf Basis von Konsumentenfeedback gewonnene Informationen auf effektive, intuitive und personalisierte Weise präsentiert werden. Fragestellungen wie das Erkennen nicht wahrheitsgetreuer Nutzerrückmeldungen und der Schutz vor Feedbackmanipulationen sind ausführlich in anderen Forschungsbereichen, beispielsweise im Gebiet der Vertrauensbildungs- und Reputationssysteme oder im Bereich der Empfehlungssysteme, diskutiert worden und werden in dieser Arbeit nicht berücksichtigt.

Abschließend stellen wir eine Implementierung unseres Ansatzes zur Verfügung, um dessen Realisierbarkeit zu demonstrieren. Im Gegensatz zu vielen anderen Lösungen im Bereich der semantischen Webdienste, haben wir unsere Arbeit ausführlich und gründlich evaluiert und die daraus resultierenden Ergebnisse umfassend dokumentiert. Letztere belegen die Effizienz und die Effektivität unseres Ansatzes. Die Arbeit schließt mit einer Zusammenfassung und der Diskussion möglicher weiterführender Fragestellungen, die sich aus den vorliegenden Forschungsergebnissen ergeben.

Abstract

In the early 1990s, Tim Berners-Lee published his vision of the Semantic Web. By providing information with a well-defined and machine-comprehensible meaning, the Semantic Web was expected to enable communication and cooperation between humans and computers. This idea is not restricted to information, but has been also applied to functionality offered through the Internet as Web Service. The latter is the most prominent implementation of a Service Oriented Architecture (SOA), which has become increasingly popular as a powerful way of designing flexible and adaptive distributed applications based on loosely coupled, standalone services that are accessible through the Internet via a well-defined interface. Web Services provided with a machine-processable, semantic description of their capabilities are called Semantic Web Services. Based on a semantic description of a service requester's service requirements, suitable service offers can be automatically and effectively retrieved by comparing (matching) the given service request with available offer descriptions. Services might be automatically configured and composed and finally invoked over the Internet. While Web Services at the heart of agile business processes are widely used in enterprises, activities around Semantic Web Services have long been restricted to research projects. However recently, we observe a trend of leading IT-companies such as Amazon.com or Google.com to open up their internal functionality encapsulated as business services and make it accessible via proprietary Web-APIs or classic Web Services. This includes services such as required for booking a flight, making a ticket reservation or getting directions for a trip. Leading researchers and practitioners in the field expect an exponential growth of the number of publicly available Web Services and forecast a shift from the current Internet of Information to an Internet of Services, a global, user-centric SOA based on easily accessible Web Services. Semantic technologies are expected to play a central role in the realization of this vision, since they enable automatic discovery, composition and mediation of services.

In this thesis, we argue that the emerging Internet of Services based on Semantic Web Services will offer a new way of retrieving and using functionality over the Internet, which has clear advantages over prevalent retrieval and usage schemes that are employed by end-users, i.e. service consumers.

- Through the use of semantic service description and matchmaking techniques, Semantic Web Service technology promises to solve the problem of ineffective Web Service retrieval that existing keyword-based search engines currently have.
- Moreover, semantic descriptions of available services and their characteristics will furnish the information that are required to provide sophisticated support for selecting an appropriate service among the huge number of potentially fitting services offered over the Internet.
- Automatic or semi-automatic configuration, composition and on-demand invocation of services will enable the efficient, convenient and thus more user-friendly use of functionality offered over the Internet.

However, though existing approaches to (Semantic) Web Service retrieval offer many advantages, they are not well-suited to this new usage scenario. This currently hampers the realization of the outlined vision.

- Though Semantic Web Service technology provides adequate means to semantically describe service capabilities and in particular service needs, it requires users to formulate their service requirements at a formal, logic-based level that is not appropriate for end-users.
- Moreover, existing approaches to Semantic Web Service selection do not account for the fact, that, at the time a service selection problem arises, users typically do not have a complete picture of the service functionality they desire. Service requirements are rather constructed over time as the user becomes familiar with available service alternatives and their properties. Hence, support for incremental requirements elicitation and matchmaking based on incomplete and/or inaccurate service requirements, as part of sophisticated decision support are necessary prerequisites for user-mediated service selection.
- Since today services are no longer offered by known and trusted providers, service offer descriptions describe a service's capabilities not necessarily correctly. Thus, service selection decisions on the basis of the properties promised in offer descriptions are associated with a certain risk. To still allow for well-founded selection decisions, service consumers need to be informed about this risk.

The objective of this thesis is to demonstrate the feasibility of the vision of the Internet of Services based on Semantic Web Services by suggesting an approach that overcomes the identified shortcomings of existing approaches, particularly those that hamper their application by end-users. More specifically, we propose an approach to Semantic Web Service selection that **effectively supports service consumers, i.e. end-users, in efficiently making well-informed, balanced and consistent service selection decisions in the presence of uncertainty arising from inaccurate and incomplete knowledge about both, a consumer's service requirements and the capabilities of offered services.** The system targets to enable end-user mediated Semantic Web Service selection as required in, but not restricted to an E-Commerce setting. The focus of our work is on making Semantic Web Service technology usable for end-users. Developing effective and efficient techniques for Web-scale Semantic Web Services retrieval is orthogonal to this issue and out of the scope this thesis.

Our main contribution is an incremental and interactive approach to requirements elicitation and service selection that is inspired by example critiquing recommender systems. It alternates phases of intermediate service recommendation and phases of informal requirements specification based on the presented service alternatives or by using a modifiable graphical requirements representation. During that process, the user incrementally develops his service requirements and preferences and finally makes a selection decision. As part of this system, we describe a model of the consumer's service requirements and preferences that is maintained to effectively support and guide the user in his tasks. We explain how uncertainty about the service consumer's true requirements and preferences, that is caused by missing and/or inaccurate knowledge, is explicitly represented within this model and how the model is continuously updated based on the user's interactions to accurately reflect the systems's growing knowledge about the user's service requirements and preferences. We also explain how suitable service offers can be retrieved based on the requirements model and how they can be appropriately displayed. Moreover, we discuss how users can be enabled to indicate desirable service characteristics based on the presented alternatives. As we demonstrate in our work, standard matchmaking with a minor extension can be applied to retrieve potentially matching service offers based on incomplete and/or inaccurate, i.e. uncertain, service requirements. Finally, we demonstrate how the requirements elicitation and service selection process can be directed and focused to effectively reduce model uncertainty and thus to contribute to the efficiency of the service selection process.

To acquire information about the actual performance of available services and thus about the risk that is associated with their execution, we propose a flexible feedback system, that leverages reported consumer experiences made in past service interactions. In this context, the focus of our work is on how to effectively utilize feedback information to evaluate a service's expected performance. In particular, we provide means that allow to detailedly describe a service's performance with respect to its multiple facets. This is supplemented by a user-adaptive method that effectively assists service consumers in providing such feedback. As we will demonstrate, this method supports the user in supplying appropriate, comprehensive and meaningful feedback and thereby adjusts to his willingness to provide feedback. The mechanism also accounts for the privacy requirements of the user by allowing him to customize the detailedness of the feedback information that are elicited and shared with others. It makes use of an obfuscation technique to enable the propagation of meaningful feedback while at the same time revealing as little personal information as possible. Finally, we demonstrate that available consumer feedback can be effectively exploited to assess the degree and kind of risk that is associated with the execution of an offered service and show how these information can be appropriately considered during the process of service ranking. The suggested solution accounts for the subjective and context-dependent nature of consumer feedback and exploits consumer experiences made in one context to infer knowledge about a service's behavior in another context. It makes the user aware of the risk that is associated with the execution of a service by presenting feedback-derived information in an effective, intuitive and personalized manner. Aspects such as detecting dishonest feedback and counteracting feedback manipulation have been extensively discussed in other research areas, such as trust and reputation systems and recommender systems research and are out of the scope of our work.

Finally, we provide a proof of concept implementation of our solution to demonstrate its feasibility. In contrast to many other approaches related to Semantic Web Service technology, we performed an extensive and thorough evaluation of our contribution and documented its results. These show the effectiveness and efficiency of our approach. The thesis concludes with a summary and a discussion of future research directions related to the considered area.

Contents

List of Tables

List of Figures

List of Symbols

$w_{f,a}(r,s)$	attribute- and feedback-item-specific relevance weight of feedback item f for the performance prediction of service s with respect to the request r and attribute a
$w_f(r,s)$	feedback-item-specific relevance weight of feedback item f for the performance prediction of service s with respect to the request r
$w_{fb}(a)$	feedback weight of attribute a
ac	attribute condition
$FS_r(a)$	set of all valid feedback structures that can be derived from the request (model) subtree rooted at attribute a of request (model) r
$path_r(a)$	path of attribute a in the request (model) r
R_I	range of the instance set I
V_a^{spec}	the set of values that have been considered in the direct condition specified for the request (model) attribute a
a	request (model) attribute
A	set of request (model) attributes
$attrSim_{provider}(f,r,a)$	attribute-specific feedback provider similarity of feedback item f and service request r with respect to the request (model) attribute a
$attrSim_{service}(f,s,a)$	attribute-specific service (offer) similarity of feedback item f and service s with respect to the request (model) attribute a
$attrSim(f,r,s,a)$	attribute-specific similarity of feedback item f with respect to service request r, service s and request (model) attribute a
v	an attribute value
V_a	the set of values that the request (model) attribute a might take
conf(R)	confidence of an association rule R
c	service consumer
conv(R)	conviction of an association rule R
$Pref_I^{cond}(i)$	random variable, indicating the user's preference for the service instance i w.r.t. the condition *cond* specified for the set I
$pref_I^{cond}(i)$	the user's preference for the service instance i w.r.t. the condition *cond* specified for the set I
C	set of attribute-specific and user-specified critiques
cs	connecting strategy
dc	direct condition
E	set of judgment experiences

f	feedback item
FM_F	feedback matrix over the feedback items in F
fs	feedback structure
F	set of feedback items
A_{fs}	set of attributes considered in the feedback structure fs
A_{I_a}	set of attributes defined for the target set I_a of request (model) attribute a
T_a	target type of attribute a
fs_{judged}	judged feedback structure
fs_{rec}	recommended feedback structure
$mv_a(r, o)$	matching value for the request (model) r and the offer o with respect to the attribute a
$MV_I(r, i)$	expected matching value for the request (model) r and the instance i with respect to the instance set I
$name(I)$	name of the attribute condition whose target set is I
in	in-condition
i	service instance
I	set of service instances
In_i^I	random variable for the instance i of instance set I, indicating whether i is acceptable or not
o_s	offer of service s
$type(I)$	type of the instance set I
$\overrightarrow{MR}_a(r, \vec{O})$	vector of matching results for the request (model) r and the offers in the vector of offers $\vec{O}$ with respect to the attribute a
A_{fs}^{judged}	set of attributes that has to be judged according to the feedback structure fs
$\overline{rat}_a$	mean rating provided for the request (model) attribute a
$mv(r, o)$	(overall) matching value for the request (model) r and the offer o
$notIn$	not-in-condition
$NotIn_i^I$	random variable for the instance i of instance set I, indicating whether i is not acceptable or not
O	set of service offers
$ovSim_{provider}(f, r)$	overall feedback provider similarity of feedback item f with respect to request r
$ovSim_{service}(f, r, s)$	overall service similarity of feedback item f with respect to request r and service s

$\mathit{ovSim}(f, r, s)$	overall similarity of feedback item f with respect to request r and service s determined based on the attribute-specific similarity values of f
$P_a^{acc}(v)$	probability of the attribute value v being acceptable with respect to the direct condition(s) that have been specified for the request (model) attribute a
$p_a^{acc}(v)$	probability distribution indicating for each attribute value v its probability of being acceptable with respect to the direct condition(s) that have been specified for the request (model) attribute a
$p_{attr}(r,\mathit{fs})$	probability distribution indicating for each feedback structure fs derived from r the likelihood of the user being willing to judge the kinds of service aspects as required by it
$p_{\mathit{In}_i^I}$	probability distribution over possible values of In_i^I
$p_{judges}(r,\mathit{fs};\alpha)$	parameterized probability distribution indicating for each feedback structure fs derived from r the likelihood of being judged
p_{Max_I}	probability distribution over possible values of Max_I
p_{Min_I}	probability distribution over possible values of Min_I
$p_{\mathit{NotIn}_i^I}$	probability distribution over possible values of NotIn_i^I
$p_{num}(r,\mathit{fs})$	probability distribution indicating for each feedback structure fs derived from r the likelihood of the user being willing to judge as many service aspects as is required by it
$\mathit{pref}_I^{cond}(i;\mathit{par}_1,\ldots,\mathit{par}_n)$	preference function over instances i of instance set I depending on the parameters $\mathit{par}_1,\ldots,\mathit{par}_n$ w.r.t. the condition *cond*
$\widetilde{\mathit{rat}}_a(r, s)$	predicted rating for request (model) attribute a with respect to the service s and the request (model) r
p	service provider
$p_{W_{ac}}$	probability distribution over possible weights of an attribute condition *ac*
range	range condition
$\mathit{rat}_a(f)$	rating provided for request (model) attribute a by feedback item f
r	request (model)
r_a	request (model) subtree rooted at the attribute a in r
Max_I	maximum of the range for instance set I (random variable)
max_I	maximum of the range for instance set I
Min_I	minimum of the range for instance set I (random variable)
min_I	minimum of the range for instance set I
$\mathit{root}(r)$	root attribute of the request (model) r
s	service

$\mathit{sim}_{attr}(\mathit{fs},\mathit{fs}')$.	similarity of two feedback structures *fs* and fs' with respect to the kinds of service aspects they require to be judged
$\mathit{sim}_{dc}(r_a, r'_{a'})$.	similarity of the given request model subtrees' root attributes a and a' w.r.t. the uncertain direct conditions that have been specified on their target sets (direct condition similarity)
$\mathit{sim}_{num}(\mathit{fs},\mathit{fs}')$	similarity of the two feedback structures *fs* and fs' with respect to the number of service aspects they require to be judged
$\mathit{sim}_{provider}(f, r)$	feedback provider similarity of feedback item f and service request r
$\mathit{sim}_{req}(r, r')$ ...	semantic similarity of two requests (request models) r and r'
$\mathit{sim}_{service}(f, s)$	service (offer) similarity of feedback item f and service s
$\mathit{sim}_{sub}(r_a, r'_{a'})$	similarity of the given request model subtrees' root attributes a and a' w.r.t. their child trees (subtree similarity)
$\mathit{sim}_{type}(r_a, r'_{a'})$	similarity of the given request model subtrees' root nodes a and a' with respect to their ontological type (type similarity)
supp(X)	support of an itemset or association rule X
tc	type condition
I_a	target set of the request (model) attribute a
$w^*_{r,a}$	total weight of request (model) attribute a in the request (model) r
U($\mathit{Pref}^*_I(i)$) ...	uncertainty about the uncertain preference value $\mathit{Pref}^*_I(i)$
$W_{r,ac}$	uncertain weight of an attribute condition *ac* in the request (model) r
$w_{r,a}$	weight of request (model) attribute a in the request (model) r

Part I.

Introduction

1 Motivation & Overview

In this chapter, we will motivate our work and demonstrate its relevance by providing a possible usage scenario (Section 1.2). Based on this scenario, we will derive the overall goal of the thesis (Section 1.3) and identify a set of operational objectives, whose achievement will lead to the fulfillment of this goal (Section1.4). We will also provide the reader with necessary background information related the topic at hand (Chapter 2) and finally will give an outline of the thesis (Section 1.5).

1.1 Motivation

Since its early beginnings in the 1990s, the World Wide Web has become an integral part of our everyday life. Today, it serves not just as a global platform for social interactions, but is also increasingly used as a medium for commercial activities such as purchasing goods or using paid services. According to Nielsen, one of the world's leading companies for marketing research, the proportion of Internet users that had ever made an online purchase tremendously increased from 10% in 2005 to 84% in 2010, which amounts to a number of 875 million people [Com08, Com10]. Though the focus is still on purchasing products such as books or clothing, now services become more and more attractive to online consumers. In a survey conducted in 2010, 32% of the 27,000 respondents indicated that they intent to book a flight online, 26% said that they will make a hotel reservation. This is an increase of 7% and 9% respectively, compared to 2009 [Com10]. People were also not afraid of making costly purchases such as buying electronic equipment, computer hardware or cars or booking a trip [Com10].

This increasing popularity of online shopping can be mainly explained by two facts. Firstly, shopping anywhere and at any time with products delivered directly to the customer is convenient and time-saving. Secondly, compared to brick and mortar stores, the World Wide Web offers easy, fast and cheap access to more detailed information about a wider range of products and services, information that are required to research and compare available offers and that would have had otherwise to be assembled laboriously from a number of physical stores or would not have been available at all [Cha05]. The latter particularly refers to consumer recommendations and reviews, which are considered to be highly important by the majority of the customers, especially when it comes to purchase costly products or services such as cars, consumer electronics and airline tickets or those that affect personal welfare and health such as medication, cosmetics and insurances [Com10].

These facts are in contrast to the finding, that just a small fraction of the customers makes use of these information to identify the good or service that is most suitable to them. 37% of the Internet users employ search engines, 19% use shopping comparison websites and 32% rely on personal recommendations [Com08]. However, the vast majority, 60% of the users [Com08], does not research and compare available offers and simply buys from the site they regularly use [Com08]. The reason for this is that available information are scattered across the Web and that their amount is huge. Search engines leave the task of collecting, weighting and structuring these information completely to the user and thus make research related to a certain purchase decision mentally demanding and time-consuming. In contrast to that, shopping comparison sites such as Shopping.com or Kelkoo.com search multiple online stores at a time and aggregate information related to a desired product or service. Since they are explicitly dedicated to product and service search, they can leverage structured data feeds from the retailers or focused crawlers and thus can provide more relevant information than general search engines. However, the provided information are typically restricted to the product or service price and to product or vendor ratings and reviews. Basic decision support in terms of browsing facilities is offered, but in general limited to brands, shop names and simple product categories. Finally, the effectiveness of both, search engines and shopping comparison engines, as a research assistant is restricted, since they solely offer keyword-based search and thus have to rely on information extraction techniques to acquire relevant information, which are imprecise per se. This is particularly true for search queries referring to transactional needs, i.e. Web-mediated activities, such as using a service [Bro02, HCG01, TT08].

Hence, compared to traditional shopping, buying from home might be convenient, but due to the lack of sophisticated assistance for product and in particular service research is still tedious and time-consuming. Moreover, it requires the customer to employ a two-step shopping approach comprising of a preceding research phase and the actual purchase [DZN$^+$06]. For the latter, the user has to explicitly navigate to the website of the product or service provider, has to manually input required payment and shipping information and finally has to commit the purchasing transaction. This is particularly annoying, if the user's shopping need, requires the invocation of a number of related services that are spread over several websites, such as buying a train ticket and booking a hotel for a single trip. These findings do not just apply to commercial activities, but also to other Web-mediated activities such as downloading a file or getting directions to a location.

Beside the growing interest in E-Commerce from both, the consumer and the business side, we observe another trend in the World Wide Web. In the early 1990s, Tim Berners-Lee published his vision of the Semantic Web as ”an extension of the current one, in which information is given well-defined [and machine-comprehensible] meaning, better enabling computers and people to work in cooperation” [BLHL01]. Since those days, sophisticated semantic technologies have been developed and implemented that bring us closer to the realization of this vision [SLH06]. However, the idea is not restricted to information, but has been also applied to functionality offered through the Internet as Web Service. The latter is the most prominent implementation of a Service Oriented Architecture (SOA), which has become increasingly popular as a powerful way of designing flexible and adaptive distributed applications based on loosely coupled, standalone services that are accessible through the Internet via a well-defined interface. Web Services provided with a machine-processable, semantic description of their capabilities are called Semantic Web Services [Yu07, CS06, FLP$^+$07]. Based on the semantic description of a service requester's service requirements, suitable service offers can be automatically and effectively retrieved by comparing (matching) the given service request with available offer descriptions. Services might be automatically configured and composed and finally invoked over the Internet.

While Service Oriented Architectures as a means to integrate heterogeneous business applications into agile business processes based on loosely coupled Web Services are widely used in enterprises, activities around Semantic Web Services have long been restricted to research projects. Hence, though Semantic Web Services are an active area of research and are on the focus of numerous EU funded research projects [1], up to now virtually no real-world applications that use this technology are available [BF08]. However recently, we observe a trend of leading IT-companies such as Amazon.com or Google.com to open up their internal functionality encapsulated as business services and make it accessible via proprietary Web-APIs or classic Web Services. This includes services such as required for booking a flight, making a ticket reservation or getting directions for a trip. According to the Web Service Search Engine Seekda[2], there are about 28,000 Web Services publicly available on the Internet today. Leading researchers and practitioners in the field expect this number to exponentially grow as other companies will follow the current trend and provide access to their internal business services [FFST11]. They forecast a shift from the current Internet of Information to an Internet of Services [SJ07, BDB05], a global, user-centric SOA based on easily accessible Web Services. Semantic technologies are expected to play a central role in the realization of this vision, since they enable automatic discovery, composition and mediation of services [FFST11]. Hence, to date, a number of research projects such as the EU-funded projects SOA4All[3] and INSEMTIVES[4] have been launched to provide the technology that is required to create, retrieve and use Semantic Web Services at a large scale.

In this thesis, we argue that the emerging Internet of Services based on Semantic Web Services will offer a new way of retrieving and using functionality over the Internet, which has clear advantages over prevalent retrieval and usage schemes that are employed by end-users, i.e. service consumers. In particular, Semantic Web Service technology promises to

[1]For related European research projects of the Sixth and Seventh Research Framework Program refer to http://cordis.europa.eu/fp7/ict/ssai/fp6_en.html and http://cordis.europa.eu/fp7/ict/ssai/projects_en.html, respectively.

[2]http://webservices.seekda.com/

[3]http://www.soa4all.eu

[4]http://www.insemtives.eu

solve the problem of ineffective Web Service retrieval that existing keyword-based search engines currently have. Moreover, semantic descriptions of available services and their characteristics will furnish the information that are required to provide sophisticated support for selecting an appropriate service among the huge number of potentially fitting services offered over the Internet. Automatic or semi-automatic configuration, composition and on-demand invocation of services will enable the efficient, convenient and thus more user-friendly use of functionality offered over the Internet.

However, though existing approaches to (Semantic) Web Service retrieval offer many advantages, they are not well-suited to this new usage scenario. This currently hampers the realization of the outlined vision. Since the emphasis of previous Semantic Web Service research was on Web Services as basic building blocks of enterprise integration architectures, existing approaches in this area mostly target to support application developers and experienced business processes designers. Hence, though Semantic Web Service technology provides adequate means to semantically describe service capabilities and in particular service needs, it requires users to formulate their service requirements at a formal, logic-based level that is not appropriate for end-users, i.e. the service consumer in an E-Commerce setting. Basic tools that support this task exist, but mainly address Web Service developers. Virtually, no end-user support for specifying service requirements is available. To complicate matters, when intending to purchase online, service consumers are typically unfamiliar with the often complex and interdependent characteristics of available products and services and thus have no clear-cut requirements. Moreover, they usually have complex preferences, that are not known in advance, but are rather constructed while learning about the product or service space [JWP93]. Current Semantic Web Service solutions do not account for these facts. They implicitly assume that semantic descriptions of a consumer's service requirements are readily available and do not consider incremental and interactive requirements elicitation as well as sophisticated decision support as necessary prerequisites for user-mediated service selection in a Web-scale. The latter is particularly important, since researching the often huge space of products and services that are offered over the Internet is mentally demanding and time-consuming, but worthwhile, especially when it comes to costly purchases. In addition, existing approaches presume that request descriptions perfectly describe a consumer's actual requirements. However, as already argued, this is not the case. In fact, the system's knowledge about a consumer's requirements is incomplete and often inaccurate and has to be incrementally extended as additional requirements emerge. As a consequence, there is uncertainty about a user's true requirements. This fact should be considered when matching and ranking available service alternatives to allow for adequate service selection decisions.

Another restraint to the vision of Semantic Web Service enabled E-Commerce applications is that traditional approaches to semantic matchmaking evaluate the suitability of available service offers exclusively by comparing the published offer descriptions with a given request description [TT08]. Hence, they implicitly assume that offer descriptions describe a service's capabilities correctly. However, since in a today's E-Commerce setting services are not necessarily offered by known and trusted providers, this assumption is no longer valid. In addition, service descriptions are necessarily incomplete and incorrect to some degree [GMP06] and thus may not accurately describe a service's capabilities. To date, this problem is even amplified, since the creation of complex offer descriptions is left to the application developer without much assistance. As a consequence, service selection decisions on the basis of the properties promised in offer descriptions is associated with uncertainty about the actual outcome of a service invocation. Existing approaches to service selection typically either ignore this fact or exclusively consider Quality of Service aspects. However, to allow for well-founded selection decisions, service consumers need and, as already argued, demand to be informed about the risk that is associated with a certain selection decision. Thereby, risk assessment should not be restricted to non-functional aspects such as response time, but should rather refer to functional aspects of the delivered service, which are far more relevant to the consumer in an E-Commerce context [LMRD09].

However, the issues raised above are not entirely new and have been partly addressed in other research areas. For example in case-based reasoning and recommender systems research [Smy07], solutions for incremental requirements elicitation already exist. However, those approaches lack the powerful knowledge representation and matchmaking capabilities provided by Semantic Web Service technology. Moreover, they do not sufficiently address the issue of uncertainty about consumer requirements. Evaluating the trustworthiness of service providers is also an active research topic mainly discussed in the context of trust and reputation systems research [JIB07]. However, existing solutions typically make a number of simplifying assumptions about services and the behavior of their providers and thus are not directly applicable to a Semantic Web Service based E-Commerce scenario, where we have complex and richly described services. We think that Semantic Web Service research can benefit from this work and in return will generate helpful insights and input to those areas.

In this thesis, we will therefore suggest a framework that utilizes this knowledge from related research areas to address the deficiencies of traditional Semantic Web Service frameworks as identified above. The envisioned framework targets to enable end-user mediated Semantic Web Service selection as required in, but not restricted to an E-Commerce setting. In particular, the proposed solution will account for the constructive nature of consumer requirements and will consider the uncertainty arising from incomplete and inaccurate knowledge about both, consumer requirements and service capabilities. The envisioned system, is intended to effectively support consumers in efficiently making well-informed, balanced and consistent service selection decisions in the presence of uncertainty.

We believe that Semantic Web Service research will benefit from such an approach in several ways. In particular, it addresses the following major research challenges identified by leading researchers and practitioners in the area [5]:

[5]We mainly considered research priorities for the 6th [eu206] and 7th [nes09] European Research Framework Program as highlighted by the European Technology Platform on Software Architectures and Services Infrastructures, NESSI, as well as the findings of a comprehensive Delphi study on the potential of Semantic Web Services as a basis for integration architectures conducted with leading practitioners and researchers in 2007 [BF08].

- Semantic Web Service frameworks should provide ”intelligent” end-user support [eu206], ”so that these can easily be utilized by human users with only little IT literacy” [nes07]. This includes effective and intelligent tool-, retrieval- and service management support [Agr06, AMLM07, KMSF09, nes07, TT08].
- Service-based systems should provide mechanisms to ”achieve end to end security and trust” [nes09] and ensure Quality of Service [BF08, eu206, nes09].
- The Semantic Web Service community requires ”pilot applications focusing on every-day needs of consumers, citizens, industry etc., which can demonstrate the benefits of using semantics” [AMLM07, BF08].

1.2 Usage Scenario

Using a typical real world usage scenario from the E-Commerce domain, we will demonstrate the relevance of Web-scale Semantic Web Service selection and emphasize the need for effective decision support to enable this. In particular, we will illustrate the shortcomings of using online services as it is performed and supported today, especially with respect to the quality of decision making. We will discuss how Semantic Web Service technology and the emerging Internet of Services can be leveraged to overcome these problems and will identify what is currently missing, but required to actually implement this. This will motivate the overall goal of the thesis, which is presented in the subsequent section.

Scenario *Max, a PhD student at the university in Jena got a paper accepted at a conference in Lyon, France. He is starting to make arrangements for travel and accommodation and researches the Web for available offers. In a first attempt, he enters the search terms "book", "trip", "Jena" and "Lyon" at Google.de. The top result that is returned by Google.de is a video interview in the Look Book series of the New York Magazine with Jenna Lyons, the creative director of the apparel retailer J.Crew. Since all other links provided at the first result page refer to Jenna Lyons as well, Max decides to try another combination of keywords and inputs "book flight Lyon". As a result, Google.de returns a list of links referring to flight offerings from several airlines. Max selects the first one, which takes him to the website of British Midland International. As Max soon finds out, the airline offers flights to Lyon, but not from Germany.*

Slightly irritated, he is calling his colleague, who recommends the travel website Expedia.de to make his arrangements and to compare available offers. After that, Max points his browser to Expedia.de. To save money, he looks for combined offers of flight and hotel. After having completed the query form, he receives a list of 129 *flight and hotel packages. While indecisively scrolling through the list, it comes to his mind that it would be good to have a hotel close to the conference location. Since Expedia.de does not provide means to search for hotels nearby a certain location, Max decides to use Google Maps*[6] *to search for*

[6]http://maps.google.de

appropriate hotel offers. He enters the conference location extracted from the conference website and looks for hotels nearby. As a result, he is provided with a map displaying a number of 10 *close-by hotels. Max decides in favor of a hotel that is fairly close to the conference location and fits to his budget. Since Google Maps does not offer a review or rating for the selected hotel, Max copies the hotel name and enters it at Expedia.de. As he finds out, Expedia.de users rated the hotel with* 3.8 *out of* 5 *stars. Curious about the reasons for the downgrade, he checks the detailed ratings, which are available for the aspects "cleanliness of the rooms", "service and staff", "room comfort" and "hotel condition". As it turns out the staff is not particularly obliging, but Max does not care about that. However, due to bad experiences in previous conference trips, he would like to know whether the breakfast that is offered is ok. Since a detailed rating for that aspect is not available, he reads through the* 14 *available textual reviews to find the desired information. Finally, Max books the selected hotel. To take advantage of the discount for bundles of hotel and flight provided by Expedia.de, he takes the flight that is offered with the selected hotel.*

Two month later, Max starts on his journey one day before the conference. As he arrives in Lyon, he notices that the hotel does not offer W-LAN access to his guests, which is inconvenient, since he can neither check emails from the hotel nor talk to his girlfriend via voice over IP. When planning his route to the conference location for the next day, Max realizes, that the closest metro station is 15 *walking minutes away, which adds up to* 30 *minutes to get to the conference every day. On the next day, as if it all were not annoying enough, he meets a conference participant, who stays at a hotel that is further away from the conference location than that of Max, but is cheaper and next to the metro, which takes him conveniently to the conference venue within* 20 *minutes. Finally, Max gets to know that a colleague of him traveled to the conference by train, which was* 50 *€ cheaper, took all in all about the same time and was more convenient.*

In order to evaluate the quality of the decision making process particularized in the given scenario, i.e. the process of choosing services to make arrangements for travel and accommodation for a conference trip, and in order to evaluate the quality of the decision support that was provided, we first need to define the abstract term "decision making quality" more precisely and operationally, i.e. in terms of verifiable characteristics. For that purpose, we adopt the following definition by Payne et al. [PBS99]:

> "[Effective decision making] is based on thorough processing of information (reason and reflection) that is transparent and in proportion to the importance of the question at hand [...]. Such processing should include consideration of a range of alternative courses of action, consideration of the full range of objectives to be fulfilled, thorough consideration of the information most critical to the individual, the making of tradeoffs, and careful review of responses to detect inconsistencies."

The definition identifies three types of information that have to be taken into account in order to make a good decision: **information about relevant decision alternatives**, i.e. the service alternatives, **information about the decision maker's objectives**, i.e. his service requirements, and **knowledge about how to resolve conflicting requirements**. Those information have to be acquired, thoroughly processed and evaluated to make a service selection decision that is optimal or close to optimal with respect to the service consumer's service requirements and the available service alternatives.

Definition 1.1. (**Decision making effectiveness**) *The process of service selection is effective, if the resulting decision is*

- *well-informed, i.e. taken in consideration of relevant service alternatives and their properties,*
- *balanced, i.e. taken after deliberate resolution of conflicting service requirements, and*
- *consistent, i.e. made in consciousness of the user's service needs and optimal or close to optimal with respect to these needs and the available service alternatives.*

Evaluating the introduced scenario with respect to these standards, leads to the conclusion that the described decision making process is not effective. This is due to the following facts:

- The selection decision was not made in consideration of an appropriate range of available service alternatives, since Max was not aware of the fact, that he could have traveled by train much cheaper and more conveniently.
- It was also not the result of a careful resolution of conflicting requirements, since although Max's hotel selection was a tradeoff between the distance to the conference location and price, it did not trade off between price and commuting time.
- Finally, the decision did not account for all of Max's requirements, since he selected a hotel that did not offer W-LAN access, which was important to Max.

Hence, the resulting selection decisions that were made, i.e. booking of a certain hotel and a flight, are not (close to) optimal with respect to Max's service requirements and with respect to the available service alternatives.

The decision making process was also not efficient, since it was tedious and overly time-consuming, particularly in light of the decision quality.

Definition 1.2. (**Decision making efficiency**) *The process of service selection is efficient, if a selection is made within an appropriate period of time and with adequate mental effort. Appropriateness thereby refers to the (subjective) importance of the service selection task at hand.*

There are two major reasons for this:

- **Ineffective and inefficient retrieval:** Required knowledge about relevant services and their properties was scattered over different websites and could not be effectively and efficiently acquired using available keyword-based retrieval techniques. This made the process of knowledge acquisition difficult and time-consuming and hampered the comparison of different decision opportunities.
- **Lack of sophisticated decision making support:** Max was not appropriately supported in making the decision that was complex, both, in terms of the number of available decision alternatives and in terms of the number of interrelated requirements that had to be considered. Assistance was neither available for effectively and efficiently processing those information, nor for evaluating them with acceptable mental effort.

As already argued, we believe that the emerging Internet of Services based on Semantic Web Services, that is propagated by leading researchers, can be leveraged to overcome these shortcomings and will pave the way for a new generation of E-Commerce based on Semantic Web Services. This is for the following reasons:

- Semantic Web Service technology can solve the problem of ineffective and inefficient retrieval. This is due to the fact, that semantic service description languages allow for the precise and machine-comprehensible specification of service requirements. Those semantic requirements descriptions can then be leveraged to effectively and automatically retrieve, configure, compose and invoke on-demand appropriate functionality offered through the Internet as standalone Web Services.
- Semantic Web Service solutions can provide sophisticated decision support for service selection, since they benefit from the availability of semantic service descriptions, that provide rich and machine-comprehensible information about available service alternatives and their characteristics.

While the issue of effective and efficient Semantic Web Service retrieval in a Web-scale is addressed by current research, the provision of sophisticated decision support for Semantic Web Service selection has been neglected so far.

1.3 Thesis Objective

The previous section motivates the overall objective of this thesis, which is the creation of a system, that provides effective end-user support for making both, effective and efficient, service selection decisions. Such a system shall be applicable, but is not restricted to, service selection decisions, as they arise in an E-Commerce setting.

As already argued, consumer requirements are constructive by nature and information resources on the Web are not necessarily trustworthy. Hence, the envisioned system has to account for the fact that, both, knowledge about the consumer's service requirements as well as knowledge about the capabilities of available service offers might be inaccurate and incomplete, i.e. is uncertain.

The overall objective of this thesis can be formulated as follows:

Objective 1. (**Thesis objective**) *Develop a system that* effectively *supports service consumers in* efficiently *making* well-informed, balanced *and* consistent *service selection decisions in the presence of uncertainty arising from inaccurate and incomplete knowledge about a consumer's service requirements and the capabilities of available services.*

Appropriate targets of such a system are decision problems that are

- complex, in terms of the number of available service alternatives and/or in terms of the number of interrelated service requirements that have to be considered, and
- important in terms of resulting in some kind of (not necessarily monetary) loss for the decision maker, if not solved properly.

Otherwise, providing sophisticated decision support is not required and/or not appropriate.

1.4 Solution Design

In the previous sections, we motivated and formulated the thesis objective of providing effective decision support for service selection and compiled a list of criteria that allow for the verification of its fulfillment. However, we did not provide any specific actions to take in order to achieve this goal. In this section, we will derive those actions in terms of operational objectives, indicating how to implement the envisioned system. An in-depth analysis of service selection as a class of decision problems will deliver those objectives.

Howard [How88], who coined the term decision analysis for that process, suggested a 3-step iterative analysis cycle to perform this task (Figure 1.1). The first step, termed *Problem formulation*, comprises the process of formally modeling the considered decision problem at hand. According to Howard, this includes the specification of three main components: the decision alternatives, the preferences of the decision-maker and the information that is relevant for for making the decision (Figure 1.2). The latter refers to "any models, relationships or probability assignments that may be important in characterizing the connection between decisions and outcomes". Based on the resulting model, which is referred to as decision basis, the optimal decision alternative is determined (*Evaluation phase*). In a last step, called *Appraisal*, it is assessed whether the acquired decision is convincing or not. In the latter case, a refinement or reformulation of the decision problem and thus another pass of the analysis cycle, is required. The appraisal step often involves a sensitivity analysis, which investigates the robustness of the optimal decision against slight changes of the

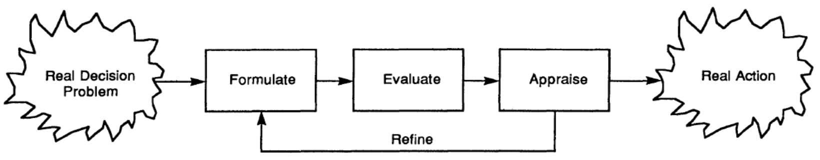


Figure 1.1.: Howard's decision analysis process (taken from [How88])

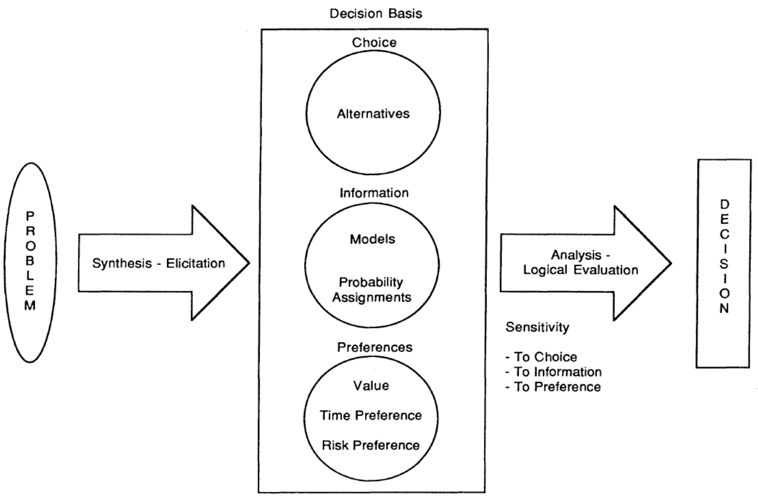


Figure 1.2.: Problem formulation (taken from [How88])

problem model. The cycle stops if the problem model is "requisite", i.e. "its form and content are sufficient to solve the problem". In particular, this means that "everything required to solve the problem is represented in the model or can be simulated by it" [Phi84].

In the remainder of this section, we analyze service selection as a class of decision problems and derive essential components of a system that assists consumers in making such decisions. Adopting the presented 3-step analysis procedure, we start by defining and formulating service selection as a decision problem to identify and characterize the type of knowledge that is required to solve it (Section 1.4.1). After that, we determine the components that are necessary to evaluate this knowledge in order to identify promising decision alternatives and to enable effective and efficient decision making based on it (Section 1.4.2). Finally, required means to evaluate the robustness of a decision in terms of the reliability and completeness of the information it is based on as well as means to determine relevant, but missing knowledge, to contribute to the quality of a decision are derived (Section 1.4.3).

1.4.1. Problem Formulation

We provide a very general definition of service selection.

Definition 1.3. (**Service selection**) *Service selection is the problem of choosing a service out of a set of alternative services that best fits to a consumer's potentially conflicting service requirements.*

This definition implies that service selection is in fact a decision problem. However, it should be noted, that while aiming at providing decision making support for service selection, we actually do not have to deal with a single decision problem, but with a class of decision problems. Thus, when modeling service selection as a decision problem, we have to provide generic models that can be instantiated to characterize a particular decision situation. More specifically, we have to model decision alternatives, i.e. services and their capabilities, and the user's objectives, i.e. his service requirements including his preferences. The latter will allow us to compare and rank service alternatives according to their suitability.

Modeling service requirements A model of the consumer's requirements should cover a specification of the service functionality that is required by the consumer as well as a model of his preferences regarding the properties of alternative service offers. However, in general we cannot assume to have an accurate and complete model of those requirements readily available. In fact, at the time a service selection problem arises, users typically do not have a complete picture of the functionality they desire and an even vaguer idea of their preferences. Those requirements are rather constructed over time as the user becomes familiar with available service alternatives and their properties [JWP93]. This fact should affect both, the way knowledge about service requirements is modeled and elicited. In particular, knowledge about the user's service requirements cannot be merely elicited from the user, it rather has to be interactively acquired and incrementally refined as it is constructed by the user. Since service requirements and preferences are constructed when being faced with available service alternatives, the two processes of requirements elicitation and service selection cannot be separated, they rather have to be interwoven into a process of incremental requirements elicitation and service selection that alternates phases of intermediate service recommendation based on partially known requirements and requirements refinement based on the presented service alternatives. The fact, that the system's knowledge about the consumer's requirements is incomplete and potentially inaccurate, i.e. uncertain, should be considered when making service recommendations and when providing personalized assistance. Hence, model uncertainty has to be explicitly represented within the requirements model.

Modeling service capabilities In order to assess the suitability of available service alternatives with respect to the consumer's service requirements, a rich model of the functionality provided by those services is required. Those models are typically published by the service providers in terms of service offer descriptions. However as already argued (Section 1.1), we cannot be sure about the validity of those information. Hence, service selection decisions based on the properties promised in published offer descriptions are associated with uncertainty about the actual outcome of a service invocation. Uncertainty per se is not bad. However, if the outcome of a service interaction is worse than promised, service execution is also associated with some kind of loss and thus with a certain risk [Hub07]. Awareness and knowledge about this risk is essential for making well-founded service selection decisions. To acquire this knowledge, information about the actual outcome of available service alternatives are required. However, in general, a single consumer has experiences with just a few service providers at his disposal and thus lacks sufficient knowledge. As a solution to this problem, collaborative feedback mechanisms have been proposed in the literature (see [JIB07] for an overview). This type of approaches has been successfully and widely used [JIB07, Del02] and allows to predict a service's future performance. It thus enables risk assessment. In order to assess the risk that is associated with the execution of a service, we have to devise such a feedback mechanism. This includes the design of a model, that is capable of describing feedback referring to multi-faceted service interactions, and the development of an effective mechanism for eliciting this feedback.

1.4.2. Problem Evaluation

In order to support the user in identifying service offers, that can fulfill his requirements, there is a need for a mechanism that is capable of matching the requirements encoded in the requirements model to the capabilities offered by available services as described in their capabilities models, i.e. the service offer descriptions, and finally rank matching offers according to their suitability. This requires both, requirement and capability models to be constructed in a way that allows for such a comparison and ranking in an effective and efficient way. As discussed earlier, this functionality is already provided by existing semantic matchmakers. However, since the knowledge about a consumer's service requirements is potentially inaccurate and incomplete, existing matchmaking functionality has to be appropriately extended to be capable of retrieving matching service offers based on uncertain requirements. In order to support the user in making well-informed, balanced and consistent service selection decisions, we have to provide means to educate him about the characteristics of available services and have to encourage him to explicitly resolve conflicting requirements. We already argued, that due to the fact that service offer descriptions are potentially inaccurate and incomplete, the outcome of a service interaction might not be as promised in the offer and thus might result in some kind of loss, e.g. in case a service provides "less than expected" by the consumer. Hence, to enable well-informed decisions, the user should be made aware of the risk that is associated with the execution of a certain service. This requires the development of both, a procedure that effectively leverages available consumer feedback to predict a service's future performance and thus the risk that is associated with its execution, and means to effectively communicate this knowledge to the user. Thereby, the latter should account for different risk attitudes.

1.4.3. Appraisal

The aim of the formulation phase is the construction of a requisite model of the decision problem. Such a model should have an appropriate form and should consider all aspects that are relevant for the decision problem at hand. Sensitivity analysis is the instrument that helps to decide whether an existing model is requisite or not. In terms of our problem, this means that in order to make robust decisions, we have to ensure: 1) that requirement models and service offer descriptions take an appropriate form, 2) that they are based on a domain model that is provided in an appropriate format and allows to describe the service aspects that are relevant for service selection decisions in a certain domain and 3) that they are complete, i.e. include all service aspects that are relevant in the context of a certain service selection decision. While the fulfillment of the requirements 1) and 2) can be verified at development time, this cannot be done for the third one. In fact, for service offer descriptions, the fulfillment of requirement 3) cannot be guaranteed at all, since their creation is in the responsibility of the service provider. The completeness and accuracy of the consumer's requirements model is essential for making well-informed and consistent service selection decisions and has to be ensured by both, the system and the user at runtime, when a specific requirements model is available. To enable this, we have to provide means to identify knowledge gaps related to the maintained requirements model and means to to acquire required knowledge. This will effectively reduce model uncertainty and thus contributes to the efficiency of the decision making process. However, effectively closing knowledge gaps requires the service consumer to be able to communicate his service requirements to the system. Empowering end-users to do so, requires the provision of means to specify service requirements in an intuitive and informal way.

1.4.4. Operational Objectives

The summarize the above discussion, we can state, that the thesis objective of effectively supporting service consumers in efficiently making well-informed, balanced and consistent service selection decisions (Objective 1), can be achieved by accomplishing the two subsequent objectives.

Objective 2. (**Modeling, elicitation and usage of consumer feedback**) *Devise a feedback mechanism that is both, effective in terms of acquiring knowledge about the risk that is associated with the execution of a service and effective in terms of its ability to support service selection in the presence of this risk. This includes*

- *the design of a model, that is capable of describing feedback referring to multi-faceted service interactions,*
- *the development of an effective mechanism for eliciting this feedback,*
- *the provision of a procedure that effectively leverages available consumer feedback to predict a service's future performance and thus the risk that is associated with its execution, and*
- *a means to effectively communicate this knowledge to the user and thereby accounting for different risk attitudes.*

Objective 3. (**Modeling and elicitation of consumer requirements for service selection**) *Develop an incremental and interactive method for requirements elicitation and service selection that effectively supports service consumers in making well-informed, balanced and consistent service selection decisions and enables them to do this efficiently. This includes*

- *the design of a requirements model, that is capable of describing a user's service requirements and preferences and explicitly represents model uncertainty,*
- *the development of a mechanism that is capable of retrieving matching service offers based on uncertain requirements encoded in the requirements model,*
- *the provision of means to educate the user about the characteristics of available services and means to encourage him to explicitly resolve conflicting requirements,*
- *the provision of means to effectively reduce uncertainty about the consumer's service requirements, and*
- *means to specify service requirements in an intuitive and informal way.*

This thesis will complement research results related to the following issues, which themselves are not considered by this work:

- service mediation
- service composition
- effective Semantic Web Service retrieval at a large scale

- detecting and counteracting dishonest consumer feedback
- efficient feedback propagation and retrieval

1.5 Thesis Outline

To provide the reader with a quick overview, the basic structure of the thesis is depicted and briefly explained below.

I. Introduction
1. Motivation & Overview
2. Background

II. Decision Support for Semantic Web Service Selection
3. Overview
4. Underlying Service Description Language
5. Modeling and Elicitation of Consumer Requirements for Service Selection
6. Modeling, Elicitation and Usage of Consumer Feedback

III. Implementation and Evaluation
7. System Implementation
8. Evaluation of the Requirements Elicitation and Service Selection Mechanism
9. Evaluation of the Feedback Mechanism

IV. Final Considerations
10. Summary and Conclusions
11. Future Work

Appendix
A. Ontology for the Computer Items Domain
B. Questionnaire for the Evaluation of the Requirements Elicitation and Service Selection Mechanism
C. Questionnaire for the Evaluation of the Judgment Recommendation Mechanism

Subsequent to this section, the introductory part of the thesis (Part I) will be completed by providing background information related to the thesis topic (Chapter 2). This involves a brief introduction to Service Oriented Architectures and Web Services as well as an introduction to human decision making and decision support systems. The parts II and III make up the main part of the thesis and each are basically structured according to the two main objectives of this thesis, i.e. requirements elicitation and service selection as well as elicitation and usage of consumer feedback, as identified in Section 1.3. They provide a detailed discussion of our approach to support consumers in making service selection decisions (Chapters 3-6), its implementation (Chapter 7) and evaluation (Chapters 8 and 9). This also includes an analysis of research efforts related to each of the considered topics. Finally, Part IV summarizes our contribution and highlights future research directions. The Appendix contains supplementary information on selected aspects of the thesis.

2 Background

This thesis aims at the provision of decision support for service selection (cf. Section 1.3). We will therefore introduce Service Oriented Architectures as a flexible way of organizing and using distributed functionality (Section 2.1). We will detail on services as the basic building block of Service Oriented Architectures and explain how they are typically used. We will discuss the challenges that arise from the change in scale and target audience of Service Oriented Architectures induced by their Web-scale use and emphasize the role of semantics in addressing those challenges. We will provide background information on human decision making and its shortcomings, particularly when facing complex decision problems as they arise when being required to make service selection decisions at a Web-scale (Section 2.2). Finally, we will introduce Decision Support Systems and discuss their role in mitigating these shortcomings.

2.1 Service Oriented Architectures and Semantic Web Services

Service Oriented Architecture refers to a particular way of

> *"organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations" [OAS06]).*

More specifically, in a Service Oriented Architecture distributed capabilities are organized as network-accessible, standalone *services*, that provide an encapsulated and loosely coupled piece of functionality, i.e. that are self-contained and make no or little assumptions on other system components. Those services are offered by *service providers* and can be used by *service consumers*, who require a certain functionality (cf. Figure 2.1). Thereby, service capabilities can range from simple currency converters to complex business services that can be e.g. used to make a hotel reservation. In addition to the service itself, the service provider publishes a description of the service's capabilities in a *service registry*, which

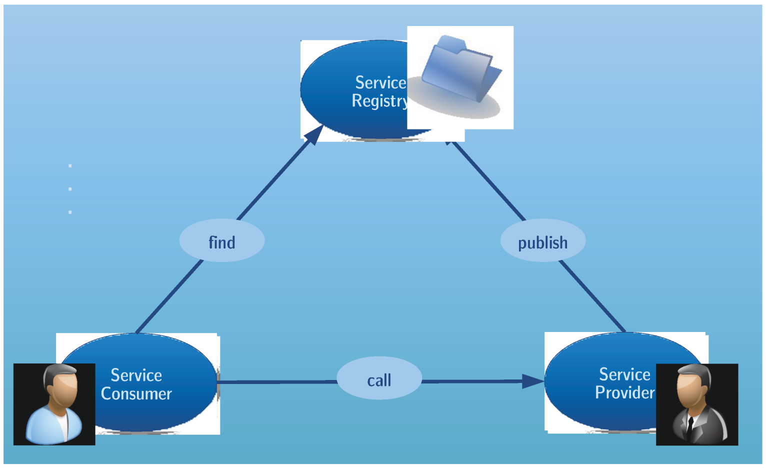


Figure 2.1.: Service Oriented Architecture- basic structure

comprises information that is required to decide whether the given service is suitable for a service consumer and information that are required for the actual interaction with the service. This includes a well-defined, machine-comprehensible and platform-independent description of the service interface and the communication protocol it adheres to as well as details about the location of the service and about how to access the service at a technical level. This might be supplemented by a description of the provided service effect and the preconditions for its usage. Based on these published *service offer descriptions*, potential service consumers can *discover*, i.e. locate, offered services and can decide whether and/or to what degree a certain service provides a desired functionality. The latter is referred to as *service matchmaking*. A matching service can then be *selected*, *bound* and *invoked* over the Internet. Thereby, the communication between and with services is based on open protocols and platform-independent data formats and is well-defined via its interface description.

Organizing distributed functionality in the described way promises to offer composability, reusability, interoperability, seamless integration and easier maintainability of IT systems. Service Orientation is thus an established principle for flexible, quick and low-cost application development in enterprises and for the integration of existing enterprise applications, particularly legacy systems [Res10]. Service Oriented Architectures can be implemented by means of Web Services. Though other realizations such as RESTful[1] Web Services became popular in the context of the Web 2.0, "classical" Web Services based on the stack of Web Service technologies[2] such as the SOAP protocol specification [3] for XML-based data exchange between and with services, WSDL [4] as the language to describe service

[1]REpresentational State Transfer, [Fie00]
[2]http://www.w3.org/TR/ws-arch/
[3]http://www.w3.org/TR/soap
[4]Web Service Description Language, http://www.w3.org/TR/wsdl

interfaces and UDDI [5] as XML-based service registry are commonly used in enterprises [Sho08, Res10].

The advantageous properties of Service Oriented Architectures predestinates Service Orientation as the underlying design principle for offering and using functionality over a huge and heterogeneous network such as the Internet. In fact, we recently observe an increasing number of services being published on the Internet as more and more companies start to offer their internal business functionality as Web Services. Experts expect this trend to continue and envision the emergence of an Internet of Services [SJ07, BDB05], a Web-scale, user-centric SOA based on easily accessible Web Services.

> *"The Internet of Services is a vision of the Internet of the Future where everything that is needed to use software applications is available as a service on the Internet, such as the software itself, the tools to develop the software, the platform (servers, storage and communication) to run the software. [...] Anybody who wants to develop applications can use the resources in the Internet of Services to develop them."*[6]

Compared to existing SOAs, the Internet of Services imposes challenges with respect to both, scale and target audience. In particular, we face a huge number of service providers that are no longer known and trusted and that offer services that are even more heterogeneous than in an intra- or inter-enterprise setting. Moreover, while up to now Web Services have been typically retrieved and used by system developers to create end-user applications, now end-users shall be enabled to perform this task. This requires scalable techniques for service discovery and matchmaking at a large scale and in particular a high degree of automation of these and other service-related tasks such as service mediation. At the same time, involving end-users in the SOA process, necessitates effective support to enable this.

Semantic technologies will play a major role in addressing these challenges [BDB05, FFST11]. Thereby, semantic service descriptions that unlike, e.g. a WSDL-document, do not just describe the interface of a service, but also the effect of a service in a machine-comprehensible way have the potential to enable automatic service discovery that goes beyond pure interface matching. They also provide the information that are needed for automatic composition of services, for automatic service binding, mediation and execution as well as for sophisticated end-user support. Though, e.g. with WSMO[7] and OWL-S[8] semantic approaches to service description and matchmaking have been proposed, these have not yet been adopted in practice [BF08]. End-user support for formulating service requirements and user-mediated service selection are hardly considered yet [Agr06].

[5]Universal Description, Discovery and Integration, http://uddi.org/pubs/uddi_v3.htm
[6]http://cordis.europa.eu/fp7/ict/ssai/home_en.html
[7]http://www.w3.org/Submission/WSMO/
[8]http://www.w3.org/Submission/OWL-S/

2.2 Decision Making and Decision Support

Decision making is the process of selecting an option out of a set of alternative courses of action that considering the current state of nature results in an/the outcome that is most valuable for the decision maker. Thereby, the outcome of a decision is not necessarily known in advance, since a taken action might have several possible outcomes depending on the state of nature (decision making under uncertainty). If the probabilities with which possible outcomes occur are known, we speak of decision making under risk [Kni21]. The value of a certain decision outcome depends on the personal preferences of the decision maker, i.e. his utility function. The best choice to be taken in a given decision situation under risk depends on both, the decision makers utility function and his risk attitude. A rational decision maker would then choose the decision that maximizes the expected utility of the outcome [JvN53].

However, human choices are typically far from being rational. This lies in the way humans construct their preferences, the way they process information required to make a decision and in the way they are actually making a choice based on these information. In particular, if a decision task is complex, e.g. in terms of the number of available alternatives or the number of involved attributes, or just appears to be complex, e.g. since the alternatives and their properties are presented in a format that is difficult to read, a wide range of heuristics simplifying the decision process are applied (see [PBS99] for an overview). This is due to the fact, that humans' capabilities as an information-processor are limited [JWP93]. As a result, decision makers typically consider an (unnecessarily) narrow range of alternatives, do not account for/are not aware of all of their objectives and all possible states of nature when making a decision (myopic decision frame). They make highly selective use of information such as focusing on the information that is most salient, but not necessarily most relevant. Moreover, they base their selections on simple heuristics to minimize cognitive effort. In particular, making explicit tradeoffs is avoided. Instead non-compensatory decision heuristics such as selecting the obvious or the alternative that is typically selected are applied. Human decisions are also often guided by other goals than just making accurate decisions, such as reducing negative emotions or easing the justification of a decision [BLP98]. Finally, human choices are context-dependent, i.e. the value of a given alternative depends on the characteristics of other alternatives (similarity of alternatives, dominance relationships between them or the comparability of choices) (see [JWP93] for an overview). As a consequence, human decisions "generally reflect both the decision maker's basic values for highlighted attributes [...] and the particular (contingent) heuristics or processing strategies used to combine information selectively in order to construct the required response to a particular situation" [PBS99]. Though this might be acceptable for decisions that are of minor relevance, it can lead to significant financial losses and/or can have serious impacts on personal or general welfare if it refers to important management or governmental decisions.

Decision Support Systems (DSSs) are a class of computer-based applications that have been developed to improve human decision making by mitigating its shortcomings and thus promise to solve the discussed issue. In particular,

> *"[Decision Support Systems] provide knowledge and/or knowledge-processing capability that is instrumental in making decisions or making sense of decision situations. They enhance the processes and/or outcomes of decision making. A Decision Support System (DSS) relaxes cognitive, temporal, spatial and/or economic limits on the decision maker. The support furnished by the system allows a decision episode to unfold*
>
> - *in more-productive ways (e.g., faster, less expensively, with less effort),*
> - *with greater agility (e.g., alertness to the unexpected, higher ability to respond),*
> - *innovatively (e.g., with greater insight, creativity, novelty, surprise),*
> - *reputably (e.g., with higher accuracy, ethics, quality, trust), and/or*
> - *with higher satisfaction by decisional stakeholders (e.g., decision participants, decision sponsors, decision consumers, decision implementers)*
>
> *versus what would be achieved if no computer-based decision support were used" [Hol08].*

Decision Support Systems provide sophisticated assistance for the single steps to take in order to make decisions in a real world setting, i.e. problem formulation (formally modeling the considered decision problem), evaluation (determining the optimal decision alternative) and appraisal (assessing if the preferable decision is convincing) [How88]. This is achieved in multiple ways, e.g. by providing support for structuring a decision problem at hand, assessing required probabilities, performing a sensitivity analysis, by allowing to assess the value of acquiring additional information, by summarizing information and by presenting them in a format that is appropriate for human processing [DF02]. In particular, Decision Support Systems provide support for making a decision that is reasonable in light of the given information and in light of the values of the decision maker. Thereby, a single Decision Support System typically does not perform all of these tasks, but is domain-specific and focuses on a certain kind of support. Service selection decisions made at a Web-scale are particularly complex and thus require effective decision making support provided by means of a Decision Support System that accounts for the peculiarities of such decisions.

Part II.

Decision Support for Semantic Web Service Selection

3 Overview

The objective of this thesis is to create a system that effectively supports service consumers in efficiently making well-informed, balanced and consistent service selection decisions in the presence of uncertainty (Objective 1). The decision analysis performed in Section 1.4 identified the system components and their main properties that are required to achieve this goal. In this chapter, we will provide an overview about our solution that is based on this system design. We will introduce its main features and their intended usage.

As required by the operational Objective 3, we suggest a solution that implements service selection as an incremental and interactive process that alternates phases of intermediate service recommendation and requirements refinement. During that process, the user incrementally develops his service requirements and preferences and finally makes a selection decision. To effectively support him in these tasks, the system maintains an internal model of the consumer's requirements and preferences, which we call *request model*. Uncertainty about the service consumer's true requirements and preferences, that is caused by missing and/or inaccurate knowledge, is explicitly represented within this model. During the requirements elicitation and refinement process, the request model is continuously updated to accurately reflect the systems's growing knowledge about the user's service requirements and preferences. Inspired by the incremental and interactive recommendation strategy employed in example critiquing recommender systems [Smy07], we propose to implement service selection in a similar fashion. Starting with a request model that is constructed from the user's initial requirements, the user is provided with a set of service alternatives that fit to these requirements. These service alternatives are determined by transforming the internal request model into a semantic service request that reflects the requirements specified in the model, but also take the system's uncertainty about this model into account. As we will demonstrate (Chapter 5), standard matchmaking with a minor extension can be applied to retrieve those matching service results sorted by their expected matching degree. After viewing intermediate service recommendations, the user may critique the presented service alternatives and thereby indicate desirable service characteristics. For instance, he might indicate that a certain service alternative is well suited to his needs, but is too expensive. The system then determines whether there are service offers that are cheaper and if this is the case, which compromise on other service aspects the user has to accept in return. Moreover, the system suggests service aspects and attribute subtypes that have not been considered yet, but which have been specified in the presented service offers and thus might be worth to be taken into account by the user. As we will see (Chapter 8), those facilities serve as a means to educate the user about available service alternatives, their characteristics and necessary tradeoffs and allow him to specify desirable service characteristics in an indirect and intuitive manner. Beside of criticizing the presented service alternatives, the user may view and directly modify the internal request model via a graphical representation. The intention behind this feature is manifold. As we will demonstrate (Chapter 8), it allows the user to correct the system, if necessary, helps him to comprehend and become aware of his requirements and thus enables him to actively develop them. This in turn fosters consistency between the request model and the user's actual requirements and preferences. However, it is unrealistic to assume that a non-expert-user is able to understand the internal request model or even make modifications to it that reflect his changing requirements. To mitigate that problem, we suggest an editable graphical requirements representation, that provides an abstract view to the internal model. This view hides and simplifies different aspects of the model and presents model-intern relationships and dependencies in an intuitive way. Effective visualizations enable the user to understand relevant aspects of the model and qualify him for making appropriate modifications (cf. Chapter 8). To keep the request model up to date, all user interactions trigger appropriate model changes. Once a set of modifications has been taken place, the user may decide to see service results fitting to the updated requirements. The process continues until the user finds an appropriate service among the presented alternatives or until he decides to stop without making a selection. To enable the development of well-constructed preferences and to contribute to the coherence of the refinement process, we advocate that the specification of service requirements, whether direct or indirect, should be user-initiated and not imposed by the system. However, to still being able to focus and direct the refinement process, we suggest to guide the user by emphasizing interaction opportunities that point to promising refinement directions. This leads to both, an effective and efficient, refinement process (Chapter 8). This is due to the fact, that the system leaves the user the opportunity to express those service needs that are important to him, while at the same time encouraging him to focus on those aspects that are relevant in light of the available service opportunities and in light of the service requirements that are already known. Those promising interaction opportunities are determined by leveraging knowledge about the current request model and the uncertainty related to it.

To acquire information about the actual performance of available services, our solution to service selection will provide a flexible feedback system, where consumers may report about their experiences with a specific service (Objective 2). In this context, the focus of our work is on how to take advantage of the full potential of feedback and how to effectively utilize those information to evaluate a service's expected performance. In particular, we will provide means that allow to detailedly describe a service's performance with respect to several service aspects. In addition, we will devise a user-adaptive method that assists the consumer in providing such feedback. As we will demonstrate (Chapter 9), this method supports the user in supplying appropriate, comprehensive and meaningful feedback and

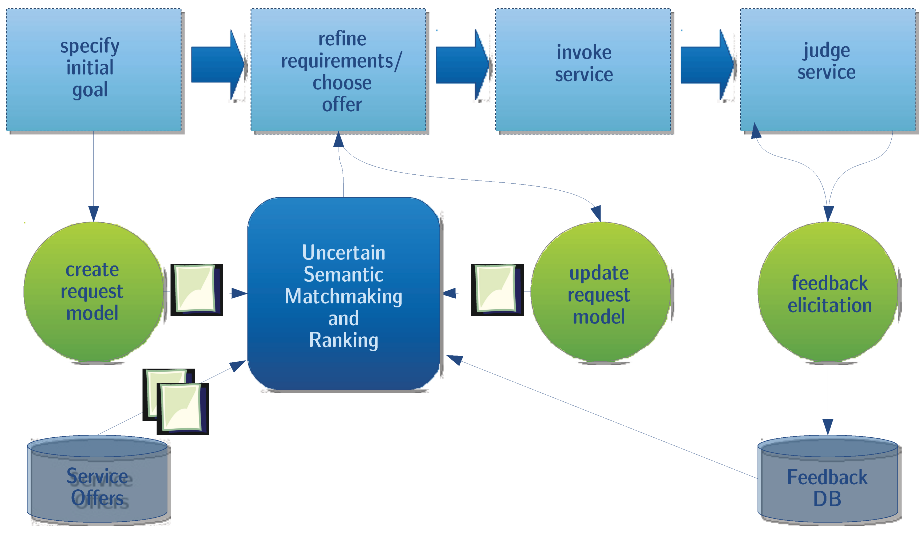


Figure 3.1.: Main features of the suggested solution

thereby adjusts to his willingness to provide feedback. The mechanism also accounts for the privacy requirements of the user by allowing him to customize the detailedness of the feedback information that is shared with others. It makes use of an obfuscation technique to enable the propagation of meaningful feedback while at the same time revealing as little personal information as possible. Finally, available consumer feedback is exploited to assess the degree and kind of uncertainty that is associated with the execution of an offered service and considered during the service ranking process. In this context, the suggested solution accounts for the subjective and context-dependent nature of consumer feedback. It will devise a mechanism that allows to use consumer experiences made in one request context to infer knowledge about the behavior of a service in another context. Finally, it makes the user aware of the risk that is associated with the execution of a service by presenting feedback information in an effective and intuitive way (cf. Chapter 9). Since risk attitudes vary among users and request contexts, the presentation of those information is adjustable. Aspects such as detecting dishonest feedback and counteracting feedback manipulation have been extensively discussed in other research areas, such as trust and reputation [JIB07] and recommender systems research (e.g. [MBBW07]) and are out of the scope of this work.

Since the system is intended to support non-expert users, it does not require special skills or particular learning effort of the user (Chapter 8). To achieve this, technical details about the internal representation of information, e.g. semantic service requests, the matchmaking process or uncertainty management have been hidden from the user. Instead, relevant knowledge about the rationale behind the system's recommendations as well as required information are transferred to an abstract level. This is accomplished by providing explanations and visualizations to present information in a compact and easily perceivable way. Figure 3.1 summarizes the main features of the solution.

In the subsequent chapters, we provide a detailed description of our approach. We will start by introducing required background information about the service description language that underlies our approach (Chapter 4). The two subsequent chapters constitute the main part of our thesis and cover the description of our suggested solution. The presentation is structured along the system's main elements: requirements elicitation (Chapter 5) as well as elicitation and usage of consumer feedback (Chapter 6).

4 Underlying Service Description Language

Considering the operational objectives (Objectives 2 and 3) derived in Section 1.4, we decided to base our solution on the semantic service description language DSD (DIANE Service Description) [KKR04, KKRM05, KKRKS07] and its mechanisms for completely automatic semantic service matching, selection and binding, which were developed within the DIANE project[1] and build up the DIANE middleware. In this chapter, we will therefore briefly introduce DSD and the capabilities of the DIANE semantic matchmaker to provide the foundations for the further discussion (Sections 4.2 and 4.3). In particular, we will expand on the advantages of the chosen service description framework compared to other potential candidates (Section 4.1) and point to the assumptions about the service descriptions that underlie our approach (Section 4.4).

4.1 Choice of the Service Description Language

Though DSD might not be as expressive as prominent logic-based semantic service description models such as OWL-S[2] or WSMO[3], its light-weighted approach turned out to be sufficiently expressive and well-suited to many practical application scenarios[4]. Moreover, it exhibits a number of properties that are desirable in light of the identified requirements. In the following, we will detailedly explain the rationale for our choice by emphasizing those properties.

Ontology-based descriptions Like many other service description models, DSD is ontology-based. More specifically, this means that the concepts used to describe services, valid constraints to them as well as valid relationships among those concepts are defined in an ontology that is shared among the system participants. In doing so,

[1]http://fusion.cs.uni-jena.de/DIANE
[2]http://www.w3.org/Submission/OWL-S/
[3]http://www.w3.org/Submission/WSMO/
[4]http://sws-challenge.org/wiki/index.php/Main_Page#Most_Recent_Aggregated_Certification_Results

the meaning of service descriptions is made explicit and machine-comprehensible. Those properties are basic requirements for any service description language, since they enable independent creation, comparability and computer-based comparison of service descriptions. By uniquely relating consumer feedback to parts of ontology-based service descriptions, it also receives a well-defined meaning. Hence, feedback provided by different consumers can also be automatically compared and related to each other by a computer program. Finally, ontologies provide a powerful instrument for computer-based assistance of humans, e.g. when formulating their service requirements.

Semantic descriptions DSD descriptions characterize a service in terms of its effect(s), i.e. semantically, and not just in terms of its interface. This is in accordance to human thinking, since when looking for service functionality, consumers (in particular those with a non-computer-science background) do not think in terms of inputs and outputs, but rather in goals that need to be achieved. Moreover, often services that provide a required functionality do not exactly offer the desired interface, and hence would not be discovered by a purely interface-based matchmaking procedure.

Besides those properties that the DIANE approach has in common with WSMO, OWL-S and other semantic service description approaches, it also exhibits a number of desirable characteristics that are not shared by other solutions.

Highly structured descriptions Logic-based service description models are very expressive, but give the user too much freedom in creating descriptions. Hence, they often lack the necessary guidance for requirements formulation and thus make the request creation process difficult and error-prone. Moreover, they complicate the comparison of descriptions. They also do not allow for creating service requests and offers independently from each other and independently from a certain service implementation. The latter is an essential requirement in dynamic service environments. In contrast to that, DSD service descriptions are based on a layered stack of ontologies and their graphs are trees. This eases the formulation of service descriptions and at the same time provides an expressiveness that is sufficient for the most real world applications. Moreover, the tree-structure can be effectively explored to enable efficient and automatic matchmaking. In opposition to other semantic service description approaches, the DIANE matchmaker is not overly complex and considers all language elements when comparing descriptions. The structured nature of DSD descriptions induces further desirable characteristics, which will be subsequently discussed.

Availability of partial matching results Due to its graph-based matchmaking approach, the DIANE matchmaker is able to provide partial matching results referring to different aspects of a description. This property makes it possible to explain matching results. In particular, it allows to explicate, why a certain match failed and thus provides a valuable foundation for assisting the user in the requirements formulation process.

Hierarchical descriptions DSD descriptions are hierarchical, i.e. service aspects at the top of the description tree are refined in lower levels of the tree. This description design perfectly fits to the constructive nature of consumer requirements by supporting an incremental refinement process. Moreover, hierarchical descriptions allow to relate consumer feedback to service aspects on different abstraction levels. As we will see, this allows for a comprehensive and at the same time user-adaptive feedback mechanism. Furthermore, hierarchical descriptions allow for the aggregation of consumer feedback that is related to several service aspects up to levels of different granularities. Aggregating feedback information reduces their detailedness and thus enables privacy-aware feedback propagation. Finally, formulating and matching service requirements in a hierarchical manner allows to locate uncertainty about the user's service requirements, i.e. allows to detect knowledge gaps and to identify the type of information that has to be acquired to fill them.

Intuitive description representation DSD provides an intuitive and comprehensible representation of service request and offer descriptions as a tree. This facilitates the knowledge transfer between the system and the user. It thus makes suggestions, that are provided by the system, comprehensible, allows to scrutinize the system and enables the user to provide meaningful responses.

Explicit preference model Service consumers have individual preferences about service characteristics, which have to be considered when matching their service requirements to available service offers and ranking the latter in a personalized way. Moreover, knowledge about a service requester's preferences is required to assess the relevance and thus the value of missing requirements information, i.e. to quantify uncertainty. DSD accounts for those facts, by providing language elements that allow to specify preferences related to different service aspects. Those preferences are considered during service matchmaking and are utilized to individually rank matching services. In this context, it is advantageous that DSD models preferences in a way that is compatible with common preference representations used in multi-attribute utility theory (MAUT) [KR93].

We would like to note that with the WSML variant WSML-MX [Kau06, KKK08] another semantic service description approach exists, that implements recursive matchmaking based on hierarchical descriptions[5]. However, this approach does not model user preferences and hence does not fulfill one of our basic requirements.

After having clarified the reasons for choosing DSD as the service description language that shall underlie our approach, we use the subsequent sections to briefly introduce the DIANE service description language and the DIANE matchmaking approach.

[5]As indicated by the authors, the approach was inspired by DSD and the DIANE matchmaker.

4.2 DIANE Service Description Language

DSD is an ontology-based and purely state oriented description language. The latter means that the functionality a service provides as well as the functionality required by a service consumer is described by means of the required state(s) of the world before (*precondition(s)*) and the resulting/required states after (*effect(s)*) the service execution. In the service request depicted in Figure 4.1, the desired effect is that a product is owned after service execution. In DSD, preconditions and effects are represented by declarative *instance sets*, where a single element of such a set corresponds to a particular precondition required by a service or the effect that is produced by a particular instantiation of a service (*service instance*), respectively. Sets are configurable via variables symbolizing inputs and outputs of a service. In doing so, the relationship between the parameters of a service and its effect(s) is made explicit. DSD descriptions are build on a layered stack of ontologies. While the *upper service ontology* defines the basic structure of service descriptions, as depicted in Figure 4.2, a relatively small set of *category ontologies* categorizes and defines possible states of the world. Finally, a wide range of specific *domain ontologies*, that provide the concepts to characterize services and their properties in the various fields of application, is available. This ontology design forms the foundation for both, a structured and flexible, service description mechanism.

The service descriptions that are created by service providers and service requesters usually differ in their precision. Whereas a service provider describes the service instances he is able to offer, a requester typically does not know about available services and is rather interested in a certain functionality, but not in a particular service. DSD takes this into account by modeling service offers and requests in slightly different, but compatible, ways. *Service offer descriptions* describe the set of service effects that can be produced by a service, e.g. the set of mobile phones that can be owned after executing a certain phone selling service, as well as the preconditions for its correct execution, e.g. the availability of valid credit card information. In addition to that, service offer descriptions specify the information that are required for the service execution and those that are delivered afterwards as a result. This is done by placing appropriate IN(put)- and OUT(put)-variables in the effect description. After service execution, exactly one of the producible effects has been achieved and the promised information are provided.

In contrast to that, *service request descriptions* declaratively characterize the set of service effects that are desirable for the service consumer and specify the information that shall be accessible after the service execution. The latter, is again done by placing appropriate OUT-variables in the request description. In the service request depicted in Figure 4.1, acceptable effects are the ownership of a mobile phone that is cheaper than 50$ (or slightly higher), that is either silver or black, is of bar or slider style and is either from Nokia or Sony Ericsson. However, typically, different service effects are not equally desirable for a service consumer. Therefore, in a service request, the set used to describe acceptable service effects can be fuzzy, where a given effect's membership value directly maps to the user's *preference value* for it. Membership and thus preference values are real-valued and range from 0.0 to 1.0. A membership/preference value of 1.0 indicates that the service effect is contained in the set of desirable effects and is totally acceptable, while a preference value of 0.0 means that the effect is not contained in the effect set and is not acceptable at all. Values between 0.0 and 1.0 indicate acceptable service effects, where a higher value indicates a higher preference, i.e. higher desirability. Preference values induce an order over the available service effects.

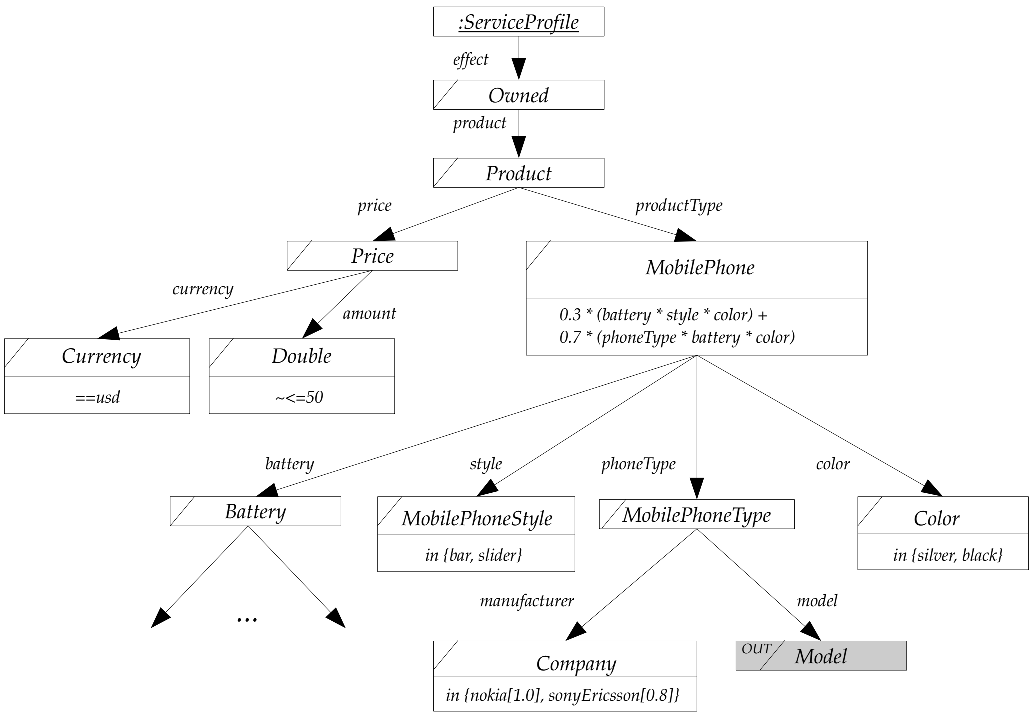


Figure 4.1.: DSD service request

Hierarchical characterization of service effects After having introduced DSD's approach to model services in general, we will particularize on its specific mechanism to declaratively and hierarchically characterize (desirable) service effects and their properties using the notion of an *instance set*. In contrast to a class, which represents all instances of a certain type, an instance set comprises of an arbitrary collection of instances. DSD offers a number of language elements to declaratively define those instances, each representing a service effect. In the remainder of this section, we will give a brief overview about those elements.

In DSD, a set is characterized by its *type condition*, that indicates the ontological type of the instances that are valid members of the set. For instance, the effect set in Figure 4.1 is of type *Owned*. By default, only instances that share the specified type of a set or one of its subtypes can be members of the set. However, DSD allows for alternative crisp (*super, super[n,1]*) and fuzzy (*super[n,f]*) *type check strategies*, which extend the set of valid elements by those having a supertype of the specified type. The type check strategies *super* and *super[n,1]* refer to crisp sets and indicate that valid elements can be of any supertype or of a supertype differing from the specified type by at most n levels in the type hierarchy, respectively. The alternative type check strategy *super[n,f]* is the counterpart of *super[n,1]* for fuzzy sets. It differs from the latter by the fact, that an instance's membership value with respect to its type is 1.0 for instances of the indicated type or one of its subtypes. It is 0.0 for elements that are instantiated from a supertype that differs from the indicated type by more than n hierarchy levels and is f^k, $0 \leq f \leq 1$, for instances of a supertype that differs from the indicated type by $k \leq n$ levels.

A set specification may be supplemented by *direct conditions* on its members and/or additional *attribute conditions*. *Direct conditions* directly constrain the elements of a set. For that purpose, the standard comparison operators $>, >=, <, <=, ==, != $ and their fuzzy counterparts $\sim>, \sim>=, \sim<, \sim<=, \sim==, \sim!=$ can be used. For instance, the direct condition $\sim<= 50$ on the price amount in the example (Figure 4.1) indicates that only prices lower than $50\$$ are acceptable. As expressed by the preceding $\sim$, slightly higher prices are also acceptable, but with a preference value lower than 1.0. As demonstrated for the *Company* attribute in Figure 4.1, the operator in and its fuzzy version allow to directly enumerate the elements of a set. The values in squared brackets indicate the preference value of each element. A set may have any number of direct conditions, each defining a membership function and thus a preference function over the set's elements [6]. In that case, the preference values of the different direct conditions are conjunctively combined, i.e. multiplied.

Sets of non-primitive types may specify conditions on the attributes of its potential instances. These *attribute conditions* refer to valid attributes defined for the ontological type that has been specified in the type condition of the set. They inherit the name of the corresponding attribute defined in the ontology and have a target set they refer to. Both, valid attributes of a type and valid target set types, are specified in the service ontology. An instance can be an element of a given set, if the attributes specified in the attribute conditions of the set are specified for that instance and their values are elements of the corresponding target sets. A *missing strategy* specified for a given attribute condition modifies this semantics. It allows to assign a predefined, non-zero preference value with respect to an attribute condition to instances that do not specify the attribute required by this condition. The predefined preference value is 1.0, if the missing strategy *assume_fulfilled* has been specified. It is n, if the missing strategy *assume_value[n]* has been specified and it is ignored[7], if the strategy *ignore* is applied. Attribute conditions allow for the nested specification of declarative sets and induce a hierarchical, more and more fine-grained characterization of (desirable) service effects. For instance, the set *MobilePhone* in Figure 4.1 is constrained

[6]For sets of primitive types (*Integer, Double, String, Boolean, Date, Time, DateTime, Duration*), all operators are defined. As an exception *Boolean* and *String* do not support fuzzy comparison operators. For sets of non-primitive types, the operators $==, !=, in$ and fuzzy in are allowed.

[7]We refer to [KKRM05] for further details.

by conditions on its attributes *battery*, *style*, *phoneType* and *color*. The set *MobilePhone-Type* characterizing the attribute *phoneType* is constrained in turn by conditions on its attributes *manufacturer* and *model*. The (defining[8]) attributes of a set uniquely define its elements. For example, in Figure 4.1, all mobile phones with an attribute configuration from the set $\{bar, slider\} \times \{nokia, sonyEricsson\} \times \{silver, black\}$ are contained in the set *MobilePhone*. Moreover, the target sets of the attribute conditions assign a preference value to each of its elements. However, how are those values resulting from a set's attribute conditions combined to a preference value for the elements of the conditioned set? For instance, how are the preference values for the elements of the set *MobilePhoneType* in Figure 4.1 derived from each element's preference value resulting from the evaluation of its attributes *manufacturer* and *model*? By default, the preference values resulting from the evaluation of an instance's attribute conditions are conjunctively combined, i.e. multiplied. For example, in Figure 4.1, for any mobile phone from Nokia, the preference value with respect to the set *MobilePhoneType* is 1.0 (with respect to *Company*) $\cdot$ 1.0 (with respect to *Model*) $= 1.0$. However, a set may specify an alternative *connecting strategy*, that indicates how the preference values resulting from the evaluation of the single attribute conditions shall be combined. It thus encodes the preferences of the user with respect to the relative importance of the specified attribute condition. Thereby, any syntactically correct mathematical expression over the attribute conditions of a set, that can be created using the operators *add, multiply, min, max, weighted sum* or *power* is a valid connecting strategy. As an example, consider the attribute *productType* in Figure 4.1. The depicted connecting strategy applies the weighted sum operator to the product of the attributes *battery*, *style* and *color*, and to the product of the attributes *phoneType*, *battery* and *color*. It expresses that the user prefers a certain mobile phone type (indicated by the weight 0.7), but that he is also willing to accept any other mobile phone with lower preference (weight 0.3), if it has the indicated style. *Color* and *battery* of the phone are important in any case. Operators applied to preference expressions result in a combined preference value from the interval $[0, 1]$.

Finally, the target set of an attribute condition can be a variable, which is a special set that needs to be filled with a value. That value has to be a valid element of the set. After filling, a variable behaves like a standard DSD set (comprising of a single element). As mentioned, DSD distinguishes between IN(put)-variables, which have to be filled before service execution, and OUT(put)-variables, that are filled after service execution. In service offers, IN-variables serve as a means to provide the service with the information that are required for its execution. OUT-variables in service offers indicate output information that can be delivered by the service. In a service request, OUT-variables (such as the attribute *model* in Figure 4.1) are used to indicate the information need of the service consumer, while IN-variables can be used to create parameterized requests.

DSD also supports a number of additional language elements to characterize sets (see [KKR04, KKRM05, KKRKS07] for more information). However, in the context of our

[8]Two instances that share the same values with respect to their defining attributes are identical. The values of any other attribute of an instance can be inferred from the values of its defining attributes.

work, we assume that service descriptions are exclusively based on the described elements. In particular, we do not allow for language elements that break the tree structure of DSD descriptions, e.g. by introducing dependencies between service aspects such as multi attribute conditions [KKR09]. This is due to the fact that the strictly hierarchical structure of service descriptions is a basic prerequisite at several points of our solution.

The general structure of service requests and offers Having elaborated on the language elements that DSD uses to characterize service requirements and service offers, we provide a brief overview about the structure of DSD service request and offer descriptions. The basic structure of DSD descriptions is depicted in Figure 4.2. As already mentioned, the graph of a DSD description is a tree. The root element of this tree is an instance of type *Service* representing the description as a whole. Its attributes indicate the name of the service provider[9], the *ServiceProfile*, which provides an abstract description of what the service does/is required to do, and the *ServiceGrounding*[9], mapping the abstract description of the service in the *ServiceProfile* to the actual, executable functions that make up the service and indicates how they can be accessed. The *ServiceProfile* constitutes the main part of a service description. It characterizes the precondition(s)[9] (if any) and the (desired) effect(s) of a service in a hierarchical fashion using the mechanisms detailed above. Thereby, the root of the effect description is given by the effect-operator that points to a set of type *State*. A service description might be supplemented by a description of (desired/offered) non-functional service aspects. In case of a service offer description, the *ServiceProfile* also specifies the name of the provided service.

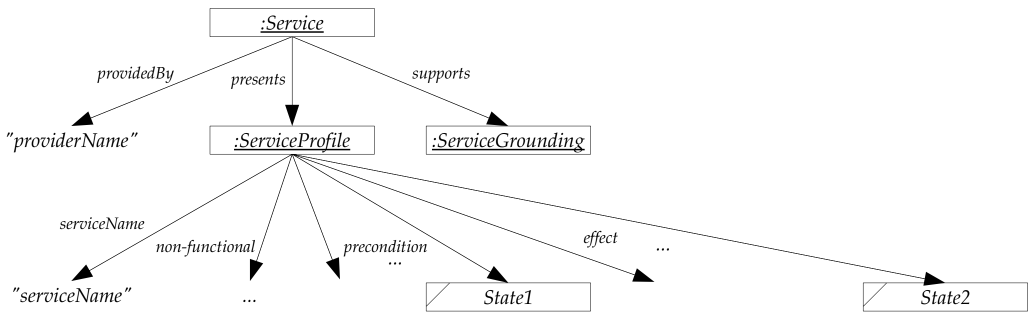


Figure 4.2.: DSD service descriptions - basic structure

4.3 DIANE Matchmaker

As demonstrated in [KKRKS07], DSD service and request descriptions can be efficiently compared in a completely automatic fashion. Given a service request, the DIANE semantic

[9]only in service offer descriptions

matchmaker determines a/the offered service or a combination of services that best fits to the requirements which have been specified in the request. How is this achieved? Since a request describes an ordered set of desirable service instances (by means of their effects) and a service offer describes a set of provided service instances (from which the service provider can choose one to provide), the matchmaker has to check if the offered instances are contained in the instance set that is described in the request. As suggested by the tree structure of DSD service descriptions, the comparison of the effect(s) (precondition(s)) described in the request and those described in the offer description is recursive and guided by the request. Starting from the root set of type *State* (or one of its subtypes) referenced by the effect-operator of the request, the matchmaker checks in each step, whether the instances (service effects) described in the offer fulfill the conditions in the request. To illustrate the procedure, imagine that the request depicted in Figure 4.1 is compared to a given offer. The matchmaker would first check whether the effects described in the offer match to the type *Owned* as indicated in the request. If this is true, the attribute condition *product* is checked. Recursive proceeding to the leaves of the request results in a mismatch or match. In case of a match, a *matching degree* or *matching value* with respect to each attribute condition is returned. These values indicate how well the considered service offer fits to the required functionality as described by those attribute conditions. The matching degree with respect to a certain attribute condition is computed as the lowest value among the offered instances' membership (preference) values with respect to that attribute condition. Thereby, the single membership values are determined as the product of the preferences resulting from the type condition, the direct conditions and the attribute conditions that have been specified for the considered attribute condition's target set. In a final pass from the leaves to the root of the effect description, those matching degrees are aggregated as indicated by the connecting strategies specified for the intermediate (i.e. non-leaf) request nodes. The comparison algorithm outputs an aggregated matching degree from $[0, 1]$ for the offer. That is, a lower bound for the preference value the user would assign to the offered service, if executed. This is reasonable, since we do not know in advance which of the offered service instances will be selected for execution by the service provider. An advantage of the described approach is, that during the recursive calculation of the (overall) matching degree, an aggregated *partial or intermediate matching degree* for each of the attribute conditions that has been specified in the request description is computed.

While the described procedure can be directly applied to offers that do not contain any variables, offers that are parameterized with IN-variables need to be configured first, by filling its variables. The matchmaker automatically determines an optimal variable assignment resulting in the highest possible matching degree. Service offer descriptions that offer more than one service instance fitting to a particular configuration are called *ambiguously specified*. In some cases, it is infeasible or undesirable to provide detailed information about all offered service instances within the service offer description, e.g. if the number of offered instances is large or frequently changes, such as the trips that can be booked via a travel agency. As a result, a valid configuration of such an offer might not necessarily point to an existing service instance. Hence, though matchmaking based on those *underde-*

termined offer descriptions can determine the best fitting service configuration(s), it cannot guarantee that the service configured this way is executable. To counteract that problem, the service matchmaker can gather additional information about an offer during the matchmaking process. For that purpose, the matchmaker needs to fill specific IN-variables of the offer (so-called *estimation variables*) with information about the user's particular service request and in return receives more detailed information about the offered service instances provided via specific estimation OUT-variables of the offer. The matchmaker might perform several of those *estimation steps* to acquire all required information.

Based on the matching degrees returned by the matchmaker, fitting service offers might be ranked and presented to the service consumer for selection. The service(s) corresponding to a/the selected offer(s) may then be invoked by the DIANE middleware without requiring additional human intervention (if the preconditions are fulfilled). After the service execution, the matchmaker uses the information provided in the configured service offer description and the filled OUT-variables of the offer to fill the OUT-variables of the request with the information desired by the user.

For a detailed discussion of DSD and the DIANE matchmaker we refer to [KKR04, KKRM05, KKRKS07].

4.4 Assumptions

The basic assumption underlying any predictive algorithm is that the events that have been observed in the past are equal to or at least similar to the events, which will occur in a similar situation in the future. That is, to unfold their predictive power, those algorithms require (relevant) knowledge about the circumstances of the past observations that are leveraged for making a prediction. This equally applies to any procedure that makes predictions about a service's future performance based on consumer judgments referring to the outcome of past service invocations, as will be proposed in this thesis. This implies, that meaningful predictions about the future performance of a service can only be made, if the service instance a consumer judgment refers to is known. This is only true, if we presume, that any available service offer is either *uniquely specified*, i.e. for any configuration of this offer, the service instance that will be executed by the service provider can be uniquely identified, or if the service instance that has been executed can be inferred by leveraging the information that is provided by the filled OUT-variables of the user's service request.

Assumption 4.1. (**Availability of information about an executed service instance**) *Let i be any service instance that has been invoked by a service consumer. We presume that, at the time the outcome of this invocation is judged by the service consumer, the semantic description of this instance is known to the system.*

While the DIANE-matchmaker was originally intended to enable completely automatic service discovery, selection, binding and invocation based on known service requirements, the approach proposed in this thesis aims at the elicitation of unknown, yet to be constructed service requirements and user-mediated service selection. This requires detailed information about available service offers and their properties, since as argued in Section 1.4.1, users construct their preferences and requirements over time when facing choices to be made. Hence, we presume that the required information about potentially matching service offers are available to the user/matchmaker. In particular, we assume that any available service instance is either completely determined (i.e. the corresponding service offer description is not underdetermined) or required knowledge about the service instance can be completely acquired during an estimation phase. Looking at contemporary online platforms for flight/trip booking (e.g. Expedia.com) or product purchasing (e.g. Amazon.com), which already fulfill this requirement, this assumption is not particularly restrictive.

Assumption 4.2. (**Availability of detailed information about available service instances**) *Let i be any available service instance that might be suitable to the service requirements of a consumer. We presume that detailed knowledge about the properties of this instance in terms of a semantic description of its attributes is available to the system.*

5 Modeling and Elicitation of Consumer Requirements for Service Selection

This chapter introduces the requirements elicitation and service selection process that is part of our approach. As argued in Section 1.4, such a component is an essential part of any solution that aims at providing effective support for service selection. However, so far we owe an analysis of the conditions under which it will both, effectively support service consumers in making well-informed, balanced and consistent service selection decisions and enables them to do this efficiently (Objective 3). This chapter starts with the missing analysis (Section 5.1), followed by a thorough investigation of related research efforts and their limitations with respect to the identified requirements (Section 5.2). The remainder of the chapter is dedicated to the presentation of the devised requirements elicitation and service selection mechanism. After outlining the basic idea of the developed solution (Section 5.3), we will describe our internal request model (Section 5.4) and its graphical representation (Section 5.5). We will also explain how suitable service offers can be retrieved based on this model (Section 5.6), how they can be appropriately displayed and how the user can be enabled to indicate desirable service characteristics based on the presented alternatives (Section 5.7). Finally, we will argue how the requirements elicitation and service selection process can be directed and focused to contribute to its efficiency (Section 5.8) and elucidate how the request model is updated based on the interactions of the user (Section 5.9). The contributions of this chapter have been partially published in [KKR11].

5.1 Requirements

The goal of this thesis is to develop a system that effectively supports consumers in making well-informed, balanced and consistent service selection decisions (Objective 1). To be able to perform this task, the system requires a **model of the user's service requirements and preferences**. While the former allow to identify appropriate service offers, the latter enable the comparison and ranking of those service alternatives according to their suitability. To allow for effective matchmaking, the mechanism that is used to model service requirements **should provide means to semantically and richly describe desired service effects**. Since service requirements might refer to various service domains, that mechanism should also be **flexible enough to represent requirements related to different target domains**.

However, as noted earlier (Section 1.4.1), research results from behavioral decision theory indicate, that service consumers often do not have clear-cut requirements and preferences when looking for service functionality. People rather construct them instantaneously when facing choices to be made [PBJ92, JWP93, Slo95]. This particularly holds for decision situations that are complex and unfamiliar [JWP93], such as those arising in a service selection scenario. As March argues, the reason for this might be that humans lack the cognitive resources to precompute and store requirements and preferences for many decision situations [Mar78]. As a consequence, a system that supports users in making service selection decisions cannot merely elicit existing requirements and preferences from the user, it rather has to interactively acquire and incrementally refine those information as they are constructed by the user. Moreover, since service consumers construct their requirements and preferences when being faced with available service alternatives, the two processes of requirements elicitation and service selection cannot be separated, they rather have to be interwoven into a process of **incremental requirements elicitation and service selection** that alternates phases of **intermediate service recommendation based on partially known requirements** and **requirements refinement based on the presented service alternatives**. A system that is intended to assist users in making service selection decisions should therefore have the ability to provide this kind of service recommendations and should support the described way of specifying service requirements and making a service selection.

Due to the constructive nature of consumers' service requirements and preferences, the system's as well the user's knowledge about them is typically incomplete. Furthermore, it is potentially inaccurate. The reasons for this are twofold. On the one hand, the user might provide inconsistent information. On the other hand, the system's knowledge about the user's requirements and preferences is not directly drawn from the user's mind, but derived by interpreting human choices and other input and thus is potentially incorrect. As a consequence, **the fact, that the system cannot be sure about its knowledge about the consumer's requirements, should be considered when making service recommendations and when providing personalized assistance**. To enable this, **uncertainty about the consumer's actual service requirements and his preferences has to be explicitly represented, measured and located within the requirements model**. As we will see later on, this knowledge is also essential for identifying knowledge gaps and actively closing them by acquiring missing information.

Indispensable for any system, that is supposed to provide meaningful recommendations and personalized assistance with service selection, is its ability to **maintain a requirements model that is consistent with the actual requirements developed by the user**. This means, that the system has to be aware of all the service requirements established by the user so far and correctly reflects them. Likewise, it is required that the **user is aware of all his requirements**. This is to ensure that the user is able to comprehend and accept service recommendations made by the system (see [CP02] for similar ideas) and is essential for enabling him to make well-informed and consistent service selection decisions. However, absolute consistency between the user's mental requirements model and its counterpart maintained by the system will never be achieved. Hence, the **user himself should be involved in the model construction process and should be enabled to interactively contribute to the quality of the system's requirements model** [Tin07, NM90, Shn98]. This could for instance be done by allowing him to directly adjust model parameters or by allowing him to indirectly modify the system's requirements model, e.g. by critiquing the service alternative(s) proposed by the system (scrutability of the model) and would give the user a feeling of control and thus would contribute to his confidence in the system's abilities [Shn98]. However, service consumers are typically not able to formulate their service needs in a formal or semi-formal service description language. Offering an effective model visualization and informal interaction opportunities would enable them to do so [NM90]. Visualized service requirements would also make already constructed preferences more perceivable for the user and thus would **contribute to the user's comprehension and awareness of his service requirements**. This is essential for making well-informed and consistent service selection decisions, since evaluating available service alternatives based on vague and ill-defined requirements will cause service consumers to make choices that are based on irrelevant values which are not in compliance with their actual objectives [PBS99, Kee92].

As indicated in [PBS99, Kee92] (see also Section 1.2), thoughtful selection decisions are characterized by the fact that they are made in awareness and consideration of all the service requirements and preferences that are important to the decision maker and relevant in light of the available service alternatives. Moreover, they should result from a process, where conflicts between requirements have been resolved by making explicit tradeoffs instead of being based on the application of non-compensatory heuristics as often performed by decision makers. Hence, as argued in [PBS99] and [Kee92] and empirically shown in [Che08], the effectiveness of the requirements elicitation and refinement process is largely determined by the system's ability to **provide incentives for thinking about and subsequently expressing preferences and requirements** and by its ability to **encourage decision makers to think thoroughly about and finally make tradeoffs**. This means not only to stimulate the user to make preferential statements and express his service requirements, but also to make them correctly. In particular, it is not advisable to prompt or enforce a consumer to provide preferential statements he is not able or not willing to provide, e.g. because he simply does not (yet) have a certain requirement. Hence, we advocate that **expressions of preference and requirements should be user-initiated** and not predetermined by the system (see [VPF08, Kee92] for similar thoughts).

At the same time, **the process of service selection should be efficient**, i.e. made within an appropriate period of time and with adequate mental effort (cf. Definition 1.2). To ensure this, the process of requirements and preference elicitation should be directed and focused in a way that **effectively reduces uncertainty about the service consumer's requirements**. However, in this context it is important to **focus on parts of the model that are relevant in light of the available service options and in light of the user's requirements and preferences**. Consider for example a ticket booking scenario. If all available services offer free delivery of the purchased tickets, then there is no need to know whether the user would also accept some fee for delivery. As well, if the price is not relevant for a consumer's decision, it is not useful to explore in detail which prices are more desirable for this user. Results from critiquing-based recommender systems indicate that purely user-initiated approaches lack this ability [Che08]. Instead, **user-initiated preference expression paired with effective guidance** will enable both, efficient requirements elicitation and well-constructed requirements [Che08]. However, effective guidance of the refinement process **should exhibit some sort of coherence**, i.e. should adjust to the user's intention and the context of the task at hand rather than forcing the user to follow a certain sequence of requirements specification.

Definition 5.1. (**Coherent guidance**) *Guidance is coherent, if it adjusts to the user's intention and the context of the task at hand.*

This is essential for enabling the user to understand and adjust to the refinement process and thus motivates and enables him to provide useful information.

In their article on a building code for well-constructed preferences, Payne et al. [PBS99] emphasize that, in addition to stimulate users to reveal preferences, it is crucial to **educate them about relevant service alternatives and their characteristics and to motivate them to consider this knowledge when making a selection**, i.e. to make well-informed decisions (cf. Definition 1.1). This is required, since consumers tend to base their decisions on a narrow range of options (myopic decision frame) without considering relevant and potentially more desirable service alternatives.

Results from behavioral decision theory also suggest that the main reason for making suboptimal decisions lies in the fact that humans' capabilities as an information-processor are limited [JWP93] and hence, a wide range of heuristics simplifying the decision process are applied. To counteract this phenomenon, they argue that important information should be appropriately presented to the user to reduce mental effort. This particularly means that **decision support systems should emphasize relevant and important information** to avoid information overflow and **should present them in a format that makes them easy to perceive and easy to comprehend** [PBS99].

Finally, it might happen that although a service consumer has well-constructed preferences and requirements in mind, he fails in making an optimal decision. A system that assists users in making thoughtful selection decisions should avoid that by **supporting the user in making a selection that is consistent with his requirements, i.e. optimal or close to optimal with respect to the user's requirements and preferences and the available service alternatives** (cf. Definition 1.1). According to [PBS99], the major causes for inconsistent selection decisions are compatibility issues between the scale of an aspect of the user's requirements and the corresponding attribute of the service alternatives as well as cognitive biases in scale usage such as anchoring effects (tendency of humans to overly rely on a certain, "anchored" aspect). As a means to mitigate those undesirable effects, Payne et al. [PBS99] suggest that decision support systems should facilitate the easy comparability of requirements and service alternatives and thus allow decision makers to correctly translate their preferences and requirements into a selection, i.e. to make a consistent selection without any biases.

The identified requirements to the requirements model, the requirements elicitation and service recommendation process as well as to the presentation of information constitute the criteria with respect to which the quality of the user-modeling and service selection mechanism that will be introduced in the subsequent sections will be measured (Chapter 8). They can be summarized as follows:

Requirements to the Requirements Model

Requirement U.1. (**Model contents**) *The requirements model should comprise of knowledge about the consumer's service requirements and his preferences.*

Requirement U.2. (**Descriptive power**) *The mechanism used to model service requirements should provide means to semantically and richly describe desired service effects and should be flexible enough to represent requirements related to different target domains.*

Requirement U.3. (**Model uncertainty**) *Uncertainty about the consumer's actual service requirements and his preferences should be explicitly represented, measured and located within the requirements model.*

Requirements to Requirements Elicitation and Service Recommendation

Requirement U.4. (**Service recommendation**) *The system should be able to provide service recommendations and personalized assistance based on uncertain requirements.*

Requirement U.5. (**Model construction**) *The processes of requirements elicitation and service selection should be unified and incremental.*

Requirement U.6. (**Incentives**) *The system should provide incentives to think about and to express (correct) preferences and requirements and should encourage decision makers to think thoroughly about and to make tradeoffs.*

Requirement U.7. (**Requirements specification**) *The system should allow for requirements specification based on presented service alternatives.*

Requirement U.8. (**Requirements awareness**) *The system should contribute to the user's comprehension and awareness of his service requirements.*

Requirement U.9. (**Model Consistency**) *The system should maintain consistency between the requirements model and the user's actual service requirements and preferences.*

Requirement U.10. (**User involvement**) *The user should be involved in the model construction process and should be enabled to interactively contribute to the quality of the system's requirements model.*

Requirement U.11. (**Uncertainty reduction**) *The system should effectively reduce uncertainty about the consumer's service requirements. It should thereby focus on parts of the model that are relevant in light of the available service options and in light of the user's requirements and preferences.*

Requirement U.12. (**User-initiated actions**) *Expressions of preferences and requirements should be user-initiated and should be effectively guided by the system.*

Requirement U.13. (**Process coherence**) *Guidance should result in a process of requirements elicitation and service selection that is coherent.*

Requirement U.14. (**User education**) *The system should educate users about relevant service alternatives and their characteristics and motivate them to consider this knowledge when making a selection.*

Requirement U.15. (**Selection efficiency**) *The process of service selection should be efficient, i.e. made within an appropriate period of time and with adequate mental effort.*

Requirements to the Presentation of Information

Requirement U.16. (**Information presentation**) *The system should emphasize relevant and important information and present them in a format that makes them easy to perceive and easy to comprehend.*

Requirement U.17. (**Selection consistency**) *The system should support the user in making a selection that is consistent with his requirements.*

5.2 Related Work

In this section, we will analyze related research efforts with respect to the list of requirements that has been compiled in the last section and identify open research issues. We focus our analysis on recommender systems (Section 5.2.1), selected approaches to utility elicitation (Section 5.2.2) and Semantic Web Service selection (Section 5.2.3). We conclude with a brief summary of the analysis results (Section 5.2.4).

5.2.1. Recommender Systems

Recommender systems are closely related to Semantic Web Service based retrieval, both in purpose and functionality, and hence should be discussed in this section and be compared with our requirements. Similarly to Semantic Web Service retrieval, recommender systems are designed to assist users in finding items of interest in domains with huge and complex item spaces, such as in an e-commerce scenario. They guide consumer decisions by providing item recommendations. In the last 20 years recommender systems have been an active area of research resulting in a huge number of approaches employing many different recommendation strategies. According to the source of knowledge they exploit to make recommendations, existing systems can be classified into collaborative filtering, content-, demographic-, utility- and knowledge-based systems [Bur02, Bur07]. The appropriateness of a certain type of recommender system for a certain recommendation task strongly depends on the characteristics of the application domain's users and items [SFHS07, Bur02, Zan09]. The strengths and weaknesses of each technique in different scenarios, such as the cold-start problem in collaborative filtering systems, are well-known. Recently, so called hybrid recommender systems, that try to combine several recommendation techniques to improve recommendation performance, gain more and more interest [Bur02, Bur07].

However, our evaluation of related research efforts in the area of recommender systems will focus on utility- and knowledge-based systems, since similarly to Semantic Web Service solutions, they employ explicit domain knowledge to richly describe items and consumer requirements. Moreover, they focus on similar application domains. In this section, we will first describe the basic architecture of those systems and subsequently explain how they operate. In particular, we will provide a brief overview about the techniques usually applied to represent knowledge about items and user requirements and will explain how this knowledge is used to generate recommendations. Moreover, we look at how existing approaches construct requirements models and analyze their strengths and weaknesses.

Utility- and knowledge-based recommender systems are particularly suitable for domains, where items are relatively heterogeneous and richly described, where user needs frequently change and the number of items usually exceeds the number of users [SFHS07]. They rely on an explicit model of a user's requirements and his preferences as well as on an explicit model of the available items to generate recommendations. Instead of eliciting customer requirements directly in a preceding interview, they typically operate as *conversational recommender systems* [Smy07] that construct a consumer model step-by-step during an iterative and interactive recommendation dialog with the user.

Knowledge Representation and Recommendation Strategy

Conversational recommender systems originate from conversational case-based reasoning and hence most of the approaches in this area implement case-based recommendation. A second line of research employs constraint-based techniques to generate recommendations. Finally, utility-based recommenders are a third group of approaches, that can be seen as an extension to constraint-based methods.

Case-based recommenders [Smy07, LR05] rely on items (cases) represented by a well-defined set of attributes with values in a well-defined domain. Those attributes represent the properties of the items. User requirements (user queries) may not have this structured form and may take different shapes. Recommendations are generated based on domain-specific similarity knowledge that allows to assess the similarity between items and the user query. Similarity is usually assessed at the attribute level and then aggregated to identify the most similar items for recommendation. Similarity knowledge is typically created manually or automatically learned. Due to their similarity-based retrieval strategy, case-based recommenders often faced the problem of providing the user with resembling recommendations and thus failed in presenting real alternatives. Several methods improving the diversity of recommendations while preserving relevance to the query have been proposed since then to mitigate this problem (see [Smy07] for an overview).

Similar to case-based recommender systems, items in constraint-based recommenders [FB08] are modeled as a set of attribute variables, whose possible values are well-defined. Item characteristics are encoded via constraints on those variables, that restrict possible instances, e.g. the attribute variable *price* of type integer might be restricted to 20$ for a certain item. Similarly, user requirements are represented by constraints over a set of requirements variables. Compatibility constraints restrict possible instances of those variables and thus ensure the consistency of consumer requirements. If, for instance the value of the requirements variable *price* is smaller than 100$, then the value of the requirements variable *quality* may be at most 'medium'. The relationship between consumer requirements and items is encoded by filter constraints. More specifically, filter constraints are rules that describe which item variable values are consistent with which requirements variable values of the user. For instance, a filter constraint might indicate, that the value of the item variable *price* has to be lower or equal to the value of the requirements variable *maxprice*. The task of recommendation calculation can be seen as a Constraint Satisfaction Problem [Tsa95], where those items are recommended, whose item constraints allow a value assignment to the user and item variables that is consistent with the item constraints, the customer's requirements constraints and the filter constraints.

A limitation of the constraint-based approach is that it does not provide an ordering relation over the recommended items. Utility-based recommenders mitigate this problem by allowing recommendations to be ordered according to their utility. For that purpose, dimensions of user interest are identified, e.g. dimensions corresponding to the requirements variables. Multi Attribute Utility Theory [Cle91] is then used to calculate the utility, i.e. the fitting degree, of an recommended item with respect to the consumer requirements.

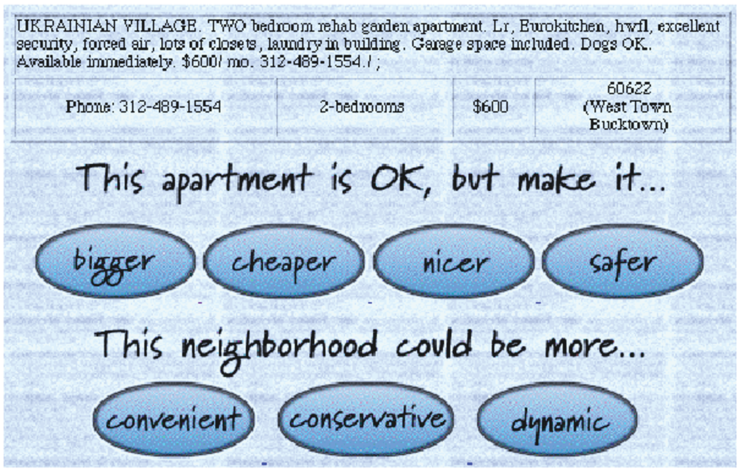
Knowledge Acquisition

As noted earlier, most knowledge-based recommender systems today are implemented as conversational recommenders. Those systems engage the consumer in a dialog, analogously to a real world sales conversation. During this conversation the system guides the user through the item space and incrementally constructs and refines a model of the user's requirements. Conversational recommender systems employ two major types of conversational styles. Adopting the classification of Shimazu [Shi01, SSN01, Shi02] we can differentiate between Navigation-by-asking and Navigation-by-proposing. Systems supporting Navigation-by-asking ask the user a series of questions to learn more about his requirements and preferences. The asking sequence can be defined by the user, who picks out interesting item attributes on his own, or can be defined by the recommender system. Order, number and type of the questions that are asked strongly influence the effectiveness and efficiency of those approaches [BC02, Smy07]. Question selection techniques range from entropy-based methods, that effectively reduce the number of possible recommendations by asking questions with a high information gain, to solutions that employ more user-centered question orderings and/or provide natural language support. Moreover, a number of approaches, so-called mixed-initiative systems, have been developed that combine different styles of conversation and feedback acquisition [Smy07, FB08]. In our opinion Navigation-by-asking has two major drawbacks. On the one hand, questions are abstract and not related to a concrete item. This might make it difficult for consumers to answer those questions and limits the user's possibilities to learn more about the item space and to build a mental model of his requirements and preferences. On the other hand, leaving the user mainly in a reactive position without explaining why the system asks certain questions and how these will help to select an item, does not motivate the user to answer those questions. Moreover, due to the lack of system transparency the user might loose his confidence in the system's ability to be helpful in the item selection process.

Systems adopting a Navigation-by-proposing style learn more about a user's requirements by engaging him in sequence of recommendation phases. In each round, the user is provided with some interim recommendations (sample items). He is then asked to provide feedback about the presented items. Those feedback information are used to revise the system's knowledge about the user's requirements and to provide better recommendations in the next round. The process finishes as soon as the user found the desired product among the recommendations. According to Smyth [Smy07], two major types of feedback can be distinguished. One type which requires only little knowledge and mental effort by the user is preference-based feedback, where the user simply states a preference of one item over the others. However, this advantage comes at a cost, since it is often not clear why a user preferred an option. Hence, the usefulness of this type of feedback for learning a model of the user's requirements is limited. A type of feedback that has been shown to effectively guide the user's navigation and selection process [CP06], while at the same time requiring limited domain knowledge from the user, is critique-based feedback. A critique is a directional constraint over one (unit critique) or several item attributes (compound critique) applied to the intermediate recommendations of the recommender system and serves as a recommendation filter in the next round. For instance, when looking for a computer, the compound critique "cheaper and faster" with respect to the attributes *price* and *CPU speed* of a certain recommended item, indicates that the user is interested in computers that are cheaper than the recommended item and have a higher CPU clock speed. In the context of those iterative, critiquing-based recommendation strategies, called example critiquing, compound critiques seem to be very promising and are an active area of research. This is particularly due to two properties. Firstly, compound critiques allow for constraints on multiple features at a time [CP06] and thus facilitate the effective acquisition of requirements information [CP06]. Secondly, by presenting applicable critiques, the system provides the user with explanatory feedback that makes dependencies between item or product features explicit [PC06, MRMS04]. Hence, in the subsequent analysis of related research efforts from the area of recommender systems, we will focus on example critiquing approaches that use compound critiques. We center our discussion on representative solutions that mark cornerstones of the progress in this area.

Representative Approaches

Burke et al.'s FindMe approach [BHY96, BHY97] was the first that used example critiquing as a means to assist the user in searching and browsing complex product spaces. He implemented several variants of his approach for different domains such as cars (CAR NAVIGATOR), movie videos (VIDEO NAVIGATOR, PICKAFLICK), rental apartments (RENTME), restaurants (ENTREE) and for configuring home audio systems (KENWOOD). In his approach, a user's product requirements as well as product characteristics are modeled as a set of low-level features. In contrast to this, user requirements are specified on a more abstract level, e.g. by allowing the user to provide an example product that is close to his needs or by allowing him to choose a suitable category of products. This relieves the user from having to deal with low-level features. In order to find products that meet the user-specified requirements, the system maps the abstract requirements definition to a set of features. It then retrieves all items that contain one or more of those features. Finally, a hierarchical sort of the retrieved items is performed. In the course of this process, items are iteratively ranked according to the user's preferences. Preferences are predefined within the system and are expressed as goals that refer to one low-level feature, such as "as cheap as possible". For each goal, the system provides a discrete domain-dependent similarity metric that compares items according to this goal. The approach assumes a fixed and ordered list of goals, e.g. in the restaurant finder ENTREE the order *cuisine* $>$ *price* $>$ *atmosphere* $>$ *quality* is assumed. Items are then iteratively sorted according to those goals starting with the most important one. The process finishes when all goals have been applied or the retrieved items are totally-ordered. Once the retrieved items are ranked, the user is asked to choose an item whose properties are close to the required. The user may now refine the results by restricting the viewed items to those that have similar values with respect to a fixed feature of the chosen example (VIDEO NAVIGATOR) or by critiquing the selected item. The user may choose from a set of critiques given in natural language (cf. Figure 5.1), which can refer to a single, but also to more than one feature, e.g. a sportier car. Critiques are used to filter the retrieved items in the next recommendation phase. Unfortunately, the proposed critiques or tweaks as Burke calls them, are not dynamically generated at runtime, but predefined for a fixed type and set of products. They are designed by a knowledge engineer for a single possible use. Hence, it may happen, that after filtering no results remain and thus query relaxation techniques have to be employed. In addition, predefined tweaks are not adaptive to the user's requirements. Another draw-


UKRAINIAN VILLAGE. TWO bedroom rehab garden apartment. Lr, Eurokitchen, hwfl, excellent security, forced air, lots of closets, laundry in building. Garage space included. Dogs OK. Available immediately. $600/ mo. 312-489-1554./ ;

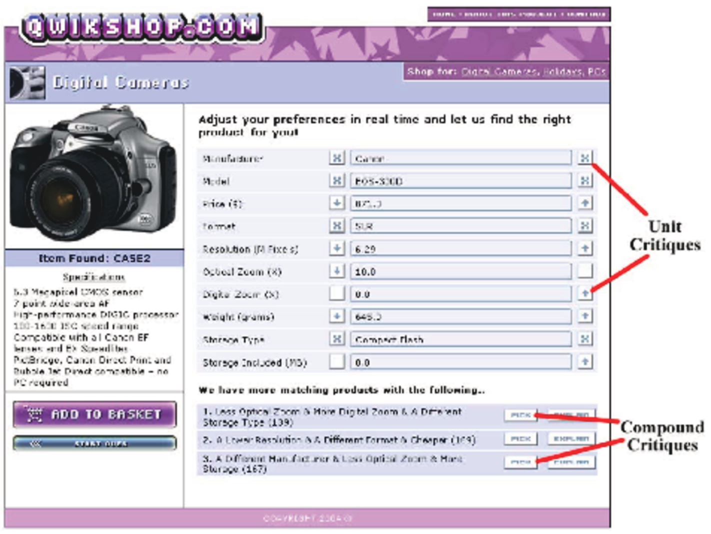
Phone: 312-489-1554	2-bedrooms	$600	60622 (West Town Bucktown)

This apartment is OK, but make it...
bigger
cheaper
nicer
safer
This neighborhood could be more...
convenient
conservative
dynamic

Figure 5.1.: Burke et al.'s FindMe approach (taken from [BHY96])

back of Burke's solution is that the user is forced to choose from system-proposed tweaks and is not able to provide self-initiated critiques. This can result in a non-coherent refinement process and hence may lead to incorrect preference expressions and poor motivation. This in turn would result in the elicitation of inconsistent requirements models. We also like to note, that though providing critiques in natural language rather than in terms of low-level features may make those critiques more comprehensible for the user, but also makes them subject to different interpretations. This is particularly true for critiques such as ”sportier” or ”nicer” (cf. Figure 5.1). Though they have a well-defined meaning for the system, they may have a different meaning for the user. Nonetheless, tweaking of intermediate recommendations stimulates the user to express preferences and requirements (at least as long as they meet the user's intention). It also educates the user about available items and inherent tradeoffs between their characteristics and guides the user in refining his requirements. However, as already noted, those effects are limited, since tweaks are predefined and hence might not fit to the actual item data and user requirements. One of Burke's FindMe implementations, CAR NAVIGATOR, provides additional guidance by tradeoff explanation features. If the user chooses an impossible feature combination that violates feature constraints, the system explains, that he needs to compromise here. However, again, the knowledge about required tradeoffs has to be manually provided by a knowledge engineer. Moreover, the guidance provided by the system is limited in that the refinement process itself is not directed and not focused on promising directions. This is partially attributed to the fact that uncertainty about the user's requirements is not modeled and hence an important ingredient for enabling a directed refinement process is missing. Finally, a major drawback of the suggested approach is that, to the best of our knowledge, it has never been formally evaluated, hence it remains open, whether the elicited requirements and preferences comply with the user's actual requirements, whether the suggested approach contributes to the user's awareness of his requirements and whether the recommended items are relevant for the user. It is also not known, if the system successfully assists consumers in making informed and consistent selections.

Reilly et al. [RMMS04, Rei05, RSMM05, RZM$^+$07] proposed a case-based conversational recommender. A user's requirements are modeled as a vector of constraints over a set of attributes, that is predefined for a certain domain. Preferences are not considered. Item retrieval is similarity-based. To assist the user in the selection process, Reilly et al. employ a critiquing approach. After posing an initial query, the user is provided with a textual description of the item recommendation that best fits to his requirements, i.e. with the item (case) that is most similar to his query. In a subsequent step, he may criticize this item (cf. Figure 5.2). The provided critique works as an item filter in the next retrieval phase, where the item that is most similar to the previous recommendation and compatible with the critique is proposed. Critiques are either unit critiques, which are user-initiated, or compound critiques, that are system-proposed and thus can break the coherence of the refinement process. In contrast to the solution of Burke et al., compound critiques are dynamically generated at runtime considering available items and their properties. To emphasize this characteristic, Reilly called his approach "dynamic critiquing". Critique generation works as follows. In a first step, item vectors, i.e. cases, are transformed into vectors of attribute-related critiques that characterize the considered items relative to the current recommendation. For instance, if an item is cheaper, but of less quality than the recommended item, the resulting critique vector would be $\langle \uparrow, \downarrow \rangle$, indicating that the first attribute's value was improved and the latter's declined. In a second step, recurring critique patterns are identified utilizing the Apriori-algorithm [AIS93] for frequent itemset and association rule mining. The resulting set of frequent critique patterns is typically large. For that reason, Reilly proposed to present just 5 of those critiques to the user. This is done by providing a textual description of the critiquing pattern (cf. Figure 5.2), which, according to [RZM$^+$07], was easy to understand for the majority of the test users. Critique selection is guided by the support of the mined critique, indicating the proportion of items to which the pattern applies. Though dynamic critiquing educates the user about available products and their characteristics and encourages the user to make tradeoffs, it is limited in that the suggested compound critiques do not necessarily meet the user's intention and thus do not necessarily provide relevant information, that is required for making an informed selection. This is confirmed by Reilly's evaluation results [RZM$^+$07] and attributed to



Adjust your preferences in real time and let us find the right product for you!

Manufacturer	Canon
Model	EOS-300D
Price ($)	871.0
Format	SLR
Resolution (M Pixels)	6.29
Optical Zoom (X)	10.0
Digital Zoom (X)	0.0
Weight (grams)	645.0
Storage Type	Compact Flash
Storage Included (MB)	0.0

Item Found: CASE2

Specifications

6.3 Megapixel CMOS sensor
7 point wide-area AF
High-performance DIGIC processor
100-1600 ISO speed range
Compatible with all Canon EF lenses and EX Speedlites
PictBridge, Canon Direct Print and Bubble Jet Direct compatible – no PC required

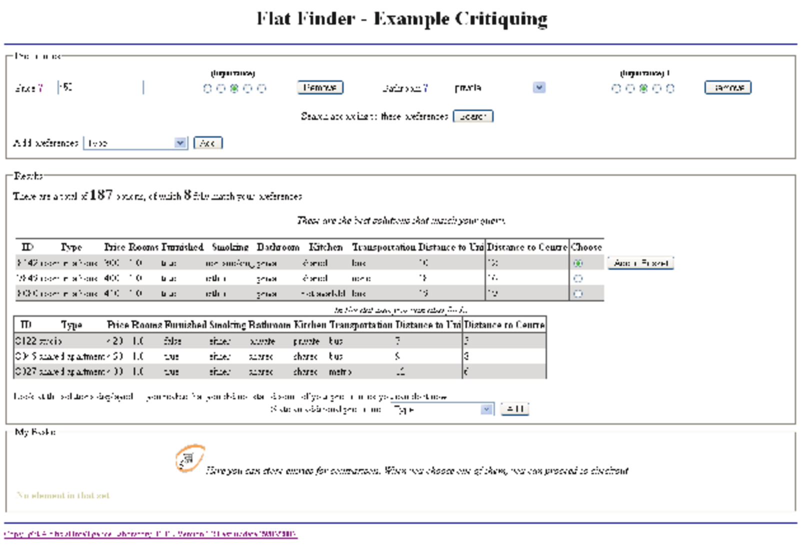
We have more matching products with the following..

1. Less Optical Zoom & More Digital Zoom & A Different Storage Type (139)
2. A Lower Resolution & A Different Format & Cheaper (169)
3. A Different Manufacturer & Less Optical Zoom & More Storage (167)

Figure 5.2.: Reilly's dynamic critiquing approach (taken from [Rei05])

the fact, that the user's requirements and preferences are not considered when selecting critiques. As a consequence, the user can be discouraged or even worse can be induced to state preferences he does not have, which finally results in an inconsistent consumer model. Unfortunately, the authors did not evaluate those aspects. The mentioned problems are mitigated to some degree, since the system allows the user to provide self-initiated unit critiques. Moreover, Reilly et al. measure uncertainty about or confidence in the user's requirements respectively. This is done by maintaining a history of the critiques that have been previously applied during a session. The compatibility of an item's attribute values with the critiques that have been already stated is taken as a measure of confidence in the fact, that the user requires the considered attribute to have this value. For instance, if most of the critiques do not conflict with a product price of 200$, then the system's confidence in the fact, that the user accepts a product for 200$ is high. Item recommendations are ranked based on the system's confidence in the recommendation and the recommended item's similarity to the critiqued product. In addition, when presenting the top-ranked recommendation, the attribute related confidence as well as the overall confidence in this recommendation are communicated to the user. This is to allow users to identify gaps in the system's knowledge and thus enable them to direct and focus the refinement process by providing critiques related to those attribute values, the system is not sure about. However, on the other hand, by focusing recommendation on those items that the system is confident in, the refinement process is not directed in a way that reduces model uncertainty. Moreover, confidence in the user's requirements is not considered when selecting promising compound critiques. Nonetheless, according to the authors, leveraging model confidence to improve the refinement process results in a reduction of the average session length by up to 46% (simulative evaluation [RSMM05]), which emphasizes the importance of effective elicitation techniques that aim at reducing model uncertainty. However in this context, it was not analyzed and thus remains open, whether the suggested approach also contributes to the user's awareness of his requirements and whether the final item selection was consistent with the user's requirements.

The previous approaches have in common, that they enforce users to express their requirements and preferences over a fixed set of attributes. However, as already argued, users actually might not have preferences on all attributes. Hence, by obligating to provide non-existing preferences, those approaches might elicit incorrect preferences, which in turn might lead to inconsistent consumer models. The solution of **Viappiani et al.** [VFP06, VPF07, VPF08] mitigates that problem. It allows the user to state initial preferences on fundamental attributes and then motivates him to freely state further, hidden, preferences by showing him challenging example items. Since the user is not forced to state preferences, this method stimulates correct preference expressions. As Viappiani et al. argue, stimulating item suggestions should be reasonable choices under the current requirements model and should provide a high probability of being optimal after an additional preference has been added. The authors implemented a conversational recommender system based on this policy, which they called lookahead principle. For the implementation, Pareto-optimality was chosen as the optimality concept. Consumer requirements and preferences are modeled as a set of attributes each associated with a parameterized preference function. Uncertainty about the existence of certain preferences is encoded by probability distributions. Uncertainty about the kind of preferences is encoded via probability distributions over the actual values of the preference functions' parameters. Once the user has provided some initial preferences, he is given several relevant item recommendations that differ in a possible preference. Those recommendations are not already optimal for the stated preferences, but have the highest likelihood to become Pareto-optimal when adding a new preference and thus stimulate preference expression. The latter is facilitated by presenting the recommended items in a table containing a column for each attribute (cf. Figure 5.3). This allows for an easy comparison of the single items' attribute values. The set of shown example items is chosen in a way that maximizes the probability that at least one of the suggestions in this set will become optimal due to the user having specified a yet unstated preference. Since explicit optimization is combinatorial complex, an approximate solution is provided. Users may state additional preferences as long as they think it gets them to a better choice. After that, the probability distributions are updated based on the user's reaction to the shown examples. Finally, a new set of recommendations based on the updated requirements model is presented. The process finishes successfully when the user has found the desired item among the recommendations. As shown in [VPF08], by choosing item recommendations following the lookahead principle, the introduced approach to requirements and preference elicitation stimulates correct preference expressions, contributes to the user's awareness of his requirements and motivates informed selections. However, it remains open, whether the elicited model is consistent with the user's actual requirements and whether the suggested approach facilitates consistent selections. In the course of the elicitation process, model uncertainty is effectively reduced in the sense that preferences about yet unconsidered attributes are added. However, this does not hold for uncertainty associated with already stated preferences that might be


Flat Finder - Example Critiquing

There are a total of 187 [illegible], of which 8 [illegible] match your [illegible]

ID	Type	Price	Rooms	Furnished	Smoking	Bathroom	Kitchen	Transportation	Distance to Uni	Distance to Centre	Choose
[illegible]	[illegible]	900	1.0	[illegible]	[illegible]	[illegible]	[illegible]	bus	[illegible]	[illegible]	●
[illegible]	[illegible]	400	1.0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	○
[illegible]	[illegible]	4[illegible]	1.0	[illegible]	[illegible]	[illegible]	[illegible]	bus	[illegible]	[illegible]	○

ID	Type	Price	Rooms	Furnished	Smoking	Bathroom	Kitchen	Transportation	Distance to Uni	Distance to Centre
0122	studio	[illegible]	1.0	false	[illegible]	[illegible]	private	bus	7	[illegible]
[illegible]	shared apartment	[illegible]	1.0	true	[illegible]	[illegible]	shared	bus	[illegible]	3
0027	shared apartment	[illegible]	1.0	true	[illegible]	[illegible]	shared	metro	[illegible]	[illegible]

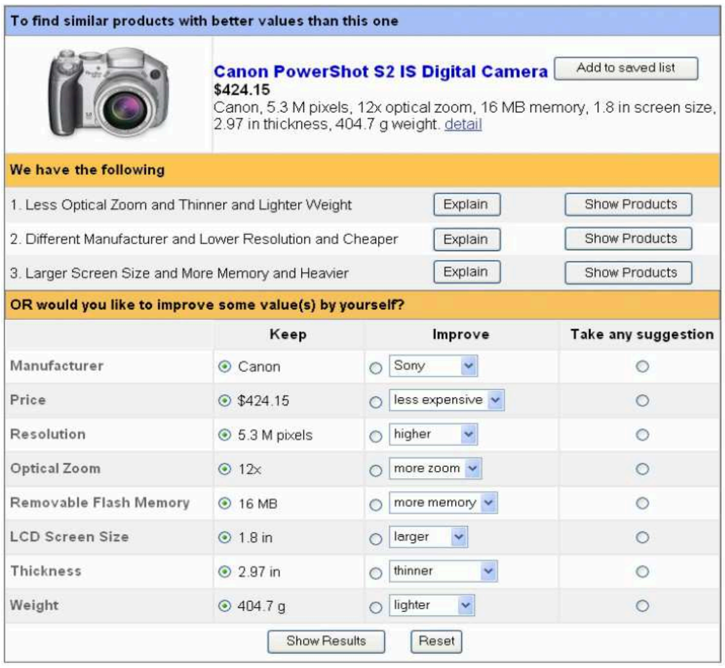
My Basket

No element in that set

Figure 5.3.: Viappiani's example application Flat Finder (taken from [VPF08])

caused by inconsistent user input. In addition, the system just elicits preferences related to individual attributes, but does not account for tradeoffs that are necessary when dealing with conflicting requirements. This may lead to non-optimal recommendations, since there is a preference order over Pareto-optimal items, which is not considered. The authors argue that preferences related to the importance of attributes can be elicited in a subsequent step. However, such a 2-step optimization process would be suboptimal in terms of its efficiency. In addition, the proposed refinement process is likely to be not coherent, since, although the user may freely state preferences, the shown options motivate preference expressions related to a system-selected attribute that does not necessarily correspond to the attribute the user is currently interested in. However, this aspect was not evaluated by the authors.

Chen et al. [CP07b, CP07c, CP07a, Che08] also proposed a constraint-based example critiquing recommender. In contrast to Viappiani et al., they focus on assisting the user in making tradeoffs between conflicting requirements. In their approach, items and their characteristics are modeled as a vector of values for a fixed set of attributes. A consumer's requirements are encoded as parameterized preference functions over those attributes' values that indicate to what degree the user likes a certain attribute value. Weights are used to cover the relative importance of attributes. This allows to explicitly resolve conflicting requirements by defining a user' s overall preference for an item as the weighted sum of the preference values related to its single attributes. Based on the introduced requirements model, Chen et al. proposed an example critiquing system. It allows a user to apply self-initiated unit critiques or system-suggested, dynamically generated compound critiques to intermediate recommendations. Similar to Reilly et al., Chen focuses on compound critiques as a means to support the user in performing tradeoffs. However, as indicated, those critiques are system-suggested and cannot be formulated individually by the user. This is particularly annoying in cases where the proposed critiques do not correspond well to the user-intended tradeoff criteria. Hence, offering only system-suggested critiques can discourage the user and/or can lead to incorrect preference expressions. As a solution to this problem, Chen et al. proposed a hybrid system that supports both, user-initiated as well as system-suggested compound critiques (Figure 5.4). However, Chen's system does not actually integrate both styles of critiquing. Instead, the user can either choose to provide a self-initiated critique or can select a system-suggested compound critique during a single recommendation phase. As for self-initiated critiques, the user may change condi-



To find similar products with better values than this one

Canon PowerShot S2 IS Digital Camera Add to saved list
$424.15
Canon, 5.3 M pixels, 12x optical zoom, 16 MB memory, 1.8 in screen size, 2.97 in thickness, 404.7 g weight. detail

We have the following

1. Less Optical Zoom and Thinner and Lighter Weight	Explain	Show Products
2. Different Manufacturer and Lower Resolution and Cheaper	Explain	Show Products
3. Larger Screen Size and More Memory and Heavier	Explain	Show Products

OR would you like to improve some value(s) by yourself?

	Keep	Improve	Take any suggestion
Manufacturer	[✓ Canon]	Sony	○
Price	[✓ $424.15]	less expensive	○
Resolution	[✓ 5.3 M pixels]	higher	○
Optical Zoom	[✓ 12x]	more zoom	○
Removable Flash Memory	[✓ 16 MB]	more memory	○
LCD Screen Size	[✓ 1.8 in]	larger	○
Thickness	[✓ 2.97 in]	thinner	○
Weight	[✓ 404.7 g]	lighter	○

Show Results Reset

Figure 5.4.: Chen et al.'s hybrid recommender system (taken from [CP07b])

tions on single attributes, change attribute weights or add additional preferences. Moreover, he may perform tradeoffs by indicating attributes that should be optimized and those that may be compromised to achieve this (see Figure 5.4). As a result of those critiques, the requirements model is updated accordingly. The update strategy triggered by user-initiated critiques, is heuristic and directly adjusts model parameters to reflect the user's feedback. For instance, in case of a tradeoff, the weights of the optimized attributes are increased and those of the compromised are decreased. After user-initiated critiquing, the system starts a new recommendation phase, where a number of matching items is presented to the user. For that purpose, the set of available items is first reduced by applying an elimination-by-aspect procedure [JWP93]. This eliminates those items that do not satisfy certain attribute-related thresholds. To inform the user about possible tradeoffs, the system then determines among those items those that maximally satisfy subsets of the stated preferences and presents them to the user. The items are ranked by their overall preference value. However, since the user is not guided in composing compound critiques, the described process can lead to an empty result set, if the user's requirements are too restrictive.

In addition to user-initiated critiques, the system can suggest compound critiques to the user. Those critiques represent tradeoffs that are required in light of the available items and that are likely to be acceptable for the user. For example, the system might indicate that there are a number of items that are cheaper, but of worse quality than the current top item and that there are also items that are more expensive, but of higher quality (see also Figure 5.4 for examples from the domain of digital cameras). By being presented those patterns, the user learns that he has to compromise between price and quality. By choosing one of the suggested critiques, the user actually makes a compromise by weighting one attribute over the other. As shown in [Che08], this contributes to the user's awareness of his requirements. Like Reilly et al., Chen et al. apply the Apriori-algorithm [AIS93] for frequent itemset mining to find potential critiquing patterns related to the top item. To reduce computational effort, the set of input items is reduced to 50 by applying heuristic selection strategies. From the set of frequent critique patterns produced by the Apriori-algorithm, a subset is chosen which fulfills some heuristically defined properties. For instance, all patterns that refer to more than 3 attributes are dropped. Finally, 4 critiques are selected and presented to the user. In contrast to the approach of Reilly et al., critique selection is not just guided by the characteristics of the available items, but also by the system's knowledge about the user's preferences and requirements. More specifically, critique patterns are chosen in a way that maximizes the tradeoff utility of the selected patterns as well as their diversity in terms of shared attributes and the items they are applicable to. Thereby, tradeoff utility is a heuristically defined measure, indicating the degree of conformance between the critique pattern and the consumer's requirements. Finally, the critique patterns presented to the user are labeled. The label explains which improvements relative to the top-ranked item are achieved by applying this critique, but also which compromises have to be accepted. Hence, the label is a means to educate and support the user. The selection of a certain critique pattern by the user triggers a model update and starts a new recommendation phase. As Chen's evaluation results indicate [CP07b], the consideration of a user's requirements and preferences, when proposing promising critique patterns and the usage of explaining critique labels improved decision accuracy, i.e. led to an increased proportion of informed selections. It also reduced decision making effort and improved the critique prediction accuracy (indicating how well the suggested critiques suit to the user's critiquing intention) compared to the one achieved with the approach of Reilly et al.

However, though the solution of Chen et al. exhibits some major advantages, it requires improvements with respect to a number of aspects. First of all, the proposed procedure for critique selection excludes many item opportunities that might be interesting for the user. Moreover, it shows just a very small subset of interesting critique patterns and thus is likely to miss those that meet the user's critiquing intention. As Chen's evaluation results indicate [CP07b], the user's critiquing intention was met in just about 44% of the cases. Hence, though the suggested critiques motivate active participation and provide guidance for making tradeoffs, the refinement process is not coherent (in case system-suggested critiques are chosen). This observation is also supported by the evaluation results of Chen [CP07b].

Though proposed critique patterns were chosen more often than in a system implementing Reilly's algorithm, still half of the critiquing interactions were self-initiated. This is particularly regrettable, since, as noted earlier, the proposed system does not assist the user in providing self-initiated critiques. This results in a low decision accuracy of about 33% for self-initiated critiques, whereas the decision accuracy when exclusively choosing system-suggested critiques was at about 67%. Unfortunately, it was not evaluated whether these decisions were consistent with the user's actual requirements. Another drawback of Chen's approach is, that uncertainty about the consumer's requirements is not taken into account. As a result, the model refinement process is not driven by the desire to reduce uncertainty about the user's requirements and preferences and hence lacks an important feature. Finally, contrary to Viappiani et al., the authors assume that a user has preferences on all the predefined item attributes. Hence, by implicitly assuming preferences over attributes for which no preferences exist or by enforcing preference expressions over those attributes, the system elicits incorrect preferences. This leads to inconsistent requirements models. Unfortunately, this aspect was not evaluated by the authors.

5.2.2. Utility Elicitation

Expected utility theory [Ber54, JvN53] assumes that a decision maker's preferences with respect to uncertain outcomes can be described by a mathematical relation, called utility. Thereby, utility accounts not only for the payout size of an outcome, but also for the risk attitude of the decision maker and the fact that a certain payout might have a different value for different people. According to the theory, when facing risky choices, rational decision makers choose the decision strategy that maximizes the expected utility of the outcome. However, in order to actually make decisions in accordance with this principle, knowledge about the likelihood and the user-specific utilities of possible outcomes is required. While the process of acquiring the probabilities is well-known [CKP00], the elicitation of a decision maker's utility function [1] is problematic. This is for several reasons. Traditional approaches to utility elicitation (see [Cle91] for an overview) require individuals to answer a large number of cognitively hard questions and hence are error-prone and time-consuming. Due to the complexity of many real world decision problems, the application of those methodologies is often even not feasible at all. In those cases, the amount of utility information that can be acquired is limited and hence, the decisions based on those information are not optimal. In general, the more we know about a user's utility function, the higher is the decision accuracy that will be achieved. However, different aspects of a utility function might have a different impact on the decision quality. Hence, elicitation questions should be chosen carefully. In the following, we analyze two prominent, non-traditional approaches to utility elicitation that consider this issue and investigate how they fit to our requirements. Though expected utility theory makes a number of assumptions that are hardly given in many real world decision situations and though the process of utility elicitation is not feasible for our purposes, closer inspection of the mentioned approaches is

[1] A utility function maps outcomes to real numbers, the utilities.

advisable. This is due to the fact, that they achieve a significant reduction of the number of required elicitation questions by considering uncertainty about the elicited utility function.

The approach of **Chajewska et al.** [CKP00] aims at eliciting the utility function of a decision maker by posing as few as possible elicitation questions, while at the same time ensuring a good decision quality. The proposed solution presumes that the standard gamble approach to utility elicitation [JvN53] is used. This means, the used elicitation questions are of the type "Given the choice between outcome o for sure and a lottery which gives the best outcome with probability s and the worst with probability $1 - s$, which will you choose?". Uncertainty about the utility function to be elicited is encoded by modeling outcome utility as a random variable. The proposed elicitation algorithm works as follows. It first computes the optimal decision based on the current (uncertain) utility model, then asks the elicitation question with the highest value of information and finally updates the utility model according to the answer. The process stops, if the expected utility loss, that is caused by taking a sub-optimal instead of the optimal decision (due to incomplete utility information), falls below a predefined threshold. Since computing the expected utility loss exactly is impractical, it is approximated by using Monte Carlo methods. In this context, the optimal decision based on a given uncertain utility function, is the one that has the highest expected expected (mean) utility over all possible outcomes. The value of information is defined as the difference between the posterior expected utility of the optimal solution and the expected utility of the currently optimal solution considering the likelihood of both possible answers to the elicitation questions. The algorithm determines which outcome o and which probability s to choose in the next elicitation question. The solution achieves a significant reduction of the number of elicitation questions that are required. Having a threshold of 0.05 for the expected utility loss and 108 possible outcomes, the average number of questions was between 2.3 and 3.9 . The proposed solution is also applicable to correlated decision outcomes. However, it is myopic in the sense of not accounting for the value of future questions and hence may underestimate the value of information gained by asking a certain elicitation question.

Boutilier [Bou02] extends the approach of Chajewska et al. in that he considers not only the value of the current elicitation question, but also the value of future questions when determining the next elicitation question. This is achieved by modeling the elicitation process itself as a sequential decision problem, in which a sequence of elicitation questions has to be chosen in a way that balances elicitation effort and decision accuracy. Boutilier formulates this decision problem as a partially-observable Markov decision process (POMDP). However, due to the continuous state and action spaces of the resulting POMDPs, standard techniques cannot be applied to solve it. As a solution, Boutilier provides an algorithm that computes an approximately optimal solution. The runtime of this procedure is high. However, most of the calculation can be done offline.

The introduced approaches explicitly model uncertainty about a user's preferences (and his risk attitude) and impressively demonstrate how this knowledge can be leveraged to effectively direct and focus the preference refinement process. The resulting elicitation process is adaptive to the elicited knowledge about the user's preferences and considers available decision alternatives. The applied techniques are generic in the sense, that they can be easily applied to several domains or application scenarios, but are restricted to the standard gamble approach to utility elicitation. Since elicitation questions are directly posed to the user, the method stimulates preference expression. By the selection of appropriate elicitation questions the system guides the user and directs the elicitation process. However, the proposed techniques exhibit a disadvantage they share with many other approaches to utility elicitation. They leave no control to the user and impose a purely reactive position on him. More precisely, the sequence of elicitation questions chosen, is purely system-defined. Since the rationale behind the selection of the elicitation questions is not explained to the user, the posed questions must appear arbitrarily chosen. Moreover, the elicitation questions which are posed are rather abstract and their relation to the actual decision alternatives and to the user's preferences remains unclear. This is likely to discourage the user, makes him unable to construct a mental model of his preferences and unable to develop a clear picture of the available alternatives and their characteristics. Hence, though the system might be able to present a decision strategy that is optimal with respect to the available alternatives and the user's elicited requirements, the user itself is unable to make a selection that is informed and that is consistent with his service requirements. Moreover, since the proposed elicitation procedure is not coherent, it makes it difficult for the user to adjust to the elicitation process and thus to provide correct preferences. As a consequence, it is likely that the elicited preferences are incorrect to some degree.

5.2.3. Semantic Web Service Selection

In this section, we will investigate how existing approaches to Semantic Web Service selection fit to our requirements which have been discussed in Section 5.1. In our analysis, we will discuss solutions that solely facilitate the creation of formal semantic request descriptions by providing tool-support and those that provide advanced assistance in specifying service requirements.

Assisted Request Creation

The need for tool support [Agr06] that enables end-users to actually use Semantic Web Service technologies has long been recognized by the research community. To this end, a number of standalone tools, integrated development environments and service engineering frameworks aiming at the support of developers in the full Web Service development lifecycle, have been launched. Among the most prominent representatives are the Web Service Modeling Toolkit (WSMT) for WSMO-based service descriptions [KMSF09], the INFRAWEBS Integrated Framework (IIF) [AMLM07, LPN$^+$07] and the OWL-S editors by Elenius et al. [EDM$^+$05] and Scicluna et al. [SAM04]. However, those efforts are mainly targeted at service providers and application developers and only marginally address the end-users, i.e. service consumers, who require assistance in expressing their service needs and support in the subsequent process of service selection. Typically, the approaches [AMLM07, soa08] assume that application providers create generic request templates, that cover frequent service needs in a certain application domain, at design time. Later on, those templates are instantiated by end-users to specify their needs. For that purpose, the user has to provide the missing data in order to complete the template. [soa08] propose a menu-based interface to support this process. However, typically this task is left to the developers of end-user applications. The focus of the efforts is on assisting developers in the process of template creation and management. To this end, a palette of tools, such as goal editors, validators, visualizers and browsers as well as tools for ontology visualization and management have been developed [KMSF09, AMLM07, EDM+05, SAM04]. Though, a number of graphical goal editors (INFRAWEBS Designer [Agr06, LPN+07], OWL-S editor [SAM04]) have been proposed to facilitate that process, still substantiated knowledge about the underlying description language is required. Typically, such as in the case of WSMO-based descriptions, this also premises a comprehensive understanding of mathematical logic. To ease the creation process and to foster reuse of existing descriptions, the INFRAWEBS Integrated Framework [Agr06, AMLM07, LPN+07] developed a case-based memory that allows template designers to find semantic service descriptions (or parts of it) and ontologies that are related to a certain template under construction to reuse them in a copy and paste manner. However, the developer is neither supported in the process of identifying typical service needs in a certain application domain nor supported in the task of adequately modeling them. Hence, it is very unlikely, that later on an end-user's abstract service needs are appropriately transfered into a formal request, which is then used to discover suitable services. This similarly applies to approaches that use Natural Language Processing in order to transfer service desires given in natural language into formal requests [soa08, BVMC05] and those that rely on wizards for request creation [soa08]. The more surprising it is, that uncertainty about a user's service requirements is not considered in any of the mentioned approaches. Moreover, none of the solutions accounts for the constructive nature of consumer requirements by supporting incremental requirements specification and none of the approaches considers actually available service offers and their characteristics during that process. Requests are created in advance without this knowledge, before service selection takes place.

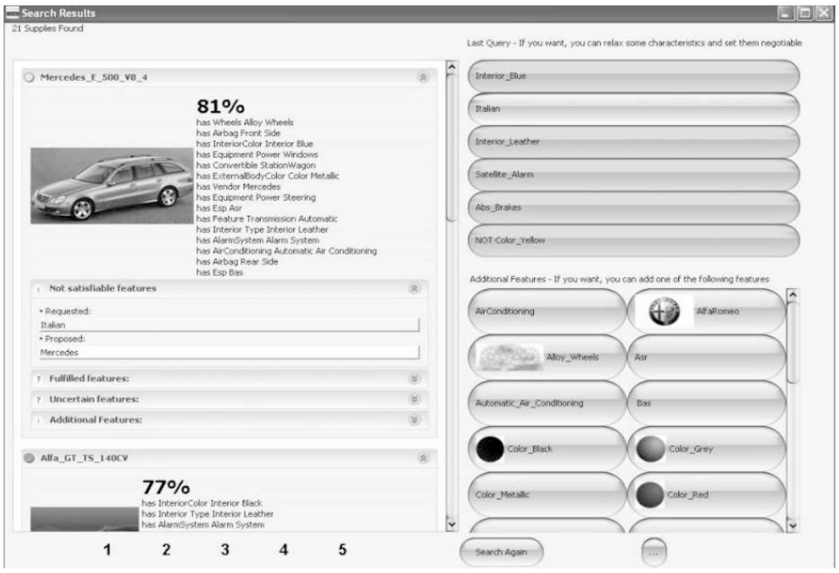
Assisted and Personalized Service Selection

In this paragraph, we will look at user-centered approaches to requirements elicitation and service selection. They differ from those that have been introduced in the previous paragraph in that they provide advanced assistance in the specification of service requirements.

Colucci et al. [CNS+06, CNS+04] propose a visual interface for assisted creation and refinement of OWL-based service requests, which was also ported to mobile devices [RNSS08]. At the beginning of the query formulation process, the user has to create an initial request. The system supports him in that task by providing ontology browsing facilities and by graphically visualizing the request. Initially, the user sees the most generic classes of the domain ontology. As he choses a particular concept, the system displays available subclasses as well as all roles having the selected class as a domain. If possible, classes and roles are visualized by an illustrative icon. Elements from the ontology may be added to (or removed from) the request, which is visualized in a separate panel of the application. This query panel shows the part of the request that is currently focused. A history bar allows for bottom-up-navigation. All requirements that have been specified in the initial service request are assumed to be strict, i.e. not negotiable. Preferences related to service characteristics cannot be expressed. Once the user decided to pose the created request, a ranked list of matching service offers is displayed. Matchmaking applies subsumption and satisfiability inference services provided by description logic reasoners to identify offers that are compatible with the given request. Depending on their compatibility properties, offers are classified into (worst to best) partial, potential, full and exact matches (see [CNS$^+$06, CNS$^+$04] for more information). Though exact and full matches indicate suitable offers, potential and partial matches might also be appropriate choices and are encountered more frequently. Hence, the request refinement process proposed by the authors is driven by the desire to upgrade those matches by eliciting additional consumer requirements. In particular, a user might relax certain requirements to make a partial match to a potential match and might specify additional requirements to reach a full match from a potential match. Colucci et al. propose two non-standard inference services, namely concept contraction and concept abduction [CNS$^+$04], to assist the user in that process. Concept abduction allows to suggest additional features that are contained in the matching offers and might be added to the user's request. Concept contraction allows to identify features that might be marked as negotiable by the user to achieve a potential match. During the refinement process, those information are displayed in addition to a ranked list of matching offers. Moreover, explanations on the matching results as well as a verbalization of the matching offers' OWL descriptions are provided (see Figure 5.5). Equipped with those information as well as with a visualization of his query (as a set of features), the user is free to refine his request by changing the specified requirements. After that, he may initiate another matchmaking phase. The refinement process stops, if the user finds a suitable service offer among the presented results.

By showing suitable service offers and their characteristics and by suggesting promising modifications to the user's requirements, the proposed approach educates the consumer about available service alternatives and stimulates requirements expression. Unfortunately, support for trading off conflicting service requirements is not provided. Since the user freely makes changes to his requirements and is not enforced to do so, the resulting requirements refinement process is coherent. However, uncertainty about a consumer's service requirements is not considered and the process of requirements elicitation is not driven by the goal to reduce model uncertainty. Though the authors indicated, that they performed preliminary tests of the system, we are not aware of a publication that presents their results. Hence, it remains open, whether the elicited requirements and preferences comply with the user's actual requirements and whether the suggested approach contributes to the user's awareness of his requirements. It is also not known, whether the system successfully assists consumers in making informed and consistent selections.

With MobiXpl, **Noppens et al.** [NLS07, NLL$^+$06, WLN$^+$04] propose a mobile user interface for personalized semantic service discovery. In their approach, services are classified



21 Supplies Found

Mercedes_E_500_V8_4

81%
has Wheels Alloy Wheels
has Airbag Front Side
has InteriorColor Interior Blue
has Equipment Power Windows
has Convertible StationWagon
has ExternalBodyColor Color Metallic
has Vendor Mercedes
has Equipment Power Steering
has Esp Asr
has Feature Transmission Automatic
has Interior Type Interior Leather
has AlarmSystem Alarm System
has AirConditioning Automatic Air Conditioning
has Airbag Rear Side
has Esp Bas

Not satisfiable features
- Requested: Italian
- Proposed: Mercedes

Fulfilled features:
Uncertain features:
Additional Features:

Alfa_GT_TS_140CV

77%
has InteriorColor Interior Black
has Interior Type Interior Leather
has AlarmSystem Alarm System

Last Query - If you want, you can relax some characteristics and set them negotiable

Additional Features - If you want, you can add one of the following features

Figure 5.5.: Colucci et al.'s interface for assisted query refinement (taken from [CNS+06])

with respect to several categories, e.g. services offering music streams might be classified with respect to their program format and their location. Each category is characterized by an OWL-based is-a-hierarchy of aspects related to this category. For instance, the program format might be sports, music and so on. Music might be further classified into Classical, Rock and so on. A service request is a set of preferences over service aspects related to the given categories. Preferences are expressed in terms of orderings between service aspects within a category, which results in partially ordered aspect sets. In particular, the user might express a preference of one aspect over another, indifference between aspects and dislike of an aspect. To assist the user in specifying those preferences, the system provides zoomable, graph-based visualizations of the category ontologies that can be browsed by the user. Interesting aspects can then be selected and organized in a preference graph (see Figure 5.6). To facilitate the aspect selection process, the system performs a preselection of aspects and provides ontology views for typical usage patterns. Moreover, preferences are checked before they will be established to avoid redundant and/or inconsistent preference expressions. Preferences are directly handled for service retrieval. In this context, Pareto accumulation and preference prioritization are applied to compare combinations of preferences related to different categories and to weight categories according to their importance. Preference prioritization is done automatically by the system and cannot be performed by the user. For instance, it is assumed that a category with more detailedly described preferences is more important to the user than one with less detailedly specified preferences. To avoid empty result sets caused by too restrictive preferences, the authors propose ontology-based preference relaxation techniques, where preference aspects are gradually relaxed to super aspects until they can be fulfilled.

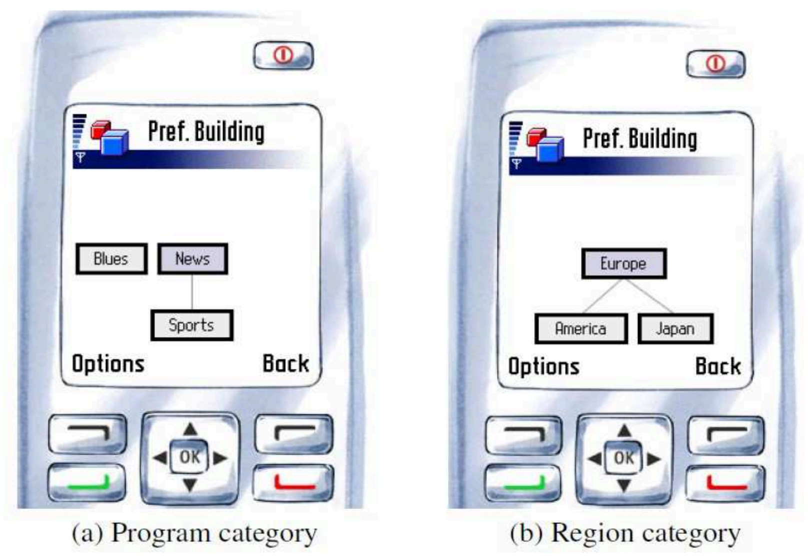


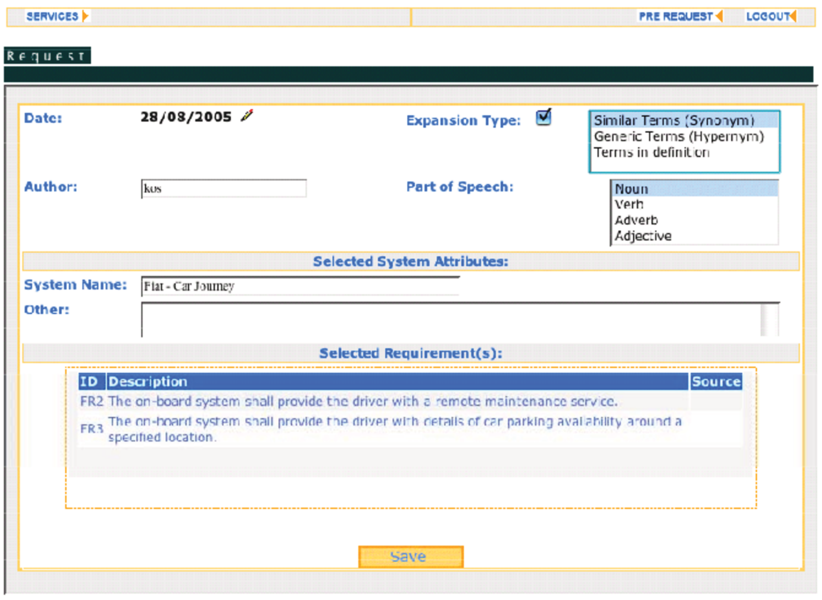
Figure 5.6.: Noppens et al.'s interface for personalized semantic service discovery (taken from [NLL+06])

The proposed approach uses qualitative preferences, that allow for intuitive preference formulation. Unfortunately, it relies on simple category-based service descriptions and does not allow to richly describe functional service requirements. Hence, its application is restricted to scenarios, where services of a known type need to be discovered, e.g. services that provide music streams as in this case. The presented interface motivates and guides self-initiated preference specification. However, the assistance is limited in that available service alternatives are not considered. Moreover, the proposed solution implements a single shot approach, where preferences cannot be refined after viewing matching results. This is in contrast to the constructive nature of preferences and thus avoids the establishment of well-constructed preferences. Finally, the suggested approach has been implemented [NLL+06], but to the best of our knowledge, has never been formally evaluated. Hence, it remains open, whether the suggested solution contributes to the user's awareness of his requirements and whether it maintains a consistent model of the user's service requirements. In addition, it is not known whether the finally selected service is consistent with the user's requirements and whether it is chosen after having thoroughly reviewed relevant service alternatives.

Users typically do not know the characteristics of available services and thus might formulate service requests that do not allow to discover all services that potentially suit to their needs. To address this issue, **Balke et al.** [BW03b, BW03a, BW04] propose a cooperative approach to service selection and discovery. The main idea is, that instead of taking a user's request as it is, the system behaves cooperatively by automatically rewriting and expanding this request to retrieve additional services that might potentially fit to the user's needs. The authors suggest several techniques to achieve this. Firstly, they expand and rewrite a given service request based on knowledge about the anticipated usage situation (usage pattern). This accounts for the fact, that a certain service type might be used for different purposes and in different contexts, each of which is associated with typical preferences and different categorizations used to classify services according to their characteristics. For instance, restaurants might be classified according to their cuisine and their location. However, in some usage situations, cuisine categorized by nationality is used, while in others a categorization by taste might be preferred. In this context, the authors also propose to rely on different conceptual views to the ontologies that serve as service classifications. Those views are clippings from the full ontologies and represent user-specific views to the service categorizations. Finally, Balke et al. suggest to expand requests with soft constraints taken from a user's profile or from general domain knowledge. However, applying too many constraints might result in an empty result set. To deal with that issue, the authors propose a strategy for the incremental relaxation of requests, i.e. a generalization of the required service class(es) along the lines of the associated conceptual view(s). The developed procedure avoids deep relaxation of single constraints and accounts for the relative importance of different conceptual views as well as for their granularity. Starting with the minimal possible relaxation, the algorithm incrementally creates and processes requests resulting from increasing relaxations, until the user is satisfied with the presented service alternatives. The actual matchmaking is carried out by transferring requests into preference-based database queries and applying cooperative database technology [BW03a, Min98]. To enable this, the DAML-S based service descriptions are stored in a classic relational database. An enhanced UDDI repository, which also holds usage patterns and OWL/DAML+OIL based service ontologies, performs the cooperative search.

Though the approach models consumer preferences and accounts for the fact that initial consumer requests might be incomplete, it does not actively involve the user in the refinement process, but restricts his contribution to triggering additional relaxations and stopping the process, if desired. This prevents the user from actively constructing his requirements and is likely to produce query relaxations that are not compliant with the user's actual requirements. However, since those aspects have not been evaluated, we will not know, whether the discussed approach contributes to the user's awareness of his requirements and whether it maintains a consistent model of his service requirements. Moreover, it remains open whether the finally selected service is consistent with the user's requirements and whether it is chosen after having thoroughly reviewed relevant service alternatives. The authors also did not provide any information about the user interface that was used to elicit service requirements and preferences and did not comment on how matching services are presented to the user.

Within the EU-project SeCSE, **Zachos et al.** [ZZMJ06, ZMHM08, sec06] developed a service discovery environment that supports software engineers in the identification of system requirements when building applications that integrate services from different providers (service-centric systems). Though the approach is not targeted to (direct) service consumers, it uses iterative service selection and requirements refinement as a means for both, building service-centric applications by discovering appropriate services and for refining system requirements. For that purpose, developers specify use cases and initial requirements in natural language via the UCaRE application. Those specifications are then used by the system to create service queries in structured natural language (cf. Figure 5.7). By



Date: 28/08/2005

Expansion Type: ☑ Similar Terms (Synonym) / Generic Terms (Hypernym) / Terms in definition

Author: kos

Part of Speech: Noun / Verb / Adverb / Adjective

Selected System Attributes:

System Name: Fiat - Car Journey

Other:

Selected Requirement(s):

ID	Description	Source
FR2	The on-board system shall provide the driver with a remote maintenance service.	
FR3	The on-board system shall provide the driver with details of car parking availability around a specified location.	

Figure 5.7.: Service query in structured natural language generated from the requirements given in the lower part of the screenshot (taken from [sec06])

using natural language and information extraction techniques, another module, called EDDiE, discovers descriptions of candidate services that are related to the requirements and ranks them according to their semantic similarity to the query. A service browser presents the retrieved service descriptions as well as the corresponding use case and requirements descriptions to the developer. Being inspired by the viewed service descriptions, the system engineer may refine and complete the system requirements and thus enables more accurate service retrieval. The approach offers several freely selectable retrieval techniques to assist the engineer in the refinement process. Analogical matching is used to identify new requirements by considering analogous services that are available in other domains. Random matching, i.e. retrieval of random services, is used to foster the generation of new ideas about the design of the system and thus might induce additional requirements. The authors also propose to match actor and use case names against predefined patterns in order to identify relevant services that facilitate the refinement and decomposition of already specified requirements. To refine requirements, they also suggest to use synonyms of actor names and terms in the requirements. Finally, they enforce system engineers to relax constraints given in non-functional requirements to explore more service offers in case of over-specified requirements. Though the evaluation results presented in [ZM08] seem to indicate that the system supports software engineers in the specification of requirements and provides them with relevant services[2], it is not directly applicable to our scenario. Since the system is based on natural language descriptions, its possibilities for effective matchmaking and for refinement support are limited. Moreover, due to the specific application scenario, the type of assistance that is provided by the system is different to that required in our scenario. Finally, it was not evaluated whether the suggested solution elicits correct software requirements and whether it maintains a consistent requirements model. In addition to that, it remains open whether the finally selected services are consistent with the identified requirements and whether they are chosen by the software engineers after thorough inspection of relevant service alternatives.

5.2.4. Summary and Open Research Issues

Table 5.1 summarizes the results of our analysis. The entries in the table indicate whether the corresponding approach partially (○) or completely (+) fulfills the considered requirement. We also indicated, if an approach does not fulfill a certain requirement (-), if a requirement is not applicable to an approach (/) or if it is not known whether the requirement is fulfilled or not (?). In some cases, an explanatory footnote has been added. In the remainder of this section, we will summarize our findings.

With example critiquing, recommender systems provide an intuitive and also effective mechanism that allows consumers to incrementally and interactively develop their requirements and preferences. The effectiveness of this type of approaches primarily arises from three facts. Firstly, example critiquing systems manage to effectively educate the user about available items, their characteristics and necessary tradeoffs between them. Secondly, with critiquing, they provide an intuitive mechanism to interact and communicate with the supporting recommender system. This stimulates and motivates the user and allows for an easy knowledge transfer between the user and the system. Finally, critiquing systems demonstrate that the requirements refinement process can be effectively directed and focused by leveraging knowledge about available items and the consumer's requirements. Unfortunately, the latter is achieved by letting the system take much of control, while restricting the user's opportunities, e.g of which critiques to pose [CP07b, CP07c, CP07a] or which item alternatives to view [VFP06, VPF07, VPF08]. In addition, there is no single system that joins all the beneficial characteristics described. Though knowledge about model uncertainty is available in some of the systems, it is not leveraged to focus the refinement process and to reduce model uncertainty.

In a sense, the solutions to utility elicitation, that we considered in our analysis, are complementary to those approaches. They actually do maintain knowledge about model uncertainty and also demonstrated how this information can be leveraged to effectively focus

[2]In the former case, the results are based on an assessment provided by 4 test persons, in the latter, they are grounded on a single test query. Hence their validity is questionable.

[2]Clearly, this depends on the underlying service description language.

[3]In the former case, the results are based on an assessment provided by 4 test persons, in the latter, they are grounded on a single test query. Hence their validity is questionable.

	Recommender Systems				Utility El.	SWS Retrieval				
	Bur.	Reil.	Viapp.	Chen	Cha./Bout.	Tools	Col.	Nopp.	Bal.	Zach.
Requirements Model										
includes requirements and preferences (R U.1)	○	○	○	+	+	$+^{2}$	-	+	+	○
semantic/rich/domain-ind. descr. mech. (R U.2)	-	-	-	-	-	+	+	-	○	-
represents/measures/locates uncertainty (R U.3)	-	○	+	-	+	-	-	-	-	-
Requirements Elicitation and Service Rec.										
considers uncertainty about reqs. (R U.4)	-	+	+	-	+	/	-	-	-	-
unified/incremental process (R U.5)	+	+	+	+	-	-	+	-	+	+
stimulates correct pref./requ. expr. (R U.6)	?	?	+	○	-	-	?	-	-	?
encourages to make/consider tradeoffs (R U.6)	○	+	-	+	+	-	-	-	-	-
alternative-based requirements spec. (R U.7)	+	+	+	+	-	-	-	-	-	-
contributes to requirements awareness (R U.8)	?	?	+	?	-	-	?	?	?	$+^{3}$
maintains model consistency (R U.9)	?	?	?	?	?	-	?	?	?	?
enables user to correct the model (R U.10)	+	+	+	+	-	+	+	+	-	+
effectively reduces model uncertainty (R U.11)	-	○	○	-	+	-	-	-	-	-
user-initiated req./pref. exp. (R U.12)	-	○	+	○	-	+	+	+	○	+
coherent elicitation process (R U.13)	○	○	?	○	-	+	+	+	-	+
educates about relevant services (R U.14)	?	○	○	○	-	-	+	○	○	$+^{3}$
motivates informed selection (R U.14)	?	○	+	+	-	/	?	?	?	?
Presentation of Information										
emph. relevant/important information (R U.16)	+	+	+	+	-	+	+	+	?	+
easy to perceive/comprehend format (R U.16)	+	+	+	+	-	-	+	+	?	+
facilitates consistent selection (R U.17)	?	?	?	?	-	/	?	?	?	?

Table 5.1.: Requirements to the requirements model, the requirements elicitation and service recommendation mechanism and the presentation of requirements- and service-related information

and direct the refinement process. However, this is at the cost of leaving no control to the user and imposing a purely reactive position on him. As a consequence, the user is not able to acquire knowledge about available opportunities and their characteristics and is not able to build a mental model of his requirements. This might lead to poor motivation and prevents the user from making well-informed and consistent decisions.

Approaches to Semantic Web Service Selection use little of the techniques and knowledge established in the two mentioned areas. Their capabilities are typically limited to provide assistance with the creation of service requests or to provide more sophisticated assistance with the definition of preferences and requirements. Unfortunately, the latter, more advanced type of solutions [NLS07, NLL$^+$06, WLN$^+$04, BW03b, BW03a, BW04] is not build upon rich semantic descriptions. The introduced approaches also neither account for the constructive nature of consumer requirements, nor consider knowledge about available service alternatives. One exception is the solution of Colucci et al. [CNS$^+$06, CNS$^+$04], which provides means for incremental query refinement based on intermediate service recommendations and devise mechanisms that support the user in this process. Unfortunately, they do not consider consumer preferences and still require considerable knowledge and capabilities of their users. For example, they do not allow for requirements specification based on the presented service alternatives. Moreover, none of the approaches to Semantic Web Service selection considers model uncertainty. Finally, it should be noted, that none of the mentioned solutions to assisted requirements specification has been adequately evaluated, if at all. Hence, it remains open whether those systems stimulate correct preference and requirement expressions, whether they maintain model consistency, whether they make the user aware of his requirements and whether they effectively assist the user in making well-informed and consistent selections. This is disappointing, particularly in case of the work presented by Zachos et al. [ZZMJ06, ZMHM08, sec06], that was accomplished within the EU-project SeCSE, which was funded with € 8,800,000.

Hence, the challenging questions of how to effectively support service consumers in incrementally developing their service requirements and preferences and how to assist them in making well-informed and balanced decisions remain. Thereby, the following research questions are of particular importance:

- How to educate the user about available service opportunities, their characteristics and necessary compromises with respect to them?
- How to support the user in constructing and correctly specifying his service requirements and preferences and how to do this in a way that results in well-constructed, i.e. true, preferences and requirements? Can the techniques that have been successfully applied in example critiquing systems be leveraged to achieve this?
- How to keep the system's knowledge about the consumer's requirements consistent with the user's actual requirements?
- How to effectively reduce uncertainty about the service consumer's requirements and preferences and how to do this while leaving the user in control of what to do?

- How to effectively assist the user in making well-informed and consistent service selection decisions?
- How to adequately evaluate a system for Semantic Web Service selection with respect to the requirements that have been identified in Section 5.1?

5.3 Interactive and Incremental Requirements Elicitation and Service Selection - Basic Idea

In compliance with the Requirements U.5 and U.10, we suggest a solution that implements requirements elicitation and service selection as a unified, incremental and interactive process that alternates phases of intermediate service recommendation and requirements refinement. During that process, the user incrementally develops his service requirements and preferences and finally makes a selection decision. To effectively support him in these tasks, the system maintains an internal model of the consumer's requirements and preferences (Requirement U.1), which we call *request model*. Uncertainty about the service consumer's true requirements is explicitly represented within this model (Requirement U.3). During the refinement process the request model is continuously updated to accurately reflect the system's growing knowledge about the user's service requirements and preferences. Starting with a generic request model, the system determines a set of service alternatives that fit well to this requirements (Requirements U.14 and U.17). These service alternatives are determined by transforming the internal request model into a semantic service request that reflects the requirements specified in the model, but also the system's uncertainty about this model. We will demonstrate that standard matchmaking with a minor extension can be applied to retrieve matching service results sorted by their expected matching degree (Requirement U.4). The user may then critique the presented service alternatives and thereby indicate desirable service characteristics (Requirements U.7 and U.10). He can also directly specify new service requirements by modifying the internal request model via its graphical representation (Requirement U.16). This will allow him to correct the system if necessary (Requirements U.9 and U.10), will help him to become aware of his requirements (Requirement U.8) and will enable him to actively develop these (Requirements U.6 and U.10). All user interactions trigger appropriate model changes (Requirement U.9). Once a set of modifications has been made, the user may decide to see service results fitting to the updated requirements. The process continues until the user finds an appropriate service among the presented alternatives or until he decides to stop without making a selection. To effectively reduce uncertainty about the user's service requirements (Requirement U.11) and thus to make the process of service selection more efficient, the system directs and focuses the process of requirements elicitation by suggesting those interaction opportunities to the user that have a high potential to increase the knowledge about the consumer's service requirements (Requirements U.12 and U.13). It thereby focuses on parts of the model that are relevant and promising in light of the available service options and in light of the requirements and preferences that have been already specified (Requirement U.11). In the subsequent sections, we provide details on our approach.

5.4 Request Model

We propose a request model that builds on DSD request descriptions (see Section 4.2) and that is in compliance with their semantics. Later on, this will allow us to use the DIANE matchmaker to compare the user's (uncertain) requirements with the offered service functionality. Though DSD descriptions are well suited for flexibly modeling service requirements and preferences (Requirements U.1 and U.2), they are not capable of representing uncertainty associated with the model. To compensate for that, we will propose appropriate extensions that allow to represent uncertainty about DSD's direct conditions and connecting strategies by means of probability distributions (Requirement U.3). At this time, we do not model uncertainty about type and attribute conditions, i.e. we assume that the types of acceptable service instances as well as their required attributes as specified in the request model reflect the user's true requirements[3]. Like DSD request descriptions, request models declaratively describe the set of service instances that are acceptable to a user by means of nested fuzzy sets. Thereby, the fuzzy membership value of an instance is interpreted as the user's preference for this instance. However, in contrast to DSD requests, there is uncertainty about a service consumer's actual preference for a certain service instance. As a consequence, a service instance's membership or preference value assigned to it by a request model is not a single, certain value as for standard DSD requests, but a random variable. This variable may take different values with certain probabilities. These probabilities indicate the likelihood of the service requester assigning a certain preference value to the given service instance. More formally, a request model is defined as follows.

Definition 5.2. (**Request model**) *A request model declaratively describes the set of service instances (service effects) that are acceptable to a service consumer by means of nested fuzzy sets. To each service instance, it assigns a probability distribution over possible membership or preference values from* $[0,1]$ *indicating the user's preference for this instance. More specifically, the preference value of a service instance with respect to a request model is given by a random variable that takes values from* $[0,1]$. *Thereby a preference value (membership value) of* 0 *means not acceptable (i.e. not a member of the set) and a preference value of* 1 *means totally acceptable. Preference values from* $(0,1)$ *indicate different preference levels. The higher the value, the stronger the preference. The set of acceptable service instances and their corresponding preference values can be specified by using the following descriptive elements, which will be subsequently defined: type conditions, uncertain direct conditions, attribute conditions as well as uncertain connecting strategies.*

[3]Nonetheless, we consider the fact, that attribute conditions that are relevant to the user might be not yet considered in the request model. This has been done by having appropriately defined the uncertainty measure that will be introduced in Section 5.8.

5.4.1. Type Conditions

As in DSD descriptions, a set defined in a request model has to specify a *type condition*, which indicates the ontological type of the instances that are elements of the set. As already noted, we do not consider uncertainty about the type of acceptable service instances, i.e. we assume, that the type indicated in the type condition is stated correctly. Hence, the random variable $Pref_I^{tc}(i)$ indicating the user's preference for a service instance i with respect to the type condition *tc* specified for a given set I takes a single value with probability 1.0. This value is 1.0, if the considered instance i is of the type specified in the type condition or a subtype of this type and otherwise is 0.0. More formally, in compliance with the semantics of DSD, a type condition is defined as follows:

Definition 5.3. (**Type condition**) *Let I be a set of instances. A mandatory type condition tc of I specifies the type of instances that belong to I. The preference $Pref_I^{tc}(i)$ for a service instance i with respect to tc of I is defined to be*

$$Pref_I^{tc}(i) = \begin{cases} 1.0 & \textit{if } i \textit{ is derived from the type specified in tc} \\ 0.0 & \textit{otherwise} \end{cases}$$

Valid types are specified in the service ontology.

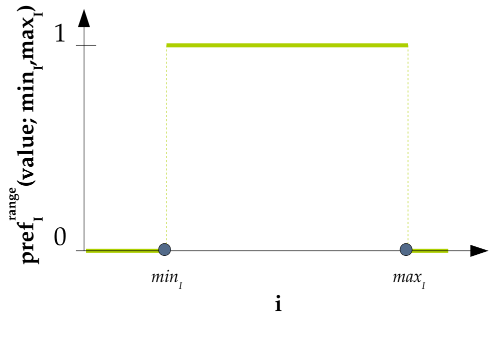
We also allow for alternative crisp (*super, super[n,1]*) and fuzzy type conditions (*super[n,f]*) as available in DSD (cf. Section 4.2). The definition of the random variable $Pref_I^{tc}(i)$ for those cases is straightforward and therefore omitted.

5.4.2. Uncertain Direct Conditions

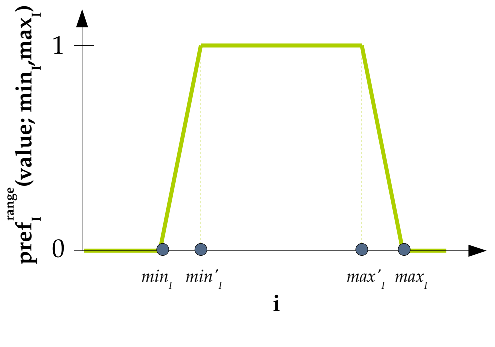
The request model supports three kinds of *uncertain direct conditions*, i.e. direct constraints on the instances of a set I (not on its attributes): uncertain range conditions as well as uncertain in- and not-in-conditions. An *uncertain range condition* defines a range of acceptable instances, e.g. a range of acceptable price values, by means of a preference (membership) function $pref_I^{range}(i; min_I, max_I)$ that assigns a preference value to each potential instance i. The preference function is parameterized with the minimum min_I and the maximum value max_I of the range. We do not make any assumptions about the type of this preference function. However, it should appropriately model the user's preferences.

Figure 5.8 shows two examples of preference functions over a range of instances. While the function depicted in Figure 5.8(a) assigns a preference value of 1.0 to all instances that lie in the range of acceptable instances, i.e.

$$pref_I^{range}(i; min_I, max_I) = \begin{cases} 1.0 & \text{if } (min_I \leq i < max_I) \\ 0.0 & \text{otherwise,} \end{cases} \tag{5.1}$$



(a)



(b)

Figure 5.8.: Possible preference functions over a range of instances

the preference function shown in Figure 5.8(b) assigns a preference value lower than 1.0 to instances that lie close to the limits of the range of acceptable instances, i.e.

$$pref_I^{range}(i; min_I, max_I) = \begin{cases} 1.0 & \text{if } (min'_I \leq i < max'_I) \\ \frac{i - min_I}{min'_I - min_I} & \text{if } (min_I \leq i < min'_I) \\ \frac{max_I - i}{max_I - max'_I} & \text{if } (max'_I \leq i < max_I) \\ 0.0 & \text{otherwise,} \end{cases} \tag{5.2}$$

where $min'_I = min_I + x \cdot |max_I - min_I|$ and $max'_I = max_I - x \cdot |max_I - min_I|$ and $x \in [0, 1]$ can be freely chosen. For $x = 0$, we obtain the preference function depicted in Figure 5.8(a) as a special case of the function shown in Figure 5.8(b).

The request model considers uncertainty about the parameters of the preference function, but not about the type of function. For that purpose, both, the minimum and the maximum of the range are modeled as random variables Min_I and Max_I, respectively, for which the request model maintains separate probability distributions p_{Min_I} and p_{Max_I}[4]. The probabilities $p_{Min_I}(i)$ and $p_{Max_I}(i)$ provide the likelihood of a certain instance i being the minimum and the maximum of the range, respectively.

Uncertain (not-)in-conditions allow the user to specify instances that are (not) acceptable (preference value of (0.0) 1.0) to him. In case of an uncertain in-condition, the user can also specify an alternative preference value for each acceptable instance. For example, he might indicate that he requires a printing service that delivers colored or black-and-white printouts, where the preference value for colored printouts is 0.8 and 0.2 for black-and-white printouts. Uncertainty about in-conditions is modeled by means of a discrete probability distribution $p_{In_i^I}$ for each potential instance i, where the probability $p_{In_i^I}(true)$ provides the likelihood of instance i being acceptable to the user (preference value of 1.0)

[4]The range R_I of potential minima and maxima is extracted from available service offers by using knowledge services.

and $p_{In_i^I}(false) = 1 - p_{In_i^I}(true)$ provides the likelihood of i being not acceptable (preference value of 0.0). By default, $p_{In_i^I}(true) = 0.0$ for all potential instances of a set for which an in-condition has been specified, i.e. in compliance with the semantics of DSD's in-conditions, we assume that a given instance is not contained in the set, if we do not have any information about the instance's set membership[5]. Alternative preference values $\{pref_I^{in}(i) | p_{In_i^I}(true) \neq 0.0\}$ for acceptable instances are user-provided. We do not consider uncertainty about the preference value of an instance. Similarly, uncertainty about not-in-conditions is modeled by maintaining a discrete probability distribution $p_{NotIn_i^I}$ for each possible instance i of I, where the probability $p_{NotIn_i^I}(true)$ provides the likelihood of instance i being not acceptable to the user and $p_{NotIn_i^I}(false) = 1 - p_{NotIn_i^I}(true)$ provides the likelihood of i being acceptable. By default, $p_{NotIn_i^I}(true) = 0.0$ for all possible instances of a set for which a not-in-condition has been specified, i.e. in compliance with the semantics of DSD's direct conditions of type *!= instance*, we assume that a given instance is contained in the set, if we do not have any information about the instance's set membership[5]. The preference value for instances that are not acceptable is 0.0, otherwise 1.0. A set may either specify a single uncertain in-condition or a single uncertain not-in-condition[6].

There may be more than one uncertain direct condition specified for a set. In those cases, an instance's (overall) preference value with respect to all direct conditions that are provided for the set is given by the product of this instance's preference values with respect to the single direct conditions. This is in accordance with the semantics of DSD, where an instance's preference values resulting from the evaluation of the different direct conditions that have been specified for a set are conjunctively combined, i.e. multiplied. This means, that an instance can only be an element of a given set, if it fulfills all direct conditions that have been specified for that set. If for a given instance set no direct conditions have been specified, the instance's (overall) preference value with respect to the direct conditions that have been specified for that set is 1, since unless proven to the contrary, we assume that all possible instances of the considered set are acceptable.

We formally define an uncertain direct condition as follows:

Definition 5.4. (**Uncertain direct condition**) *Sets may specify uncertain direct conditions, i.e. uncertain direct constraints on its instances. An uncertain direct condition dc_j of a set I specifies whether and to what degree potential instances of the set fulfill the constraint given by dc_j. The preference value $Pref_I^{dc_j}(i)$ of a service instance i with respect to dc_j of I is defined depending on the kind of direct condition.*

[5]This relieves us of having to maintain a probability distribution for each potential instance.

[6]Specifying both, an uncertain in- and an uncertain not-in-condition for a single set is not reasonable, since by default all instances that are not considered in a specified in-condition are not contained in the set and instances that are not considered in a specified not-in-condition are contained in the set. While in DSD, i.e. the certain case, specifying both types of direct conditions for a single set is not harmful (we either just provide redundant information or inconsistent information that lead to a preference value of 0.0 for the considered instance), it can lead to unresolvable inconsistencies in the request model, e.g. $p_{In_i^I}(true) \neq p_{NotIn_i^I}(false)$, in case of uncertain direct conditions.

- *The preference value* $Pref_I^{dc_j}(i)$ *of an instance* i *with respect to an uncertain range condition* dc_j *of a set* I *is given by*

$$Pref_I^{dc_j}(i) = pref_I^{range}(i; min_I, max_I) \text{ if } (Min_I = min_I) \text{ and } (Max_I = max_I), \tag{5.3}$$

where $pref_I^{range}(i; min_I, max_I)$ *is the system-specified range preference function for the set* I *with the minimum of the range being* min_I *and the maximum of the range being* max_I*.*

- *The preference value* $Pref_I^{dc_j}(i)$ *of an instance* i *with respect to an uncertain in-condition* dc_j *of a set* I *is given by*

$$Pref_I^{dc_j}(i) = \begin{cases} pref_I^{in}(i) & \text{if } (In_i^I = true) \\ 0.0 & \text{otherwise.} \end{cases} \tag{5.4}$$

- *The preference value* $Pref_I^{dc_j}(i)$ *of an instance* i *with respect to an uncertain not-in-condition* dc_j *of a set* I *is given by*

$$Pref_I^{dc_j}(i) = \begin{cases} 0.0 & \text{if } (NotIn_i^I = true) \\ 1.0 & \text{otherwise.} \end{cases} \tag{5.5}$$

A set may either specify a single uncertain in-condition or a single uncertain not-in-condition. The preference value $Pref_I^{dc}(i)$ *of an instance* i *with respect to all uncertain direct conditions that are specified for* I *is defined to be*

$$Pref_I^{dc}(i) = \begin{cases} 1.0 & \text{if no direct conditions have been specified for } I \\ \prod_{dc_j \text{ specified for } I} Pref_I^{dc_j}(i) & \text{otherwise,} \end{cases} \tag{5.6}$$

5.4.3. Attribute Conditions and Uncertain Connecting Strategies

As in DSD request descriptions, sets of non-primitive types (cf. Section 4.2) may specify conditions on the attributes of its potential instances. These *attribute conditions* refer to valid attributes defined for the ontological type that has been specified in the type condition of the set. They inherit the name of the corresponding attribute defined in the ontology and have a target set they refer to. Both, valid attributes of a type and valid target set types are specified in the service ontology. An instance can be an element of a given set, if the attributes specified in the attribute conditions of the set are specified for that instance and their values are elements of the corresponding target sets. Attribute conditions allow for the nested specification of declarative sets. By default, the preference values resulting from the attribute conditions that are specified for a set are conjunctively combined, i.e. multiplied. However, alternative connecting strategies might be specified.

Definition 5.5. (**Attribute condition**) *Sets of non-primitive types may specify attribute conditions, i.e. conditions on the attributes of its potential instances. They refer to attributes of the ontological type that has been specified in the type condition of the set and have a target set they refer to. The preference* $\mathit{Pref}_I^{ac_j}(i)$ *of the instance* i *with respect to the attribute condition* ac_j *of* I *is given by*

$$\mathit{Pref}_I^{ac_j}(i) = \begin{cases} 0.0 & \textit{if } ac_j \textit{ is not specified in } i \\ \mathit{Pref}_{I_j}^{ov}(i) & \textit{otherwise,} \end{cases} \tag{5.7}$$

where $\mathit{Pref}_{I_j}^{ov}(i)$ *is the (overall) preference value of the instance* i *with respect to the target set* I_j *of attribute condition* ac_j*, which will be subsequently defined.*

We also allow for the specification of missing strategies (*assume_fulfilled, assume_value[n]*) as available in DSD (Section 4.2). Those strategies modify the semantics of attribute conditions by allowing to assign a predefined, non-zero preference value with respect to an attribute condition to instances that do not specify the attribute required by this condition. The definition of the random variable $\mathit{Pref}_I^{ac_j}(i)$ for those cases is straightforward and therefore omitted.

A set may specify an *uncertain connecting strategy* that indicates how the preference values resulting from the evaluation of the single attribute conditions shall be combined. It thus encodes the user's preferences with respect to the relative importance of the specified attribute conditions. At this time, the request model implementation supports only weighted sum as a connecting strategy, where equal weights are assumed, if no connecting strategy has been explicitly specified. Such an additive connecting strategy (*additive preference function*) is appropriate to model a user's preferences with respect to a set of attribute conditions, if the latter are *mutually preferentially independent* [KR93]. More specifically, this means that for any subset of those attribute conditions, the preference order over service instances with varying preference values with respect to the subset conditions does not change, if the preference values with respect to the remaining attribute conditions change. The advantage of an additive preference model is that the single additive components of such a model can be treated separately. This allows for an efficient computation of preference values. Though mutual preferential independence is not a valid assumption for all decision problems, it is appropriate for many real-world decisions and works well in practice [KR93, FGE05]. As Keeney et al. argued, its applicability less depends on the decision problem itself, but rather on the appropriate choice of the objectives and attributes used to solve the problem [KvW07, Kee05]. Hence, restricting ourselves to connecting strategies of type weighted sum is a reasonable choice.

Though we presume, that the type of connecting strategy for a certain attribute is weighted sum, the parameters of the strategy, i.e. the weights, are unknown. Uncertainty about those parameters is again modeled by interpreting a connecting strategy's weights $W_{ac_1}, \cdots, W_{ac_n}$ as random variables. For each weight W_{ac_j}, a probability distribution $p_{W_{ac_j}}$ over the possible weights of attribute condition ac_j is maintained. Weights are absolute and taken from $[0, 1]$. The probability $p_{W_{ac_j}}(w)$ provides the likelihood of the weight for attribute condition ac_j being w.

Definition 5.6. (**Uncertain connecting strategy weighted sum**) *A set I may specify an uncertain connecting strategy of type weighted sum that indicates how the preference values resulting from the single attribute conditions shall be combined. Based on this connecting strategy, the preference value $Pref_I^{ac}(i)$ of an instance i with respect to all attribute conditions that are specified for I is defined to be*

$$Pref_I^{ac}(i) = \frac{\sum\limits_{ac_j \text{ specified for } I} W_{ac_j} \cdot Pref_I^{ac_j}(i)}{\sum\limits_{ac_k \text{ specified for } I} W_{ac_k}}, \tag{5.8}$$

where $Pref_I^{ac_j}(i)$ is the preference value of the instance i with respect to the attribute condition ac_j of I. $Pref_I^{ac}(i)$ is defined to be 1, if no attribute conditions have been specified for I.

The overall preference value of an instance with respect to a given set I is defined as the product of its preference values given by the type condition, the uncertain direct conditions and the attribute conditions specified for I. This is again in accordance with the semantics of DSD, where an instance's preference values resulting from the type condition, the direct conditions and the attribute conditions that have been specified for a set are conjunctively combined, i.e. multiplied.

Definition 5.7. (**Overall preference**) *The (overall) preference value $Pref_I^{ov}(i)$ of the instance i with respect to the set I is defined to be*

$$Pref_I^{ov}(i) = Pref_I^{tc}(i) \cdot Pref_I^{dc}(i) \cdot Pref_I^{ac}(i). \tag{5.9}$$

As a result, the preference value of a service instance with respect to a request model is given by its overall preference value with respect to the *effect*-set of the request model.

5.5 Visualizing Service Requirements

As argued in Section 5.1, requirements establish a user's interest in a decision situation and thus should guide the valuation of available decision opportunities. Clarifying those requirements and making them explicit prevents decision makers from basing their decisions on irrelevant, incomplete and/or ill-defined requirements and thus is essential for making thoughtful and reasonable decisions that are consistent with the user's actual objectives (Requirements U.8 and U.17). Systems aiming at the provision of support for service selection require profound and accurate knowledge about a user's known requirements in order to be able to provide effective assistance in that process (Requirement U.9). Ensuring this necessitates the decision maker's involvement in the knowledge acquisition process as well as his active participation in scrutinizing and correcting the system's requirements model if necessary (Requirement U.10). To enable this, internal knowledge about the user's requirements has to be presented in a format that makes it easy to perceive and easy to comprehend (Requirement U.16). Moreover, it has to be manipulable by means of simple and user-friendly interaction opportunities. In this section, we will propose a modifiable graphical representation of the internal request model that has been presented in the previous section. As we will demonstrate in our evaluation (Chapter 8), it will achieve both goals, offering potential service consumers appropriate means to contribute to and to correct the model and fostering the user's awareness of his service requirements.

5.5.1. Graphical Representation

The graphical representation adopts the tree-structure of the request model and arranges service requirements in a hierarchical way. As argued in [Kee92] structuring objectives helps decision makers to identify missing or redundant requirements and "promote[s] systematic and deep thinking about [them]" (Requirements U.8). Moreover, viewing requirements at different levels of abstraction as given in a hierarchy, reduces the complexity of the information at hand and thus makes them easier to comprehend. To improve the perceiv-

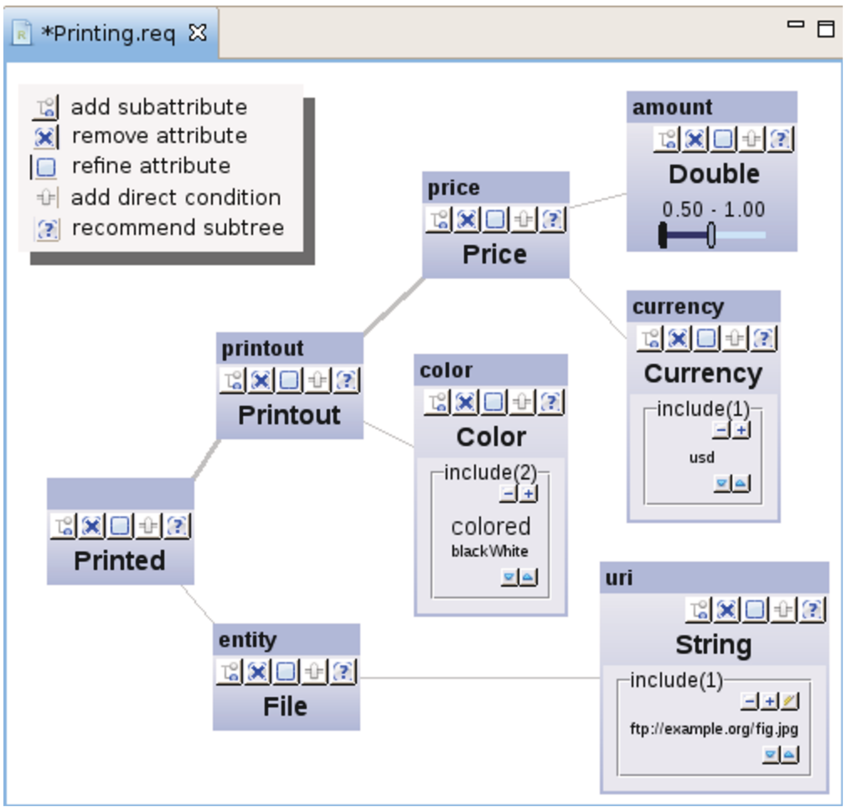


Figure 5.9.: Graphical model representation

ability of the presented information, to avoid information overflow and to facilitate their manipulation, the proposed model visualization simplifies it by hiding unnecessary details and, if possible, applies visual metaphors to illustrate different aspects (Requirement U.16). Figure 5.9 shows our model visualization using the example of a request model referring to a printing service. Attribute conditions are displayed as boxes containing the attribute name in the menu bar and the type of the target set as a string in the box (Figure 5.10). We introduced user-friendly terms for both, attribute and type names, to abstract from the concept and role names in the ontology that are sometimes difficult to comprehend.

Users can refine the type of an attribute condition's target set, i.e. restrict it to a subtype. On pressing the appropriate button (see the key displayed in the upper left corner of Figure 5.9) in the considered attribute condition's toolbar (Figures 5.9 and 5.10), a dialog with an alphabetically sorted list of possible subtypes (as defined in the underlying ontology) appears, from which the user can select (cf. Figure 5.11(b)). Straight lines emanating from the attribute condition boxes indicate nested attribute conditions. They can also be added by the user. By pressing the corresponding button in the considered attribute condition's toolbar (Figure 5.9), a dialog with an alphabetically sorted list of potential conditions (as defined within the underlying ontology) appears. Subtypes and attribute conditions that do not actually appear in the available offers are hidden from the user, since they do not qualify for evaluating and comparing available service alternatives and thus are not relevant in the given decision context.

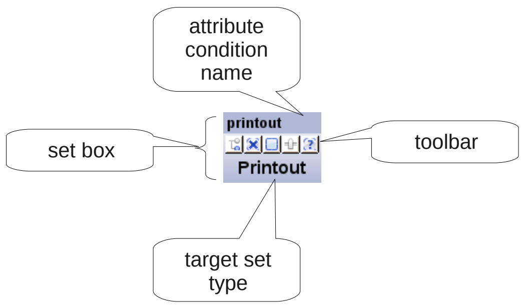


Figure 5.10.: Elements of the graphical request model representation

Graphical representations of uncertain range-, in- and not-in-conditions as implemented within the request model are also displayed and means to specify them in a graphical way are provided. Uncertain range conditions are visualized by a range slider and a label indicating the boundaries of the acceptable range (attribute condition *amount* in Figure 5.9). The latter are determined by the instances that are most likely to be the minimum and the maximum of the range, respectively. In case of having more than one instance that qualifies as a minimum or maximum, we choose the boundaries in a way that maximizes the acceptable range, since we simply do not know whether the range should be more restricted. Editable black- and whitelists are used to represent uncertain (not-)in-conditions.

As an example, consider the attribute condition *color* in Figure 5.9, that defines acceptable printout colors within a whitelist. As can be seen, such a list is visualized as a box, that the user can fill with (potentially) acceptable instances, i.e. instances with $p_{In_i^I}(true) \neq 0$. This is either done by pressing the +-button or the *edit*-button (see attribute condition *uri* in Figure 5.9). If any named instances of a certain target set type are specified in the ontology, pressing the +-button will provide a list of those instances for selection. By pressing the *edit*-button, the user can specify user-defined instances. The latter makes only sense for value types, i.e. types, where arbitrary values of this type or any combination of attribute values form a valid instance. For instance, the set of acceptable URIs (attribute condition *uri* in Figure 5.9) is of type *String* and any character string will form a valid instance (at least with respect to the underlying ontology). For the purpose of clarity and to avoid information overflow, only a user-defined number of whitelist-/ blacklist-elements is visualized at once. The total number of elements is indicated in brackets after the word *Include* (whitelists) or *Exclude* (blacklists). The whole list can be scrolled by using the scroll buttons on the lower right corner of the list box. Users can also express preferences over the list elements. By double-clicking on a list element its preference value is increased by a certain amount. A higher preference is indicated by a larger font size. Consider for instance the attribute condition *color* in Figure 5.9, where the value *colored* is preferred over *blackWhite*.

We already emphasized (Section 5.1), that among all kinds of preferences covered in the request model, those that weight different attributes conditions against each other, i.e. those expressed in the connecting strategies, are of particular importance and should be presented to the user. This is because they enable the user to make explicit compromises between service aspects (Requirement U.6). As already mentioned, at this time, we just support weighted sum as an uncertain connecting strategy. In the graphical representation, the most likely values of the attribute conditions' weights are displayed and encoded by the width of the line ending at the considered attribute condition. Thicker lines indicate a higher absolute weight. By double-clicking a certain connection, the weight of the corresponding attribute condition is increased. In doing so, a user can indicate the relative importance of attribute conditions in a user-friendly way (Requirement U.10). For example in the request model visualization depicted in Figure 5.9, the user indicated that *price* is three times as important as *color.* Finally, by double-clicking on a certain attribute condition box, users can collapse and expand subtrees of the request model to save space and keep track of the essential parts of the request model (Requirement U.16).

5.5.2. Advanced User Support

The presented model visualization tries to reduce the mental effort and knowledge that is required to specify and modify service requirements maintained in a request model by restricting available model elements to a manageable number and by allowing to describe service requirements on an informal level (Requirement U.16). Nonetheless, the typical end-user might still have difficulties with finding out how to model a certain service requirement. In our opinion, this is mainly due to two reasons. Firstly, the number of ontological types that attribute conditions' target sets might have is often large. Hence, when refining a target set's type, the number of subtypes the user has to choose from is sometimes high, which makes it difficult to find the most appropriate selection. Secondly, though the number of available model elements is restricted, it sometimes might not be clear how a certain requirement can be modeled. This is particularly true for users that are not familiar with ontologies and ontology-based modeling. Our graphical representation comes with two recommendation tools intended to help users to alleviate the mentioned problems, a subtype recommender and a subtree recommender. In the remainder of this section, we will describe those recommenders in more detail.

Subtype Recommender

The subtype recommender leverages knowledge extracted from request models that have been created by a service consumer (or other service consumers, if permitted) in the past to recommend subtypes to a certain target set's type that might be appropriate in the context of the present model. Those subtypes are presented to the user in addition to the alphabetically sorted list of all applicable subtypes (cf. Figure 5.11(b)). In doing so, scrolling through a potentially large list of subtypes can be avoided in many cases. In the following, we will describe the recommendation algorithm in more detail.

Let r be the request model under construction and I a set that is referenced in r and whose type $\mathit{type}(I)$ shall be refined, i.e. subtyped. Let further $\mathit{path}_r(I) = \langle[\mathit{name}(I_1), \mathit{type}(I_1)]. \cdots .[\mathit{name}(I_n), \mathit{type}(I_n)]\rangle$ be the path of this set in r comprising of the sequence of pairs $[\mathit{name}(I_k), \mathit{type}(I_k)]$, $1 \leq k \leq n$, where $\mathit{name}(I_k)$ is the name of the kth attribute condition lying on the tree path from r's *effect*-set to I and $\mathit{type}(I_k)$ is its type. Thereby, I_1 corresponds to the *effect*-set of r and I_n corresponds to the considered set I (cf. Definition 6.4). The basic idea of the recommendation algorithm is to recommend those valid and occurring (in an available service offer) subtypes of $\mathit{type}(I)$ that have been assigned to a set I_{past} referenced in a past request model and whose path differs from that of I just by the type $\mathit{type}(I_{past})$ of I_{past}. However, recommendations are not restricted to the types that directly appeared in past request models, but may also refer to their sub- and supertypes as defined in the ontology. Thereby, the semantic similarity of the past request model that contributed a certain type and that of the request model under construction, i.e. the usage context of the type, is considered. Let $S_{subtype}$ be the set of session data referring to the creation of past request models and comprising of pairs (r', t'), where r' is a past request model that references a set I_{past} whose path differs from that of I just by the type $t' = \mathit{type}(I_{past})$. Then T_{cand}, given by

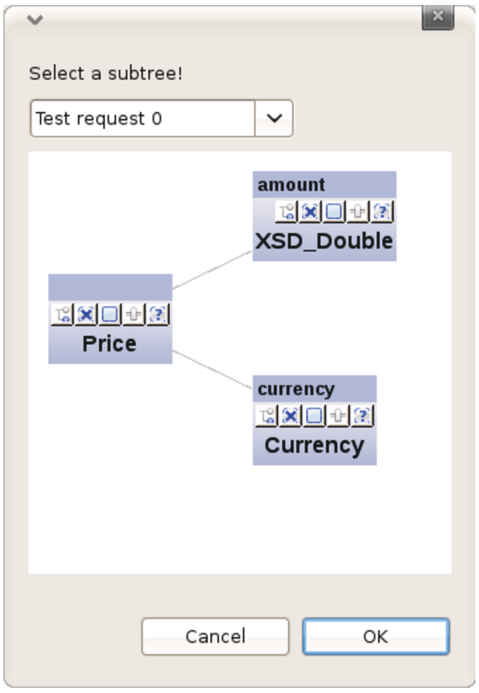
$$T_{cand} = \{t | \exists (r', t') \in S_{subtype} \textit{ with } t' = t \ \wedge \ t' \textit{ is an occurring subtype of } \mathit{type}(I)\}, \tag{5.10}$$

is the set of candidate types that qualify for being recommended as a subtype of $\mathit{type}(I)$.

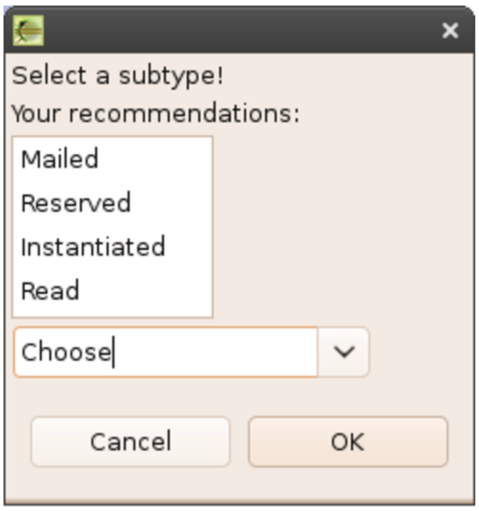
To each candidate type $t \in T_{cand}$, we assign a value $v_{suit}(t) \in [0,1]$ indicating t's suitability as a subtype of *type*(I) in r. It is defined as the mean semantic similarity of r and the past request models that contribute t, i.e.

$$v_{suit}(t) = \frac{\sum_{r' \ with \ \exists(r',t') \in S_{subtype} \wedge t'=t} sim_{req}(r', r)}{|\{r' | \exists(r',t') \in S_{subtype} \wedge t' = t\}|}, \tag{5.11}$$

where sim_{req} is a measure indicating the semantic similarity $sim_{req}(r', r)$ of two request models r' and r. It will be introduced in Section 6.5.3 of this thesis. Possible similarity values range from 0 to 1, where a similarity value of 0 means ”not similar at all” and a value of 1 means that the service requirements encoded by the two input request models are semantically identical. Values lying inbetween indicate different degrees of semantic similarity. The rationale behind the provided suitability definition is, that a candidate subtype is the more suitable, the more similar the usage contexts (i.e. the request models) in which it was applied in the past are to the present. The suitability values induce an order on the recommendation candidates. The higher the value, the more promising the candidate.



(a)



(b)

Figure 5.11.: Subtree recommender (a) and subtype dialog (b)

Beside considering types that directly appeared in a past request model for recommendation, we also take sub- and supertypes of those types as potential recommendation candidates into consideration. In particular, elements contained in the set T_{cand}^{sub} of valid subtypes of the elements in T_{cand} as defined in the ontology are also valid recommendation candidates. To each of them we assign a suitability value that is equal to that of its supertype contained T_{cand}. The reason behind that choice is that if a certain type is suitable to describe a user's requirements, then all of its subtypes have the potential to be equally suitable as well. As an example, consider a past request model referring to a file download. Imagine, there was a specified attribute condition restricting the type of files that were acceptable candidates for downloading to those of the type *ImageFile*. Having a present model also referring to a file download, that does not yet restrict acceptable kinds of files further than to those of type *File*, the type *ImageFile* is certainly a good subtype candidate. All of its subtypes would also be good candidates. However, we simply do not know, which of *ImageFile*'s subtypes, e.g. *Jpg*, *Png* or *Gif*, are preferred by the user. This argument does not hold for supertypes, i.e. the fact that *ImageFile* is a suitable choice for the user does not necessarily imply that its supertype *File* is equally suitable. However, if *ImageFile* is an appropriate type, then also some of its siblings, e.g. *VideoFile*, might be suitable. Hence, supertypes of types that have been applied in the past are also a good choice for recommendation. However, the larger the ontological distance between the supertype and the type that has been applied, the less likely it is that many of its subtypes are suitable choices as well. Hence, the weight of supertypes should decrease with the distance from the type that has been directly applied. We propose the following approach. Let T_{cand}^{super} be the set of supertypes defined for the elements in T_{cand} which are also subtypes of *type*(I). Let further $t_{super} \in T_{cand}^{super}$ be a supertype of $t \in T_{cand}$. Then its suitability is defined to be

$$v_{suit}(t_{super}) = v_{suit}(t) * 0.5^j, \tag{5.12}$$

where j is the number of levels that lie between t_{super} and t in the type hierarchy (including t_{super}).

Once T_{cand}, T_{cand}^{sub} and T_{cand}^{super} are determined, we select those of its elements that have the highest suitability value assigned and recommend them to the user[7]. The list of recommended types is sorted by decreasing suitability. Note, that instead of or in addition to base the recommendation algorithm on knowledge derived from past requests models, we can also apply it to a set of request model templates provided by an application designer.

Subtree Recommender

Sometimes users might not know how to model a certain aspect of their requirements, e.g. considering the price of a desired service as depicted in Figure 5.9, it might not be clear how to actually model a certain price constraint. In those situations, the subtree recommender can be helpful. Given a set referenced in the request model under construction, the subtree recommender suggests potential model subtrees, i.e. potential attribute conditions, for that set. For instance, the attribute conditions depicted in Figure 5.11(a) might be suggested

[7]Should a type be element of more than one of the sets, its suitability is given by the highest suitability value assigned to it with respect to one of the sets.

for the *price*-set of a given request model. The recommendations are based on knowledge about attribute conditions that have been added to that set in past request models when having similar service requirements. As in case of the subtype recommender, we may consider just those request models that have been created by the user in the past or also those contributed by other users. We may also use request model templates provided by an application designer as a knowledge base. The subtree recommendation feature does not only help to overcome difficulties with how to model a certain requirement, it also fosters unified modeling by reusing past models. Moreover, it has the potential to fasten the requirements specification process by adding whole subtrees with a single click.

Let r be the request model that is currently under construction and let I be the set referenced in r for which a subtree recommendation is required. Let further $S_{subtree}$ be the set of session data referring to the creation of past request models and comprising of pairs (r', st'), where r' is a past request model that references a set I_{past} that shares the same path in r' as I in r and *st'* is the request model subtree rooted at I_{past} in r'. Then T_{cand}, given by

$$T_{cand} = \{st | \exists (r', st') \in S_{subtree} \text{ with } st' = st\}, \tag{5.13}$$

is the set of candidate subtrees that qualify for being recommended as a subtree of I in r. To each candidate subtree $st \in T_{cand}$, we assign a value $v_{suit}(st) \in [0, 1]$ indicating *st*'s suitability as a subtree of I in r. It is defined as the average semantic similarity of the request model r that is currently constructed and the past request models that contributed the considered subtree *st*, i.e.

$$v_{suit}(st) = \frac{\sum_{r' \text{ with } \exists (r', st') \in S_{subtree} \wedge st' = st} sim_{req}(r'|st', r)}{|\{r' | \exists (r', st') \in S_{subtree} \wedge st' = st\}|}, \tag{5.14}$$

where sim_{req} is the request model similarity measure already used in the subtype recommender and $r'|st'$ the past request model that contributed st' without the subtrees rooted at the root node of st'. The rationale behind the definition is, that a candidate subtree is the more suitable, the more similar the usage contexts (i.e. the request models) in which it was applied in the past are to the present. The suitability values induce an order on the recommendation candidates. The higher the value, the more promising the candidate.

Once subtree candidates and their suitability values are determined, we select those that have the highest suitability and recommend them to the user. The list of recommended subtrees is sorted by decreasing suitability (see Figure5.11(a)). A user can open the subtree recommender by pressing the *?*-button in the toolbar of the box representing the considered set. Finally, we like to mention, that considering knowledge derived from past request models created by third parties would raise serious privacy issues, if we had access to complete request models of those users. Hence, past request models are anonymized by removing all direct conditions and clearing all connecting strategies before providing them to others. The latter means, that all sets referenced in the contributed request model specify a connecting strategy of type weighted sum, that assigns equal weights to the attribute conditions that have been specified for it. Note, that in doing so, contributed request models as well as recommended subtrees do not contain knowledge about consumer preferences and thus reveal less sensitive information of its contributor. This is not only desirable from a privacy perspective, but also in light of the fact that it is unlikely that a particular constellation of direct conditions and connecting strategies will be applicable to the current request model. Moreover, due to the amount of information provided, comparing such detailed subtrees to make a selection would be more difficult for the user than merely comparing different subtree structures without any direct conditions. We would also like to remark that the introduced graphical representation as well as the recommendation tools can be easily adjusted to work with standard DSD requests instead of request models. An appropriate measure indicating the semantic similarity of DSD service requests has been suggested and published in [FK10].

5.6 Matching Uncertain Requirements

To understand how matchmaking based on uncertain requirements as specified in a request model can take place (Requirement U.4), we first have to recap how certain, i.e. standard DSD service requests, are compared with available service offers. For a detailed discussion of this topic see Section 4.3. As already mentioned, we consider matchmaking of offered service instances rather than matchmaking of arbitrary service offers. In the DIANE matchmaker, the comparison of the acceptable service effect(s) described in a service request r and that of an offered service instance i is recursive and proceeds as follows. Starting from the *effect*-set of the request, the matchmaker checks in each step, whether the service effect described in the offer fulfills the conditions in the request. Proceeding to the leaf sets of the request results in preference values for the offered instance with respect to those sets. In a final pass, those values are incrementally aggregated to an overall preference value $pref^{ov}_{I_{effect}}(i)$ for the offered service instance, i.e. to a preference value with respect to the *effect*-set I_{effect} of the request[8]. This preference value is returned as matching value $mv(r,i)$ of the request and the offered service instance. It is recursively defined as the product of the instance's preference value $pref^{tc}_{I_{effect}}(i)$ with respect to the type condition tc specified for the *effect*-set, the instance's preference value $pref^{dc}_{I_{effect}}(i)$ with respect to the direct conditions that have been specified for the *effect*-set and the instance's preference value $pref^{ac}_{I_{effect}}(i)$ with respect to the attribute conditions specified for I_{effect}. While $pref^{dc}_{I_{effect}}(i)$ is determined as the product of the instance's preference values with respect to the single direct conditions that have been specified for the *effect*-set, the latter is an aggregate of the offered instance's preference values with respect to the target sets of the attributes that have been specified for I_{effect}. It is determined by the connecting strategy that has been specified for the *effect*-set.

In contrast to those in a DSD request, the service requirements specified in the request model are uncertain. In particular, there is uncertainty about the constraints and preferences

[8]For the sake of clarity, we omit an index indicating the request model that references the set.

specified in the direct conditions as well as uncertainty about the importance of attribute conditions, i.e. the weights of the connecting strategies (cf. Section 5.4). Hence, uncertain matchmaking can only deliver an expected matching value $\mathbb{E}[MV(r,i)] = \mathbb{E}[Pref^{ov}_{I_{effect}}(i)]$ for an offered service instance i and a given request model r. In the remainder of this section, we will show that the DIANE matchmaker can be utilized to determine this value and that only minor changes to the matchmaking mechanism are required to implement this.

Let I be a set that is referenced in the request model r. Then the partial matching degree $MV_I(r,i)$ of a given service instance i and r with respect to I is given by

$$\mathbb{E}[MV_I(r,i)] = \mathbb{E}[Pref^{ov}_I(i)] \tag{5.15}$$

Using Definition 5.7 as well as the facts that $Pref^{tc}_I(i)$ is certain and as such a constant and that $Pref^{dc}_I(i)$ and $Pref^{ac}_I(i)$ are statistically independent, we obtain

$$\begin{aligned}\mathbb{E}[Pref^{ov}_I(i)] &= \mathbb{E}[Pref^{tc}_I(i) \cdot Pref^{dc}_I(i) \cdot Pref^{ac}_I(i)]\\ &= Pref^{tc}_I(i) \cdot \mathbb{E}[Pref^{dc}_I(i) \cdot Pref^{ac}_I(i)]\\ &= Pref^{tc}_I(i) \cdot \mathbb{E}[Pref^{dc}_I(i)] \cdot \mathbb{E}[Pref^{ac}_I(i)].\end{aligned} \tag{5.16}$$

This is due to the linearity of the expectation operator and the fact that the expected value of the product of two statistically independent random variables can be written as the product of the variables' expected values.

Let $\{dc_j | 1 \le j \le m\}$ be the direct conditions that have been specified for I. Using the fact that $Pref^{dc_j}_I(i)$ and $Pref^{dc_k}_I(i)$ are statistically independent for any two direct conditions dc_j and dc_k of I with $j \ne k$[9], we obtain that the expected preference value $\mathbb{E}[Pref^{dc}_I(i)]$ of a given instance i with respect to the direct conditions that have been specified for I is given by

$$\begin{aligned}\mathbb{E}[Pref^{dc}_I(i)] &= \mathbb{E}[\prod_{j=1}^{m} Pref^{dc_j}_I(i)]\\ &= \prod_{j=1}^{m} \mathbb{E}[Pref^{dc_j}_I(i)].\end{aligned} \tag{5.17}$$

Let $\{ac_j | 1 \le j \le n\}$ be the attribute conditions that have been specified for I. Using Definition 5.6 and the facts that the normalized weight of any attribute condition ac_j is

[9]We implicitly assume here that a given instance i is constrained by at most one of the direct conditions that have been specified for a given set.

statistically independent from an instance i's preference value $\mathit{Pref}_I^{ac_j}(i)$ with respect to that condition and that the weights of any two attribute conditions ac_j and ac_k of I with $j \neq k$ are also statistically independent, it can be easily shown that the expected preference value $\mathbb{E}[\mathit{Pref}_I^{ac}(i)]$ of a given instance i with respect to the attribute conditions that have been specified for I is given by

$$\begin{aligned}
\mathbb{E}[\mathit{Pref}_I^{ac}(i)] &= \mathbb{E}[\frac{\sum_{j=1}^{n}(W_{ac_j} \cdot \mathit{Pref}_I^{ac_j}(i))}{\sum_{k=1}^{n} W_{ac_k}}] \\
&= \sum_{j=1}^{n} \mathbb{E}[\frac{W_{ac_j}}{\sum_{k=1}^{n} W_{ac_k}} \cdot \mathit{Pref}_I^{ac_j}(i)] \\
&= \sum_{j=1}^{n} (\mathbb{E}[\frac{W_{ac_j}}{\sum_{k=1}^{n} W_{ac_k}}] \cdot \mathbb{E}[\mathit{Pref}_I^{ac_j}(i)]) \\
&= \sum_{j=1}^{n} (\mathbb{E}[\frac{1}{\frac{\sum_{k=1}^{n} Wac_k}{Wac_j}}] \cdot \mathbb{E}[\mathit{Pref}_I^{ac_j}(i)]) \\
&= \sum_{j=1}^{n} (\frac{1}{\mathbb{E}[\sum_{k=1}^{n} \frac{Wac_k}{Wac_j}]} \cdot \mathbb{E}[\mathit{Pref}_I^{ac_j}(i)]) \\
&= \sum_{j=1}^{n} (\frac{1}{\sum_{k=1,k\neq j}^{n} \frac{\mathbb{E}[Wac_k]}{\mathbb{E}[Wac_j]} + 1} \cdot \mathbb{E}[\mathit{Pref}_I^{ac_j}(i)]), \qquad (5.18)
\end{aligned}$$

where $\mathbb{E}[W_{ac_j}]$, $1 \leq j \leq n$, is the expected absolute weight of attribute condition ac_j and $\mathbb{E}[\mathit{Pref}_I^{ac_j}(i)]$ is the expected preference value of i with respect to that condition. This is again due to the linearity of the expectation operator and its multiplicativity in case of statistically independent random variables.

Hence, assuming that the DIANE matchmaker is provided with the expected preference values $\{\mathbb{E}[\mathit{Pref}_I^{dc_j}(i)] \mid 1 \leq j \leq m\}$ of i with respect to the direct conditions that have been specified for I, it will return the desired expected partial matching degree $\mathbb{E}[MV_a(r,i)] = \mathbb{E}[\mathit{Pref}_I^{ov}(i)]$ of service instance i and r with respect to I, when receiving a standard DSD service request that specifies a connecting strategy of type weighted sum for the set I whose weights $\{w_{ac_j} | 1 \leq j \leq n\}$ for the attribute conditions that have been specified for I are defined by

$$w_{ac_j} = \frac{1}{\sum_{k=1,k\neq j}^{n} \frac{\mathbb{E}[Wac_k]}{\mathbb{E}[Wac_j]} + 1}. \qquad (5.19)$$

This is convenient, since we achieve the desired matchmaking functionality by simply transforming the given request model into an appropriate standard DSD request r'. We do not have to make any changes to the matchmaker's implementation. Unfortunately, it will turn out that the expected preference values $\{\mathbb{E}[\mathit{Pref}_I^{dc_j}(i)] | 1 \leq j \leq m\}$ of i with respect to the direct conditions that have been specified for I cannot always be computed by using available matchmaking functionality.

Let dc_j be an uncertain in-condition specified for I (Definition 5.4). The expected preference value $\mathbb{E}[Pref_I^{dc_j}(i)]$ of i with respect to dc_j is given by

$$\begin{aligned}\mathbb{E}[Pref_I^{dc_j}(i)] &= p_{In_i^I}(true) \cdot pref_I^{in}(i) + p_{In_i^I}(false) \cdot 0.0 \\ &= p_{In_i^I}(true) \cdot pref_I^{in}(i). \end{aligned} \tag{5.20}$$

Consequently, the desired matchmaking functionality can be accomplished by adding for each uncertain in-condition an appropriate DSD direct condition to the DSD request r' that is generated from the input request model r. Let dc_j be an uncertain in-condition specified for the set I in r. Since $p_{In_i^I}(true) = 0$ and thus $\mathbb{E}[Pref_I^{dc_j}(i)] = 0$ holds for all instances that have not been considered in dc_j, it is sufficient to add a direct condition dc'_j of type *IN* $\{i_1[pref(i_1)], \ldots, i_t[pref(i_t)]\}$ to the set I of r', where $\{i_1, \ldots, i_t\}$ are the instances that have been considered in the uncertain in-condition dc_j, and $\{pref(i_1), \ldots, pref(i_t)\}$ are their corresponding preference values with $pref(i) = p_{In_i^I}(true) \cdot pref_I^{in}(i)$.

Unfortunately, this does not work for uncertain not-in-conditions (Definition 5.4). Let dc_j be an uncertain not-in-condition specified for I. The expected preference value $\mathbb{E}[Pref_I^{dc_j}(i)]$ of i with respect to dc_j is given by

$$\begin{aligned}\mathbb{E}[Pref_I^{dc_j}(i)] &= p_{NotIn_i^I}(true) \cdot 0.0 + p_{NotIn_i^I}(false) \cdot 1.0 \\ &= p_{NotIn_i^I}(false). \end{aligned} \tag{5.21}$$

Since DSD does not provide a fuzzy *!=*-operator, we would have to implement the required matchmaking behavior by adding for each uncertain not-in-condition an appropriate DSD direct condition using the *IN*-operator to the DSD request r' that is generated from the input request model r. Since by default, $\mathbb{E}[Pref_I^{dc_j}(i)] = p_{NotIn_i^I}(false) = 1$ for all instances that are not considered in an uncertain not-in-condition, the generated direct condition would have to list all potential instances of the set for which the uncertain not-in-condition has been specified. Usually, this is not feasible. As a consequence, we had to introduce a new operator *uNOT_IN* to the DIANE matchmaker that implements the desired matchmaking functionality for not-in-conditions as given by Formula 5.21.

Finally, let dc_j be an uncertain range condition specified for I (Definition 5.4). Presuming that the system-specified range preference function $pref_I^{range}(i; min_I, max_I)$ for the set I is given by the function defined in Equation 5.1, the expected preference value $\mathbb{E}[Pref_I^{dc_j}(i)]$ of i with respect to dc_j is given by

$$\begin{aligned}\mathbb{E}[Pref_I^{dc_j}(i)] &= Prob(Min_I \leq i < Max_I) \cdot 1.0 + (1 - Prob(Min_I \leq i < Max_I)) \cdot 0.0 \\ &= Prob(Min_I \leq i < Max_I) \\ &= Prob(Min_I \leq i) \cdot Prob(i < Max_I) \\ &= \int_{z=\min(R_I)}^{i} p_{Min_I}(z)dz \cdot (1 - \int_{z=\min(R_I)}^{i} p_{Max_I}(z)dz). \end{aligned} \tag{5.22}$$

Since it is not feasible to precalculate this value for all instances that might potentially appear in an offer, we have to supply the matchmaker with another new operator-routine *uRANGE* that computes this preference value for arbitrary instances to implement this.

Summarizing, we can state that matching uncertain service requirements as specified in the request model can be implemented by generating a standard DSD service request with the properties detailed above and matching it with a slightly modified version of the standard matchmaker. The overall matchmaking procedure can be outlined as follows:

Matching uncertain requirements

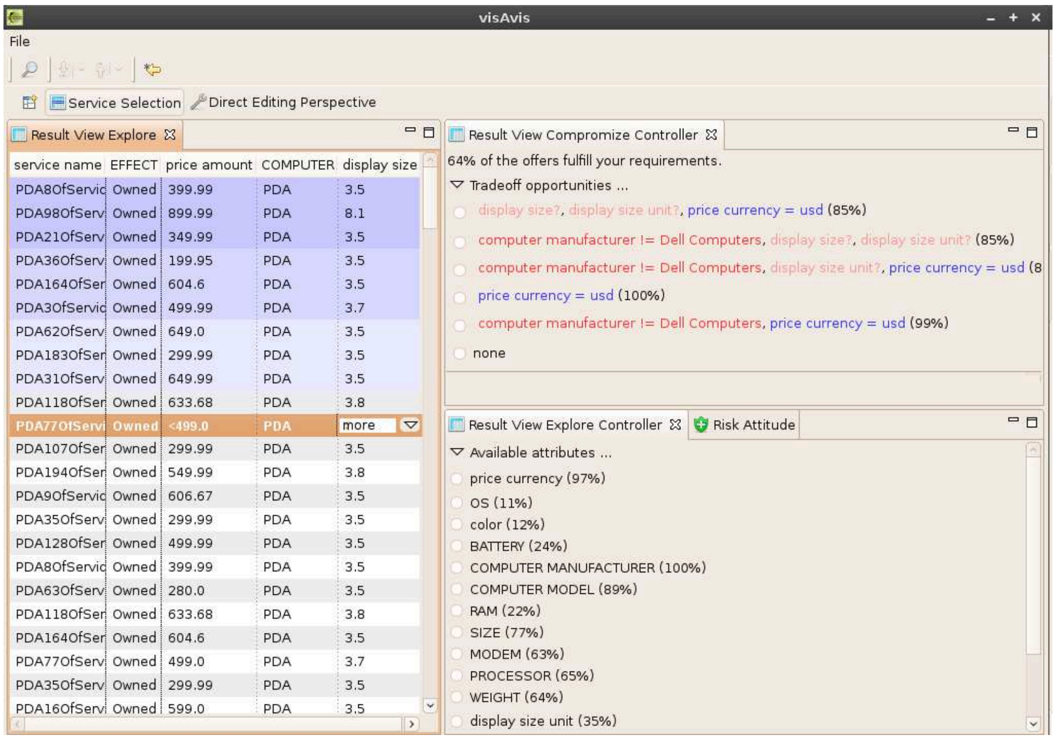
1. transform the given request model r into a DSD service request r'
   a) create a service request r' that inherits the sets referenced in r including their type and attribute conditions
   b) for each uncertain connecting strategy specified in r, add an appropriate connecting strategy of type weighted sum to r'
   c) for each uncertain in-condition specified in r, add an appropriate direct condition to r' using DSD's *IN*-operator
   d) for each uncertain not-in-condition specified in r, add an appropriate direct condition to r' using the new *uNOT_IN*-operator
   e) for each uncertain range condition specified in r, add an appropriate direct condition to r' using the new *uRANGE*-operator
2. match the generated service request r' using the modified DIANE matchmaker supporting the new operators *uNOT_IN* and *uRANGE*

5.7 Presenting Matching Service Offers and their Characteristics to Encourage Requirements Specification

After having particularized how potentially matching service offers can be determined based on uncertain requirements, the focus of this section is on how those offers can be appropriately presented to the user (Section 5.7.1) and how the user can be enabled to specify additional service requirements based on the displayed service alternatives and their characteristics (Section 5.7.2).

5.7.1. Presenting Matching Service Offers and their Characteristics

Once the service offers, that match to the requirements which are encoded in the request model, have been retrieved, the list of those offers sorted by their expected matching value is presented to the user (see Figure 5.12 left). Thereby, the most relevant offers, i.e. those receiving the highest expected matching degree, are displayed on top (Requirement U.14). The top ten results are highlighted. This is to focus the decision maker's attention to highly desirable alternatives, which, according to [Kee92], stimulates thinking about relevant and beneficial service characteristics and thus about potential service requirements. The depicted results table contains a row for each matching service alternative, which includes the name of the offered service and a column for the target set of each attribute condition that has been specified within the request model and is visible, i.e. non-collapsed, in the graphical request model representation. By using elements of the request model to structure and describe available service opportunities, we facilitate the comparison of requirements and service alternatives and thus support decision makers to correctly translate their preferences and requirements into a selection [PBS99], i.e. to make a consistent selection without any biases (Requirement U.17). The cells of a results table column show



service name	EFFECT	price amount	COMPUTER	display size
PDA8OfServic	Owned	399.99	PDA	3.5
PDA98OfServ	Owned	899.99	PDA	8.1
PDA21OfServ	Owned	349.99	PDA	3.5
PDA36OfServ	Owned	199.95	PDA	3.5
PDA164OfSer	Owned	604.6	PDA	3.5
PDA3OfServic	Owned	499.99	PDA	3.7
PDA62OfServ	Owned	649.0	PDA	3.5
PDA183OfSer	Owned	299.99	PDA	3.5
PDA31OfServ	Owned	649.99	PDA	3.5
PDA118OfSer	Owned	633.68	PDA	3.8
PDA77OfServi	Owned	<499.0	PDA	more
PDA107OfSer	Owned	299.99	PDA	3.5
PDA194OfSer	Owned	549.99	PDA	3.8
PDA9OfServic	Owned	606.67	PDA	3.5
PDA35OfServ	Owned	299.99	PDA	3.5
PDA128OfSer	Owned	499.99	PDA	3.5
PDA8OfServic	Owned	399.99	PDA	3.5
PDA63OfServ	Owned	280.0	PDA	3.5
PDA118OfSer	Owned	633.68	PDA	3.8
PDA164OfSer	Owned	604.6	PDA	3.5
PDA77OfServ	Owned	499.0	PDA	3.7
PDA35OfServ	Owned	299.99	PDA	3.5
PDA16OfServ	Owned	599.0	PDA	3.5

64% of the offers fulfill your requirements.

Tradeoff opportunities ...

- display size?, display size unit?, price currency = usd (85%)
- computer manufacturer != Dell Computers, display size?, display size unit? (85%)
- computer manufacturer != Dell Computers, display size unit?, price currency = usd (8
- price currency = usd (100%)
- computer manufacturer != Dell Computers, price currency = usd (99%)
- none

Available attributes ...

- price currency (97%)
- OS (11%)
- color (12%)
- BATTERY (24%)
- COMPUTER MANUFACTURER (100%)
- COMPUTER MODEL (89%)
- RAM (22%)
- SIZE (77%)
- MODEM (63%)
- PROCESSOR (65%)
- WEIGHT (64%)
- display size unit (35%)

Figure 5.12.: Results view

either the instances that are members of the column's corresponding set, as specified in the depicted offer's description, or the type of the set, if its instances have not been restricted. To facilitate decision making, columns can be sorted to ease the comparison of available alternatives with respect to the considered attribute conditions and can be collapsed to enable users to hide unimportant information or temporarily focus their attention to a subset of the presented alternatives' properties (Requirement U.16). The latter enables decision makers to screen large numbers of available alternatives, which otherwise would impose too much mental effort [Kee92].

5.7.2. Enabling Requirements Specification Based on Presented Service Alternatives and their Characteristics

Besides viewing matching service offers, the user may indicate desirable service characteristics based on the presented alternatives (Requirement U.7). The system supports three ways of doing this: (1) by adding a not yet specified attribute condition to the request model, (2) by refining, i.e. subtyping, an attribute condition's target set type and (3) by critiquing one of the listed service offers. To implement the first two interaction opportunities, the system provides the user with a list of potential attribute conditions, that have not yet been included into the request model, but might be added to it, as well as with a list of subtypes that can replace existing attribute conditions' target set types (Figure 5.12 right). Potential attribute conditions and subtypes are retrieved from the set of matching service offers. The suggested attribute conditions are restricted to those that can be directly added to one of the instance sets that are already part of the request model. They are displayed sorted by their ability to reduce the system's uncertainty about the service consumer's service requirements (cf. Section 5.8). Next to each recommended attribute condition or subtype a value in brackets indicates the percentage of offers that specify the considered attribute condition/subtype. This is to encourage the user to think thoroughly about choices that are highly restrictive. As soon as the user selects an attribute condition or a type, the request model is updated accordingly. Matching offers are retrieved and presented to the user.

In addition to these interaction opportunities, the user may select a service offer from the presented list, that fits reasonably well to his requirements. He may then indicate desirable service properties relative to this offer by critiquing it. For example, the user might indicate that the offer is fine, but too expensive (see Figure 5.12 left). This can be done by simply clicking on the referenced attribute value. Based on the indicated property and the properties of the available service alternatives that fulfill this requirement, the system produces a list of tradeoff alternatives on other service aspects that the user has to accept when insisting on the indicated requirement. For example the system might indicate that the user has to accept that he cannot get a PDA from Dell when critiquing on the price of a computer offer (Figure 5.12 right). The system also indicates the percentage of offers that fulfill an applied critique as well as for each tradeoff alternative, the percentage of those offers that require the user to accept the considered tradeoff. Different tradeoff types are color-coded to make the presented tradeoff opportunities more perceivable. After having viewed the presented alternatives, the user can decide to abandon his requirement, to specify an additional requirement on the same service offer or he indicates that he is willing to accept one of the presented tradeoff alternatives by simply clicking it. While the second option will lead to another set of tradeoff alternatives that are produced by taking both requirements of the user into account, the third option will result in a model update reflecting the information provided by the user (cf. Section 5.9). In this context, tradeoffs do not necessarily refer to service aspects that have been already considered in the request model, but may also refer to service attributes that have not yet been specified by the user. The presented feature encourages the user to make compromises where necessary and helps him to identify yet unconsidered, but important service aspects (Requirements U.6 and U.12). The former is particularly important, since it challenges the user to reconsider attribute constraints. According to [Kee92], this fosters the identification of promising service alternatives, that otherwise would not have been considered. In the remainder of this section, we will provide details on the features of the introduced critiquing mechanism and on its implementation.

Determining Tradeoff Alternatives

The proposed critiquing and tradeoff mechanism extends previous work in the area of recommender systems [RMMS04, CP07b] and is implemented as follows. Assume the user selected a service offer o_{crit} for critiquing which matches to the request model r that is maintained for the user and fits reasonably well to his actual service requirements. Let a be an attribute of this offer and let *offValue* be the value of the offer taken with respect to that attribute. The types of critiques which can be applied to that attribute depend on the type of the attribute. If the attribute a is of a numerical type, such as Integer or Double, the user might provide either a critique of type *more than offValue*, indicating that he is only willing to accept a-values that are larger than *offValue*, a critique of type *less than offValue*, indicating that he is only willing to accept a-values that are lower than *offValue*, or no critique at all. If a is a nominal attribute such as Color, a critique of type *!=offValue*, meaning that the user does not accept the value *offValue* for a, or no critique at all can be specified. Based on the input critique C, the system proceeds as follows to identify tradeoff opportunities. In a first step, a copy r' of r is created, where the type of the visible (in the graphical request model representation introduced in Section 5.5) instance sets that are referenced in r is replaced by that of the corresponding sets of the critiqued offer o_{crit}[10]. Subsequently, the service offers O_M that match to r' are retrieved by using the matchmaking mechanism described in Section 5.6. Since by choosing the offer o_{crit} for critiquing, the user indicated that he is interested in offers of the selected type[11], $O_M \setminus o_{crit}$ constitutes the set of service offers to which alternative tradeoff opportunities refer to and from which they should be extracted. Thereby, a tradeoff opportunity indicates a set of compromises (relative to the critiqued offer o_{crit}) on the offers O_C that fulfill the provided critique C, which have to be accepted by the user, if insisting on the provided critique. Several alternative tradeoff patterns each applying to a fraction of the offers in O_C might exist. Since in the worst case, an exponential (in the number of different attributes that occur in the matching offers) number of tradeoff patterns exists, we are interested in those that are shared by large subsets of

[10]These are valid subtypes of the corresponding types in r, since otherwise o_{crit} would not have matched to r.

[11]This is due to the fact, that we do not maintain uncertainty about the type of service attributes.

the offers in O_C. As suggested in [RMMS04], those frequent tradeoff patterns can be extracted from O_M using the Apriori-algorithm [AS94] for *frequent itemset* and *association rule* mining, which is typically used to analyze the shopping behavior of customers based on recorded purchasing transactions. Thereby, frequent itemset mining is used to find sets of items which are frequently bought by customers. Based on those sets, subsequent association rule mining allows to identify items which are usually bought together, e.g. the association rule $\mathsf{beef\ roulade} \Rightarrow \mathsf{dill\ pickle}, \mathsf{bacon}$ states that customers who purchased beef roulade often also bought dill pickles and bacon. Thereby, the frequency with which an itemset X occurs in a certain set of transactions T is called the *support* of X in T.

Definition 5.8. (**Support and frequent itemset**) *The support supp(X) of an itemset X in a set of transactions T is the percentage of transactions that contain X. A frequent itemset is an itemset whose support exceeds a certain user-defined threshold. The support of an association rule $X \Rightarrow Y$ in T is given by supp($X \cup Y$), i.e. provides the percentage of transactions that comply with that rule.*

To identify recurring tradeoff patterns using the Apriori-algorithm, each offer $o \in O_M$'s attribute values are compared to those of the critiqued offer o_{crit}. As a result, a vector comprising of the tradeoffs relative to o_{crit} that apply to each of o's individual attributes is determined. Valid attribute-specific tradeoffs are *less than offValue* ($\downarrow$) and *more than offValue* ($\uparrow$) for numerical attributes and *!=offValue* for nominal attributes. Moreover, *?* indicates that o did not specify the considered attribute or its value. For instance, the vector $\langle \mathsf{price} \downarrow, \mathsf{quality} \downarrow \rangle$ refers to offers described by the two attributes price and $\mathsf{quality}$ and indicates that compared to the critiqued offer, the considered offer refers to a service that is less expensive, but also of lower quality. Each tradeoff vector is interpreted as a purchasing transaction and each of the single attribute-specific tradeoffs as an item. Taking the tradeoff vectors P of the offers in O_M as an input, the Apriori-algorithm delivers those tradeoff patterns that occur frequently in the provided vectors and thus in the offers O_M. Based on that patterns, applying association rules can be derived. Rules of the form $C \Rightarrow$ *Tradeoff* indicate tradeoff patterns *Tradeoff* that frequently occur in those offers that fulfill the user-provided critique C, i.e. the consequents of those rules provide the searched for tradeoff alternatives. Depending on the structure of the available offers and the type of critiques that have been applied, a large number of rules and thus a large number of tradeoff patterns is generated. Presenting all those patterns is ineligible, since this would overstrain the user. Hence, selecting a subset of the determined tradeoff alternatives for presentation is required. This should happen with particular care, since only a small fraction of the potentially large set of generated alternatives can be finally selected. Two different selection strategies have been suggested in the literature. While *Reilly et al.* [RMMS04] used the support of a rule as a measure for its relevance, *Chen et al.* [PC07] used support-based prefiltering combined with a selection on the basis of a measure that preferred those tradeoff alternatives that were promising in light of the user's stated requirements and diverse in terms of the involved attributes and the offers they apply to. However, they did not consider the fact that particularly at the beginning of the requirements elicitation process, the uncertainty about the validity of the maintained requirements model is high and thus the selected alternatives might be inappropriate. This is even likely, since the initial model suggested by the authors is based on default values. As a result, both strategies are limited in their ability to select those tradeoff alternatives for presentation that are most promising in terms of the user's requirements and in terms of the knowledge gain they provide, if selected (Requirement U.11). In this thesis, we therefore suggest an alternative approach to tradeoff selection that does not have that drawback. In addition, we will show that the number of generated association rules can be reduced before the actual selection process without loosing potential tradeoff candidates. This is due to the fact, that Apriori can produce frequent itemsets and thus association rules that are redundant. To see this, assume that for a given critique C all offers that require the user to accept the set of tradeoffs Y_1 also require him to accept the set of tradeoffs Y_2, then among others, Apriori will produce the two rules $C \Rightarrow Y_1$ and $C \Rightarrow Y_1 \cup Y_2$. However, the rule $C \Rightarrow Y_1$ is redundant and can be removed, since both rules apply to the same set of offers and $C \Rightarrow Y_1$ is implied by $C \Rightarrow Y_1 \cup Y_2$. Moreover, $C \Rightarrow Y_1$ incompletely describes the commonalities of the offers it applies to and thus Y_1 should not be presented as a tradeoff opportunity to the user. These considerations lead to the concept of frequent closed itemsets [PBTL99].

Definition 5.9. (**Closed itemset**) *An itemset X is closed with respect to a set of transactions T, if none of its supersets has the same support as X in T.*

It can be easily verified, that basing association rule mining on the subset of frequent itemsets that are closed, will result in a set of association rules that is equivalent to the set of frequent rules, but does not contain redundant rules of the type that has been described above. As demonstrated in [PBTL99], the number of non-closed rules is particularly high for correlated and dense data as can be expected in a service selection scenario. This results not only in a massive reduction of rules, but also in a lower time complexity of their computation. As shown by *Uno et al.* [UKA04], all closed frequent itemsets for a given data set can be computed in a time that is linear in the number of frequent closed itemsets. In our approach, we therefore use Uno et al.'s algorithm[12] for frequent closed itemset and association rule mining to determine all frequent closed association rules applying to the input patterns given by the tradeoff vectors P of the offers in $O_M \setminus o_{crit}$. We restrict ourselves to rules whose support exceeds a certain threshold (10% in our implementation). This step is done once, if the user decides to critique a certain offer and is not required to be repeated if another attribute-related critique with regard to the same offer is added. Once association rules have been mined, we search for those rules that have the critique C specified by the user as an antecedent. Those rules' consequents indicate sets of tradeoffs that often occur in the offers that fulfill the user's critique. However, not all tradeoffs given by the filtered rule's consequents are actually implied by the user-specified critique, but occur independently from it. Those tradeoff alternatives should not be presented to the user. Hence, we have to find a means to identify those rules $C \Rightarrow Y$ that indicate true implications. A commonly used measure for the quality or interestingness of a rule is its *confidence*.

[12]The implementation provided by the authors can be found at http://fimi.ua.ac.be/src/fimi03b.tgz.

Definition 5.10. (**Confidence**) *The confidence conf($X \Rightarrow Y$) of a rule $X \Rightarrow Y$ in a set of transactions T is the probability of that rule being correct, i.e. the probability of observing Y in a given transaction $t \in T$ that also contains X. That is*

$$conf(X \Rightarrow Y) = Prob(Y|X) = \frac{Prob(X \wedge Y)}{Prob(X)} = \frac{supp(X \Rightarrow Y)}{supp(X)}.$$

Is the confidence measure suitable to identify those rules that indicate implications? Unfortunately, the answer is no. This is due to the fact, that rules having a consequent with high support, i.e. a consequent that is frequent, will be assigned a high confidence, though the occurrence of the items given by the rule's antecedent might be completely unrelated to that of those in the consequent. To account for that issue, *Brin et al.* [BMUT97] introduced *conviction* as a measure indicating to what degree a given association rule really states an implication.

Definition 5.11. (**Conviction**) *The conviction conv($X \Rightarrow Y$) of a rule $X \Rightarrow Y$ in a set of transactions T is the ratio of the probability of that rule being incorrect, i.e. the probability of not observing Y in a given transaction $t \in T$ that contains X, if X and $\neg Y$ were independent, to the observed probability of that rule being incorrect. That is*

$$conv(X \Rightarrow Y) = \frac{Prob(X)Prob(\neg Y)}{Prob(X)Prob(\neg Y|X)} = \frac{Prob(X)(1-Prob(Y))}{Prob(X)(1-Prob(Y|X)} = \frac{1-supp(Y)}{1-conf(X \Rightarrow Y)}.$$

As can be seen, the measure takes a value of 1, if X and Y are independent. Values greater than 1 indicate that incorrect predictions of that rule occur less often than this would be the case if X and Y were independent, i.e. suggest that X indeed implies Y. Conviction is maximal for perfect implications.

Using conviction as a measure, we determine those rules whose conviction value exceeds a certain threshold (2 in our implementation[13]) and rank the tradeoff opportunities they encode. Thereby, a higher conviction of the corresponding rule indicates a higher rank. We also rank the tradeoff opportunities given by the consequents of the identified rules with respect to their ability to reduce the system's uncertainty about the user's request model (cf. Section 5.8). The greater the ability to reduce uncertainty, the higher the rank. Finally, we sort the determined tradeoff alternatives by their mean rank and present the k best (10 in our implementation) to the user. Thereby, the attribute-specific tradeoffs included by each tradeoff opportunity are listed. They are described by the name of the attribute they refer to and by the kind of tradeoff that has to be accepted, e.g. the attribute-specific tradeoff *quality* $\downarrow$ indicates that a lower quality has to be accepted. If the attribute that is referred to in the tradeoff is not yet considered in the request model, its value/its type is shown next to the tradeoff type (cf. Figure 5.12). Again, a value in brackets, that is displayed next to each tradeoff alternative, indicates the percentage of offers that will match to the user's requirements, if the considered interaction opportunity is selected. The overall procedure to identify promising tradeoff opportunities is summarized in Listing 5.1.

[13]This indicates that a given rule $X \Rightarrow Y$ would be incorrect 100% more often, if X and Y were independent. The threshold is incrementally decreased, if less than 10 rules are found.

```
Tradeoffs generateTradeoffs(Critique $C$, Offer $o_{crit}$, RequestModel $r$) {
    if($o_{crit}$ has not been critiqued before) {
        - get a copy $r'$ of $r$, where the type of the visible instance sets
            that are referenced in $r$ is changed to that of the corresponding
             sets of the critiqued offer $o_{crit}$
        - retrieve the set $O_M$ of service offers matching to $r'$
        - remove the critiqued offer $o_{crit}$ from $O_M$
        - determine the set $P$ comprising of each matching service offer
            $o \in O_M$'s tradeoff pattern with respect to $o_{crit}$
        - determine the set $F$ of frequent closed itemsets from $P$ that
            exceed a certain minimal support
        - determine the set $R$ of association rules from $F$
    }
    - filter the rules $R$ according to the user's critiquing wishes $C$
    - rank the tradeoff opportunities given by the rules in $R$ (whose
        conviction exceeds a certain threshold) by their rules' conviction-
        values
    - rank the tradeoff opportunities by their ability to reduce the system
        's uncertainty about the user's request model
    - return the tradeoff $k$ opportunities that are best w.r.t. their mean
        rank
}
```

Listing 5.1: Determining tradeoff opportunities based on a user's current request model r and the critiques C that he applied to a matching service offer o_{crit}

5.8 Determining Promising Interaction Opportunities

Effectively reducing uncertainty about the consumer's service requirements, which results from incomplete and inaccurate knowledge, is key to the performance of the requirements elicitation process. To achieve this, we propose to direct and focus the process of requirements refinement by emphasizing those interaction opportunities that have a high potential to increase the system's knowledge, i.e. to reduce its uncertainty, about the consumer's service requirements. Thereby, knowledge acquisition concentrates on those aspects of the user's requirements that are relevant in light of the available service options and in light of the user's known requirements (Requirement U.11). Consider for example a flight booking scenario. If all available services offer food during the flight, then there is no need to know whether the user would also accept flight offers without this service. As well, if price is not relevant to a consumer's service selection decision, then to explore in detail which prices are more desirable for this user is pointless. Necessary prerequisites for the outlined approach are a requirements model that explicitly represents and locates uncertainty about a consumer's service requirements (Requirement U.3) as well as a measure that quantifies that uncertainty. While the required model was introduced in Section 5.4, this section is dedicated to the presentation of an uncertainty measure that covers the described notion of uncertainty (Section 5.8.1). Moreover, a procedure that leverages the proposed measure in order to identify promising interaction opportunities is presented (Section 5.8.2).

5.8.1. Measuring Model Uncertainty

The goal of the requirements elicitation process is to sort available service offers according to their suitability with respect to the consumer's service requirements. The latter is indicated by an offered instance's matching degree, i.e. the user's overall preference for this instance. In the remainder of this section, we will propose a measure that quantifies the system's uncertainty about the user's preferences for the available offers based on a given request model and thus indirectly and inversely measures the quality of the offer ranking produced by the system. As such, it qualifies as a means to identify promising interaction opportunities, that, if taken, lead to more accurate request models, thus decrease uncertainty associated with the computed matching degrees and finally result in more accurate service offer rankings.

The request model introduced in Section 5.4 represents uncertainty about a service consumer's requirements by means of probability distributions over potential and mutually exclusive consumer requirements, which concern the importance of desired service properties (encoded by uncertain connecting strategies) as well as direct constraints on the values of the latter (encoded by uncertain direct conditions). Uncertainty about the true matching degree of an offer, i.e. the true overall preference for it, is the higher the more plausible and mutually exclusive requirements that affect it exist, i.e. the more alternative service properties and direct constraints might be valid with a non-negligible probability. An uncertainty measure that covers this notion of uncertainty and therefore is a suitable choice for our purposes is the Shannon entropy [Sha48] $S(X)$ of a random variable X and its associated probability distribution[14]. It is given by

$$S(X) = -\sum_{j=1}^{n} \mathit{Prob}(x_j) \cdot \log_2(\mathit{Prob}(x_j)), \tag{5.23}$$

where $x_1, \ldots, x_n$ are the alternative values of X. The value of the entropy is zero, meaning no uncertainty, if there is exactly one alternative $X = x$ with probability $\mathit{Prob}(x) = 1$ and is maximal, meaning maximal uncertainty, for equally likely alternatives. Using Shannon entropy to measure the system's uncertainty about the validity of potential direct conditions and connecting strategies as covered in a given request model, we are able to derive a measure indicating the system's uncertainty about the overall preference for a service offer with respect to a given request model. Since the maximal entropy may differ for each of the involved random variables, we utilize Shannon's entropy measure normalized to the interval $[0, 1]$. Normalization is done by dividing the measure $S(X)$ by the maximal

[14]Shannon entropy is only applicable to random variables taking discrete values. Continuous extensions to the measure have been suggested in the literature (see e.g. [AK06]), but are not discussed in this thesis. This is due to the fact that all probability distributions that are associated with request model elements and that are considered in this thesis are either discrete by nature or are maintained in discretized form.

entropy that can be achieved with respect to X, i.e. the entropy that results, if all potential values of X are equally likely. The normalized measure S_N is given by

$$\begin{aligned} S_N(X) &= \frac{S(X)}{-\sum_{j=1}^{n} \frac{1}{n} \cdot \log_2(\frac{1}{n})} \\ &= \frac{S(X)}{\log_2 n}. \end{aligned} \tag{5.24}$$

The resulting uncertainty values range between 0 and 1, where a higher value indicates higher uncertainty.

Uncertainty about the Preference with Respect to the Direct Conditions Using Shannon entropy as an uncertainty measure, the uncertainty $U(\mathit{Pref}_I^{dc_j}(i)) \in [0,1]$ about the preference $\mathit{Pref}_I^{dc_j}(i)$ for an instance i with respect to the user's in-conditions on the instance set I as covered by the uncertain in-condition dc_j is given by

$$\begin{aligned} U(\mathit{Pref}_I^{dc_j}(i)) = S_N(\mathit{Pref}_I^{dc_j}(i)) \\ = -\ \ &(p_{\mathit{In}_i^I}(true) \cdot \log_2(p_{\mathit{In}_i^I}(true)) + \\ &(1 - p_{\mathit{In}_i^I}(true)) \cdot \log_2(1 - p_{\mathit{In}_i^I}(true))). \end{aligned} \tag{5.25}$$

Similarly, the uncertainty $U(\mathit{Pref}_I^{dc_j}(i)) \in [0,1]$ about the preference $\mathit{Pref}_I^{dc_j}(i)$ for an instance i with respect to the user's not-in-conditions on the instance set I as covered by the uncertain not-in-condition dc_j is given by

$$\begin{aligned} U(\mathit{Pref}_I^{dc_j}(i)) = S_N(\mathit{Pref}_I^{dc_j}(i)) \\ = -\ \ &(p_{\mathit{NotIn}_i^I}(true) \cdot \log_2(p_{\mathit{NotIn}_i^I}(true)) + \\ &(1 - p_{\mathit{NotIn}_i^I}(true)) \cdot \log_2(1 - p_{\mathit{NotIn}_i^I}(true))). \end{aligned} \tag{5.26}$$

Finally, the uncertainty $U(\mathit{Pref}_I^{dc_j}(i)) \in [0,1]$ about the preference $\mathit{Pref}_I^{dc_j}(i)$ for an instance i with respect to the user's range condition on the instance set I as covered by the uncertain range condition dc_j[15] is given by

$$\begin{aligned} U(\mathit{Pref}_I^{dc_j}(i)) = S_N(\mathit{Pref}_I^{dc_j}(i)) \\ = -\ \ &(\mathit{Prob}(\mathit{Min}_I \leq i < \mathit{Max}_I) \cdot \log_2(\mathit{Prob}(\mathit{Min}_I \leq i < \mathit{Max}_I)) + \\ &(1 - \mathit{Prob}(\mathit{Min}_I \leq i < \mathit{Max}_I)) \cdot \log_2(1 - \mathit{Prob}(\mathit{Min}_I \leq i < \mathit{Max}_I))) \\ = -\ \ &(\mathbb{E}[\mathit{Pref}_I^{dc_j}(i)] \cdot \log_2(\mathbb{E}[\mathit{Pref}_I^{dc_j}(i)]) + \\ &(1 - \mathbb{E}[\mathit{Pref}_I^{dc_j}(i)]) \cdot \log_2(1 - \mathbb{E}[\mathit{Pref}_I^{dc_j}(i)])), \end{aligned} \tag{5.27}$$

[15] While uncertainty about the validity of all potential (not-)in-conditions a user might have is covered by a single uncertain (not-)in-condition, an uncertain range condition just maintains uncertainty about the validity of all potential single range conditions. A (single) uncertain range condition is not capable of managing uncertainty about the validity of several concurrent range conditions a service requester might have.

where the third equation results from Formula 5.22 in Section 5.6.

According to Definition 5.4, the preference $Pref_I^{dc}(i)$ for an instance i with respect to all uncertain direct conditions $dc_j, 1 \leq j \leq n$, that have been specified for the instance set I is given by the product of the preference values $Pref_I^{dc_j}(i), 1 \leq j \leq n$, resulting from the single uncertain direct conditions. Uncertainty about the value of a product is low, if at least one of its factors is known to be low or if all factors are known. The subsequent fuzzy logic expression adequately describes that fact and therefore is a reasonable definition for the system's uncertainty U($Pref_I^{dc}(i)$) about the preference $Pref_I^{dc}(i)$. It is given by

$$\begin{aligned} U(Pref_I^{dc}(i)) = 1-(& \\ & \bigoplus_{j=1}^{n}(1 - \mathbb{E}[Pref_I^{dc_j}(i)]) \cdot (1 - U(Pref_I^{dc_j}(i))) \oplus \\ & \prod_{j=1}^{n}(1 - U(Pref_I^{dc_j}(i))) \\ &), \end{aligned} \tag{5.28}$$

where $a \oplus b = a + b - a \cdot b$ is the fuzzy *or*-connective and $\cdot$ is the fuzzy *and*-connective. The formula indicates that the uncertainty U($Pref_I^{dc}(i)$) about the value $Pref_I^{dc}(i)$ is low, if at least one of the preference values $\{Pref_I^{dc_j}(i)|, 1 \leq j \leq n\}$ is known to be low, i.e. if there exists a $j, 1 \leq j \leq n$, with $\mathbb{E}[Pref_I^{dc_j}(i)] \to 0$ *and* $U(Pref_I^{dc_j}(i)) \to 0$ (first n terms of Formula 5.28), *or* if all preference values $\{Pref_I^{dc_j}(i)|1 \leq j \leq n\}$ are known, i.e. if $U(Pref_I^{dc_j}(i)) \to 0$ holds for all $j, 1 \leq j \leq n$, (last term of Formula 5.28). In particular, the definition implies that acquiring information about other uncertain direct conditions is not required, if for a given instance, the user's preference with respect to one of the uncertain direct conditions is known to be low. This is in compliance with Requirement U.11. Since preference values take values from the interval $[0, 1]$, the resulting uncertainty values range between 0 and 1, where a higher value indicates higher uncertainty. Finally, we define the uncertainty U($Pref_I^{dc}(i)$) to be 1, if no uncertain direct conditions are maintained for I. This is consistent with the fact, that in such a case, no information about potential direct conditions of the user have been acquired yet.

Uncertainty about the Overall Preference with Respect to an Instance Set

Let I be an instance set referenced in a request model. According to Definition 5.7, the overall preference $Pref_I^{ov}(i)$ for an instance i with respect to I is given by the product of the preferences $Pref_I^{tc}(i)$, $Pref_I^{dc}(i)$ and $Pref_I^{ac}(i)$ resulting from the user's type condition, his direct conditions and the attribute conditions associated with I. Following the above argumentation about the uncertainty of a product of values, the system's uncertainty

$U(\mathit{Pref}_I^{ov}(i)) \in [0,1]$ about the user's overall preference for a given instance i with respect to I can be adequately and recursively described by the expression

$$\begin{aligned} U(\mathit{Pref}_I^{ov}(i)) = 1-(& \\ & (1 - \mathit{Pref}_I^{tc}(i)) \oplus \\ & (1 - \mathbb{E}[\mathit{Pref}_I^{dc}(i)]) \cdot (1 - U(\mathit{Pref}_I^{dc}(i))) \oplus \\ & (1 - \mathbb{E}[\mathit{Pref}_I^{ac}(i)]) \cdot (1 - U(\mathit{Pref}_I^{ac}(i))) \oplus \\ & (1 - U(\mathit{Pref}_I^{ac}(i))) \cdot (1 - U(\mathit{Pref}_I^{dc}(i))) \\ &), \end{aligned} \tag{5.29}$$

where the user's uncertainty U($\mathit{Pref}_I^{ac}(i)$) with respect to his attribute conditions on the instance set I is the recursive component and is subsequently defined. The formula indicates that the uncertainty about the product $U(\mathit{Pref}_I^{ov}(i))$ is low, if at least one of its contributing factors $\mathit{Pref}_I^{tc}(i)$, $\mathit{Pref}_I^{dc}(i)$ or $\mathit{Pref}_I^{ac}(i)$ is known to be low, i.e. if $\mathit{Pref}_I^{tc}(i) \rightarrow 0$, $\mathbb{E}[\mathit{Pref}_I^{dc}(i)] \rightarrow 0$ *and* $U(\mathit{Pref}_I^{dc}(i)) \rightarrow 0$ *or* $\mathbb{E}[\mathit{Pref}_I^{ac}(i)] \rightarrow 0$ *and* $U(\mathit{Pref}_I^{ac}(i)) \rightarrow 0$[16] (first three terms of Formula 5.29), *or* if all the values $\mathit{Pref}_I^{tc}(i)$, $\mathit{Pref}_I^{dc}(i)$ *and* $\mathit{Pref}_I^{ac}(i)$ are known, i.e. if $U(\mathit{Pref}_I^{dc}(i)) \rightarrow 0$ *and* $U(\mathit{Pref}_I^{ac}(i)) \rightarrow 0$ (4th term in Formula 5.29). The definition implies that, if a user's preference for an instance i with respect to its type, its value or its attributes related to a given instance set is known to be low, acquiring information about other requirements he might have with respect to that instance and that instance set is not required. This is again in compliance with Requirement U.11.

Let I' be the corresponding instance set of I in i and let $\{p_{W_{ac_j}} | 1 \leq j \leq n\}$ be the weight distributions maintained for the attribute conditions $ac_j, 1 \leq j \leq n$, specified either for I or I'. Thereby, the distribution $p_{W_{ac_j}}$ is defined to be a uniform distribution over possible weights (encoding missing knowledge), if the attribute condition ac_j has been specified in I', but not in the corresponding set I of the request model. The preference $\mathit{Pref}_I^{ac}(i)$ for the instance i with respect to its attributes that have been specified for I' is given by the weighted sum of the preferences $\{\mathit{Pref}_I^{ac_j}(i) | 1 \leq j \leq n\}$ for the single attributes that have been specified for I or I', normalized to the interval $[0,1]$ (Definition 5.6). The value of a (normalized) weighted sum is known, if the single addends are known and do not differ much (in this case the weights have only a marginal impact on the value of the sum and thus are not needed to be known) or if both, the single weighted addends and their weights, are known. Hence, the uncertainty U($\mathit{Pref}_I^{ac}(i)$) about the preference value $\mathit{Pref}_I^{ac}(i)$ is appropriately described by the logic expression

$$\begin{aligned} U(\mathit{Pref}_I^{ac}(i)) = 1-(& \\ & \prod_{j=1}^{n} (1 - U(\mathit{Pref}_I^{ac_j}(i))) \cdot (1 - STDEV_k(\mathbb{E}[\mathit{Pref}_I^{ac_k}(i)])) \oplus \\ & \prod_{j=1}^{n} (1 - U(S_I^j(i))) \cdot \prod_{j=1}^{n} (1 - U(W_{ac_j})) \end{aligned}$$

[16]As already mentioned, we do not maintain uncertainty about the type of acceptable service instances (cf. Section 5.4.1), thus $\mathit{Pref}_I^{tc}(i)$ is always known.

$$), \tag{5.30}$$

where $S_I^j(i) := W_{ac_j} \cdot Pref_I^{ac_j}(i)$, $\mathbb{E}[S_I^j(i)]$ is its corresponding expected value and $STDEV_k(\mathbb{E}[Pref_I^{ac_k}(i)])$ is the standard deviation of the single expected preferences $\{Pref_I^{ac_k}(i)|1 \leq k \leq n\}$. The uncertainty U($Pref_I^{ac}(i)$) is defined to be 0, if no attribute conditions have been specified for I' and I, since in that case, knowledge about the user's preferences for potential attributes is not required. As can be seen, Formula 5.30 expresses that the uncertainty about the value $Pref_I^{ac}(i)$ is low, if the single preferences $\{Pref_I^{ac_j}(i)|1 \leq j \leq n\}$ are known *and* do not differ much, i.e. $STDEV_k(\mathbb{E}[Pref_I^{ac_k}(i)]) \rightarrow 0$ *and* $U(Pref_I^{ac_j}(i)) \rightarrow 0$ holds for all $ac_j, 1 \leq j \leq n$ (first term of Formula 5.30), *or* if the single contributions $\{S_I^j(i)|1 \leq j \leq n\}$ to the sum as well as their weights $\{W_{ac_j}|1 \leq j \leq n\}$ are known, i.e. $U(S_I^j(i)) \rightarrow 0$ *and* $U(W_{ac_j}) \rightarrow 0$ holds for all $j, 1 \leq j \leq n$ (second term of Formula 5.30). Thereby, the uncertainty U($S_I^j(i)$) about the value of the product $W_{ac_j} \cdot Pref_I^{ac_j}(i)$ is given by the expression

$$\begin{aligned} U(S_I^j(i)) = 1-(& \\ & (1 - \mathbb{E}[W_{ac_j}]) \cdot (1 - U(W_{ac_j})) \oplus \\ & (1 - \mathbb{E}[Pref_I^{ac_j}(i)]) \cdot (1 - U(Pref_I^{ac_j}(i))) \oplus \\ & (1 - U(W_{ac_j})) \cdot (1 - U(Pref_I^{ac_j}(i))) \\ &). \end{aligned} \tag{5.31}$$

According to that formula, the uncertainty U($S_I^j(i)$) is low, if the weight W_{ac_j} of the considered attribute condition is known to be low, i.e. if $\mathbb{E}[W_{ac_j}] \rightarrow 0$ *and* $U(W_{ac_j}) \rightarrow 0$ (first term of Formula 5.31), the preference $Pref_I^{ac_j}(i)$ with respect to the latter is known to be low, i.e. $\mathbb{E}[Pref_I^{ac_j}(i)] \rightarrow 0$ *and* $U(Pref_I^{ac_j}(i)) \rightarrow 0$ (second term of Formula 5.31), *or* both of those values are known, i.e. $U(W_{ac_j}) \rightarrow 0$ and $U(Pref_I^{ac_j}(i)) \rightarrow 0$ (third term of Formula 5.31). The uncertainty U(W_{ac_j}) about the weight W_{ac_j} of an attribute condition ac_j is given by its normalized Shannon entropy $S_N(W_{ac_j})$, i.e.

$$\begin{aligned} U(W_{ac_j}) &= S_N(W_{ac_j}) \\ &= -\frac{\sum_{k=1}^{n} Prob(w_k) \cdot \log_2(Prob(w_k))}{\log_2 n}, \end{aligned} \tag{5.32}$$

where the number and selection of the discrete values $\{w_k|1 \leq k \leq n\}$ for which $Prob(w_k)$ is maintained depends on the discretization of $p_{W_{ac_j}}$. Finally, the uncertainty U($Pref_I^{ac_j}(i)$) about the preference $Pref_I^{ac_j}(i)$ for the instance i with respect to the specified attribute condition ac_j is defined to be

$$U(Pref_I^{ac_j}(i)) = \begin{cases} 0.0 & \text{if } ac_j \text{ is not specified in } i \\ 1.0 & \text{if } ac_j \text{ is not specified for } I \\ U(Pref_{I_j}^{ov}(i)) & \text{otherwise,} \end{cases} \tag{5.33}$$

where $Pref_{I_j}^{ov}(i)$ is the (overall) preference value of the instance i with respect to the target set I_j of attribute condition ac_j in the request model. This definition is reasonable, since knowledge about a user's preference with respect to a certain attribute is not required, if that attribute is not specified in the considered service instance (line one of Definition 5.33). Moreover, uncertainty about the user's preference with respect to an attribute that has been specified in the considered service instance, is maximal, if nothing is known about that preference (line two of Definition 5.33)[17]. In compliance with Requirement U.11, the Formulas 5.30 to 5.33 indicate that, if it is known that the user's preferences for the single attributes of a given service instance do not differ much, knowledge about these attributes' importance is not required. Besides, if the importance (the weight) of an attribute is known to be low, information about the user's requirements with respect to that attribute are not needed.

Given a request model, that represents the user's known service requirements, the system's uncertainty about the user's true (overall) preference for a given service instance i with respect to the given model is determined by recursively computing $\mathrm{U}(\mathit{Pref}^{ov}_{I_{\mathit{effect}}}(i))$ for the effect set I_{effect} of the model.

5.8.2. Selecting Promising Interaction Opportunities

Based on the proposed measure, we can determine those interaction opportunities, i.e. those subtypes, subattributes and trade-off alternatives, that, when selected, have the highest potential to reduce the system's uncertainty about the consumer's preferences for the offered services, and emphasize them to increase their visibility to the user. For that purpose, we determine for each of the interaction opportunities the request model that would result, if the user made use of this option. After that, we calculate the mean uncertainty about the k best offer's matching degrees (cf. Section 5.6) with respect to the resulting request model. Finally, the interaction opportunities are sorted by their resulting mean uncertainty and are displayed in ascending order[18]. This step is done separately for each type of interaction opportunity. Determining the uncertainty about a user's service requirements and thus about his preference for potential service offers exclusively with respect to existing and available offers is in compliance with Requirement U.11. By emphasizing promising interaction opportunities, but at the same time not concealing those that seem to be non-auspicious, the process of requirements elicitation as implemented in our approach is guided, but does not constrain the user's potential choices (Requirement U.12).

5.9 Model Update

As argued, user interactions with the system should lead to appropriate request model updates to maintain consistency with the user's evolving service requirements and preferences

[17]If a given attribute has not been considered in the request model, this might suggest that the attribute is not important to the user. However, we do not know that, so, assuming maximal uncertainty is reasonable.

[18]In some cases, additional sorting criteria are used. This is indicated and detailedly explained in the corresponding paragraphs of Section 5.7.

(Requirement U.9). However, so far we owed to explicate what kind of model changes are performed upon a given interaction and how they are accomplished. In this section, we make up for that and provide an overview about the update operations performed on the request model. The expositions are structured along the available interaction opportunities that have been detailedly described in the previous sections (most notably Sections 5.5 and 5.7). Those include explicit model interactions performed via the graphical representation (Section 5.5) of the request model (Section 5.9.1) as well as user actions that implicitly trigger model updates, but are not directly intended to change the model (Section 5.9.2). While the former typically refer to single model elements such as to an attribute condition or an attribute condition's weight, the latter typically affect several model elements. Table 5.2 provides an overview about the interaction opportunities offered by our system and indicates the probability distributions that they affect if performed.

5.9.1. Explicit Model Interactions

This section is concerned with user interactions explicitly aiming at changing the request model in a certain way. Those include attribute-condition-, direct-condition- and connecting-strategy-related interactions performed via the graphical representation of the request model (Section 5.5). Since the user's intention behind those interactions is known, it is typically clear how the model and in particular the probability distributions that are associated with it have to be updated to meet the user's goal.

Attribute-condition-related Interactions

Upon the specification of an attribute condition via the graphical model representation, an appropriate attribute condition is added to the request model. Name and target set type of this condition are chosen according to the corresponding attribute name and target set type as defined in the ontology. In addition to that, the connecting strategy of the set to which the newly created attribute condition ac_j refers to is appropriately adjusted by adding a probability distribution $p_{W_{ac_j}}$ maintaining the system's knowledge about the absolute weight $W_{ac_j} \in [0, 1]$ of the added condition. The distribution is initialized as a uniform distribution, since upon creation time, knowledge about the attribute condition's weight is not available and thus all potential weights are equally likely.

Instead of specifying a single attribute condition, a user might also apply the subtree recommender (Section 5.5.2) to provide a whole subtree of nested attribute conditions that shall be added to a given set referenced in the request model. The sets referred to in the subtree, that is provided by the recommendation tool, are already valid model elements, i.e. the nesting of the given attribute conditions is in compliance with the attribute and target set type constraints specified in the ontology. Moreover, each set provides a connecting strategy of type weighted sum that assigns equal weights to all attribute conditions specified for that set, i.e. for each condition's weight, a uniform probability distribution is maintained.

Interaction Opportunity	Affected Probability Distribution(s)
Explicit Model Interactions	
Attribute-condition-related Interactions	
add/remove attribute condition	weight distribution of the attribute condition
refine target set type	-
add/remove request model subtree	weight distributions of the affected attribute conditions
Direct-condition-related Interactions	
Range-condition-related Interactions	
add/(remove) range condition	minimum and maximum distribution of the range
adjust range	minimum or maximum distribution of the range
In-condition-related Interactions	
add/(remove) in-condition	-
add/remove instance to/from in-condition	In-distribution of the instance
increase/decrease preference for an in-condition-instance	In-distribution of the instance
Not-in-condition-related Interactions	
add/(remove) not-in-condition	-
add/remove instance to/from not-in-condition	NotIn-distribution of the instance
Connecting-strategy-related Interactions	
add connecting strategy (automatically)	-
increase/decrease attribute condition weight	weight distribution of the attribute condition
Implicit Model Interactions	
critique attributes of a service offer and choose compromise	weight distributions of the attribute conditions that refer to the critiqued and compromised attributes, direct-condition-related distributions that refer to the critiqued attributes

Table 5.2.: Available interaction opportunities

As discussed in Section 5.5.2, to none of the sets a direct condition is assigned. Hence, upon the provision of a request model subtree for addition, just the attribute conditions $\{ac_j | 1 \leq j \leq n\}$ specified for the subtree's root set have to be added to the corresponding set of the request model[19]. In addition, for each of those conditions a uniform probability distribution $p_{W_{ac_j}}$ for its weight W_{ac_j} is created. In case of a single attribute condition's or more generally, a request model subtree's removal, the concerned attribute conditions as well as their weight distributions are simply deleted.

Finally, a user might indicate his willingness to refine the type of a set that is already referenced in the considered request model (cf. Section 5.5). As a consequence, the type of the referenced set is set to the specified subtype (presuming that it is a valid subtype of the set's present type).

Direct-condition-related Interactions

As discussed in Section 5.4, the request model supports three kinds of uncertain direct conditions: uncertain range conditions, uncertain in-conditions as well as uncertain not-in-conditions. All types of those conditions can be specified and edited via the graphical model representation, that has been introduced in Section 5.5. Altering actions trigger different update operations on the model, which will be detailedly described in the remainder of this section.

Range-condition-related Interactions The creation of an uncertain range condition, indicating a range of acceptable instances of a set I, that is referenced in the request model, is either explicitly prompted by the user via the graphical model representation or implicitly triggered by a critiquing-and-compromise-operation. It involves the initialization of two probability distributions p_{Min_I} and p_{Max_I} that maintain knowledge about the minimum Min_I and the maximum Max_I of the range. Due to missing information about the actual minimum and maximum at creation time, the probability distributions are initialized as uniform distributions over the interval R_I of potential minima and maxima. The latter is extracted from available service offers by using knowledge services.

Information about the actual minimum and maximum of the range are acquired from relevant system interactions performed by the user. They are integrated into the corresponding distributions by using Bayesian inference. Thereby, the posterior probability distribution $p_{Min_I}(min_I | interaction)$, taking a certain interaction into account, is given by

$$p_{Min_I}(min_I | interaction) = c_{p_{Min_I}}^{interaction} \cdot L_{p_{Min_I}}(interaction | min_I) \cdot p_{Min_I}(min_I), \tag{5.34}$$

[19]The set of the present request model that shares the path (Definition 6.4) of the recommended subtree's root set in the contributing request model.

where $p_{Min_I}(min_I)$ is the prior probability distribution before considering the interaction and $c^{interaction}_{p_{Min_I}}$ is a normalizing constant ensuring that $p_{Min_I}(min_I|interaction)$ is in fact a probability distribution. The likelihood function $L_{p_{Min_I}}(interaction|min_I)$ indicates the likelihood of the user interaction given $Min_I = min_I$[20]. This analogously holds for the posterior probability distribution $p_{Max_I}(max_I|interaction)$.

The likelihood function differs depending on the considered distribution (either p_{Min_I} or p_{Max_I}) as well as depending on the type of interaction that has been performed by the user. Relevant actions, i.e. those providing information about the minimum or the maximum of an acceptable range of instances, include the user *having moved the minimum- or the maximum-slider in the corresponding range condition's graphical representation* and the user *having critiqued attributes of a service offer that has been recommended by the system*. Consider the former type of interaction. W.l.o.g., assume that the user moved the minimum-slider of a range condition that has been specified for the set I referenced in the request model under construction. Then, given $Min_I = min_I$, the likelihood $L_{p_{Min_I}}(\textit{moves min-slider to newInst}|min_I)$ of the user moving the minimum-slider of the considered range condition's graphical representation to a new instance *newInst* is determined by the likelihood $L_{p_{Min_I}}(\textit{moves min-slider}|min_I)$ of the user ever moving the slider and the likelihood $L_{p_{Min_I}}(\textit{moves it to newInst}|min_I)$ of moving it to *newInst*. In particular, $L_{p_{Min_I}}(\textit{moves min-slider to newInst}|min_I)$ is given by

$$\begin{aligned} &L_{p_{Min_I}}(\textit{moves min-slider to newInst}|min_I)\\ &= L_{p_{Min_I}}(\textit{moves min-slider} \ \wedge \ \textit{moves it to newInst}|min_I)\\ &= L_{p_{Min_I}}(\textit{moves min-slider}|min_I) \cdot L_{p_{Min_I}}(\textit{moves it to newInst}|min_I). \end{aligned} \tag{5.35}$$

Since both, $L_{p_{Min_I}}(\textit{moves min-slider}|min_I)$ and $L_{p_{Min_I}}(\textit{moves it to newInst}|min_I)$, are unknown, we have to estimate them appropriately. Following the intuition that users are more inclined to adjust the position of the minimum-slider, if the discrepancy between the range's actual minimum and the minimum that is displayed in the model visualization is large, rather than if it were low, we set

$$L_{p_{Min_I}}(\textit{moves min-slider}|min_I) = \frac{(l_{max} - l_{min}) \cdot |min_I - \textit{shownInst}|}{|R_I|} + l_{min}, \tag{5.36}$$

where *shownInst* is the instance indicated by the minimum-slider in the graphical representation, i.e. is the most probable minimum of the range (cf. Section 5.5), $l_{min} \geq 0$ is the likelihood of the user moving the minimum-slider, even if it is positioned correctly and $l_{max} \geq l_{min}$ is the likelihood of the user moving the minimum-slider, if it is positioned as far away as possible from the actual minimum. The likelihood function depends on the distance between the minimum instance indicated by the position of the minimum-slider and the actual minimum. It is depicted in Figure 5.13(a). Note, that any two of

[20] Note, that likelihood values do not indicate the probability of events, but are proportional to it. In particular, if related to each other, they provide the ratio of two event's probabilities.

those likelihood functions over the same range R_I, that differ just in the absolute values of l_{min} and l_{max}, but share the ratio $\frac{l_{max}}{l_{min}}$, are equivalent. In our implementation, we chose $l_{min} = 0.05$ and $l_{max} = 1$, independent from the type of the considered instance set I and the size of the range R_I. This implies that likelihood values increase faster over small than over large ranges R_I and thus models the fact that users are more sensitive to a misplaced minimum-slider, if the range of potential minima and maxima is smaller.

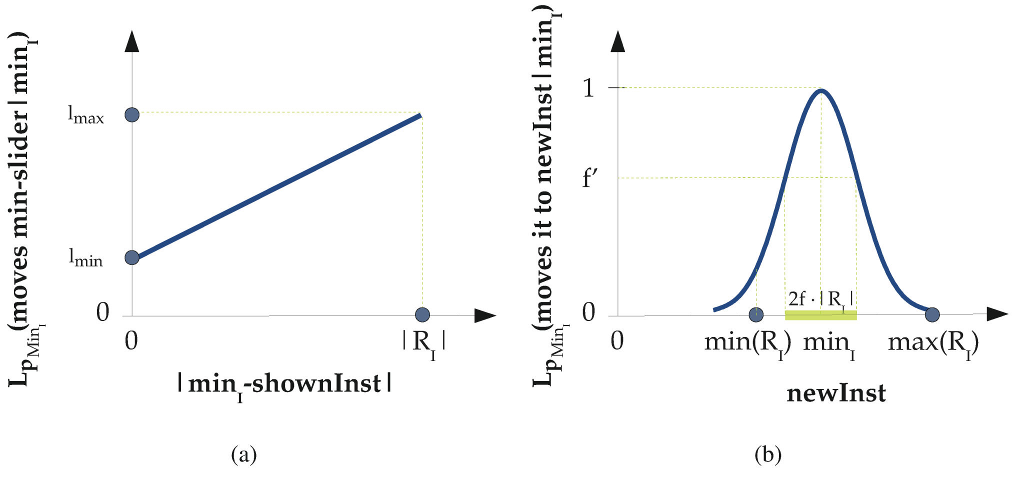


Figure 5.13.: Likelihood functions $L_{p_{Min_I}}(\textit{moves min-slider}|\textit{min}_I)$ (a) and $L_{p_{Min_I}}(\textit{moves it to newInst}|\textit{min}_I)$ (b)

The function $L_{p_{Min_I}}(\textit{moves it to newInst}|\textit{min}_I)$, indicating the likelihood of the user, when moving the minimum-slider, then moving it to *newInst*, is estimated by a Gaussian function of the form

$$L_{p_{Min_I}}(\textit{moves it to newInst}|\textit{min}_I) = e^{-\frac{(newInst-min_I)^2}{2s^2}}, \tag{5.37}$$

where the parameter $s > 0$ determines the width of the curve's peak and the likelihood of the user moving the slider to the actual minimum *min*$_I$ is 1. It is depicted in Figure 5.13(b). The rationale behind that choice is, that it is more likely, that the user will move the minimum-slider to an instance that is close to the actual minimum *min*$_I$, than moving it to an instance that is distant from that instance. The parameter s of the function is selected in a way ensuring, that the likelihood of moving the slider to an instance differing from the actual minimum *min*$_I$ by more than a certain fraction f of the range R_I, is smaller than a certain fraction f' of the actual minimum's likelihood, i.e. is smaller than f' (see Figure 5.13(b)). In our implementation we chose $f = 0.25$ and $f' = 0.05$, independent from the considered instance set's type and the size of the range R_I, i.e. the likelihood of instances that differ from *min*$_I$ by more than $\frac{|R_I|}{4}$ is lower than 5% of that of *min*$_I$. This means, that likelihood curves referring to a wide range have a wide peak and those referring to a small range have a narrow peak. This models the fact that users exhibit a higher adjustment sensitivity, if the range of potential minima and maxima is smaller. That is, given for example a small range of price values, it is likely that the user will specify the minimal acceptable price more precisely than he would, if being provided with a wide range of potential price values. At this point, we would like to remark, that both, an appropriate choice of s in Formula 5.37 as well as an appropriate choice of the parameters l_{min} and l_{max} in Formula 5.36 might be user-specific and might depend on the type of the considered set I and the importance of the attribute condition it is target set of. In our implementation, we did not account for those facts and decided in favor of the more viable likelihood estimations presented above. The evaluation results that will be provided in Chapter 8 indicate, that the request models constructed based on those likelihood functions describe the user's actual service requirements reasonably well and thus are sufficient for our implementation.

The foregoing argumentation analogously holds, if the user adjusts the maximum-slider of a range via the graphical model representation. We therefore relinquish a detailed discussion of this interaction type. Range-condition-related updates implicitly triggered by the user having critiqued attribute values of a service offer, that has been recommended by the system, are discussed in Section 5.9.2.

(Not-)in-condition-related Interactions Via the graphical request model representation, users may specify uncertain (not-)in-conditions on an attribute's target set I. Upon the addition of an instance i to an uncertain in-condition, the probability $p_{\mathit{In}_i^I}(true)$ of that instance being acceptable for the user is set to 1.0. Its preference value $\mathit{pref}_I^{in}(i)$ is set to 0.5 by default. Similarly, if an instance is added to a not-in-condition, the probability $p_{\mathit{NotIn}_i^I}(true)$ of that instance being not acceptable for the user is set to 1.0. This means, if the user explicitly adds an instance to an in- or not-in-condition, we assume, that he is confident of the stated requirement. Upon the removal of an instance i from either a not-in- or an in-condition, the probabilities $p_{\mathit{In}_i^I}(true)$ and $p_{\mathit{NotIn}_i^I}(true)$, respectively, are set to 0.0[21]. In case of an in-condition, the corresponding preference value is discarded.

Users may also increase or decrease the preference value of an instance i that is considered in an in-condition. This is done in steps of $+/-0.2$ via the graphical model representation and triggers appropriate Bayesian updates of the probability distribution $p_{\mathit{In}_i^I}$. The posterior probability $p_{\mathit{In}_i^I}(true|\mathit{interaction})$ of i being acceptable after having observed an *increase-* or *decrease-interaction* is given by

$$\frac{P(\mathit{interaction}|true) \cdot p_{\mathit{In}_i^I}(true)}{P(\mathit{interaction}|true) \cdot p_{\mathit{In}_i^I}(true) + P(\mathit{interaction}|false) \cdot (1 - p_{\mathit{In}_i^I}(true))}, \quad (5.38)$$

where $P(\mathit{interaction}|true)$ is the probability of the interaction being performed, given that In_i^I in fact is $true$, $P(\mathit{interaction}|false)$ is the probability of the interaction being performed, given $\mathit{In}_i^I = false$, and $p_{\mathit{In}_i^I}(true)$ is the prior probability, i.e. the probability

[21] Since this is the default for non-considered instances, we can simply remove the maintained probability.

of the considered instance i being acceptable before taken the evidence given by the interaction into account. Thereby, we assume that an increase interaction is very likely, if in fact $In_i^I = true$, and very unlikely, if $In_i^I = false$. This inversely holds for decrease interactions. In particular, we choose

$$P(\mathit{interaction}|true) = \begin{cases} 0.9 & \text{if interaction is } \textit{increased preference of } i \\ 0.1 & \text{if interaction is } \textit{decreased preference of } i \end{cases}$$

and $P(\mathit{interaction}|false) = 1 - P(\mathit{interaction}|true)$, since In_i^I is a binary variable. The probability $p_{In_i^I}(false|\mathit{interaction})$ of i being not acceptable after the observation of an *increase-* or *decrease-interaction* is given by $1 - p_{In_i^I}(true|\mathit{interaction})$.

Connecting-strategy-related Interactions

As discussed in Section 5.9.1, connecting strategies can not be explicitly specified by the user. Instead, they are automatically initialized, when the user adds a new attribute condition to the request model. However, users may either explicitly or implicitly change an attribute condition's expected weight by performing model interactions that cause an update of the attribute condition's weight distribution, i.e. the probability distribution that maintains the system's knowledge about the condition's absolute weight. Weight distributions are affected by two types of interactions, namely, either because the user directly adjusts the weight of an attribute condition via the graphical representation of the request model or since he chooses a compromise after having critiqued one of the recommended service offers. In both cases, a Bayesian update of the affected weight distribution(s) is performed. Considering the weight W_{ac_j} of an attribute condition ac_j, the updated distribution taking a given interaction into account is given by

$$p_{W_{ac_j}}(w|\mathit{interaction}) = c_{p_{W_{ac_j}}}^{\mathit{interaction}} \cdot L_{p_{W_{ac_j}}}(\mathit{interaction}|w) \cdot p_{W_{ac_j}}(w), \tag{5.39}$$

where $p_{W_{ac_j}}(w)$ is the prior weight distribution before the update, $L_{p_{W_{ac_j}}}(\mathit{interaction}|w)$ is the likelihood function indicating the likelihood of observing the interaction when the attribute's true weight is w and $c_{p_{W_{ac_j}}}^{\mathit{interaction}}$ is a normalizing constant.

Similar to the likelihood function that is used for updates caused by the adjustment of a range slider (cf. Formula 5.35), the likelihood $L_{p_{W_{ac_j}}}(\textit{adjusts weight of } ac_j \textit{ to newWeight}|w)$ of the user changing the weight of attribute condition ac_j to the new weight *newWeight* via the graphical representation of the request model is given by

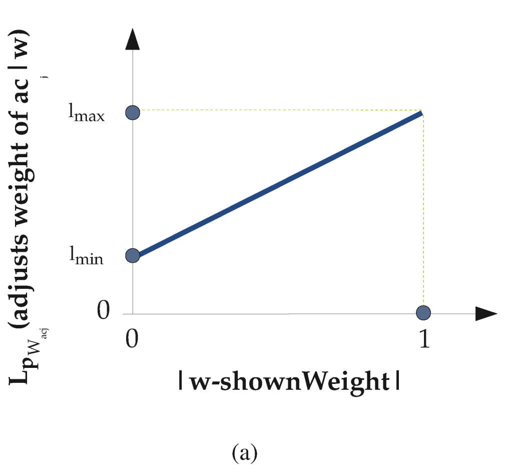
$$\begin{aligned} & L_{p_{W_{ac_j}}}(\textit{adjusts weight of } ac_j \textit{ to newWeight}|w) \\ & = L_{p_{W_{ac_j}}}(\textit{adjusts weight of } ac_j|w) \cdot L_{p_{W_{ac_j}}}(\textit{adjusts it to newWeight}|w), \end{aligned}$$

i.e. the product of the likelihood $L_{p_{W_{ac_j}}}(\textit{adjusts weight of } ac_j|w)$ of the user ever adjusting the weight of ac_j and the likelihood $L_{p_{W_{ac_j}}}(\textit{adjusts it to newWeight}|w)$ of changing it to *newWeight*, in case of making an adjustment.

Since it is likely that a user adjusts an attribute condition's weight, if it differs much from the actual weight of the condition, the likelihood $L_{pW_{ac_j}}(\textit{adjusts weight of } ac_j|w)$ is set to

$$L_{pW_{ac_j}}(\textit{adjusts weight of } ac_j|w) = (l_{max} - l_{min}) \cdot |w - \textit{shownWeight}| + l_{min}, \tag{5.40}$$

where *shownWeight* is the weight of attribute condition ac_j as displayed in the graphical model representation, i.e. is the most probable weight of the attribute condition ac_j (cf. Section 5.5), $l_{min} \geq 0$ is the likelihood of the user adjusting the weight, even if the displayed value corresponds to the actual weight, and $l_{max} \geq l_{min}$ is the likelihood of the user adjusting the weight, if its value maximally differs from the actual weight. The resulting likelihood function depends on the distance $|w - \textit{shownWeight}|$ between the displayed weight and the actual weight. It is depicted in Figure 5.14(a).



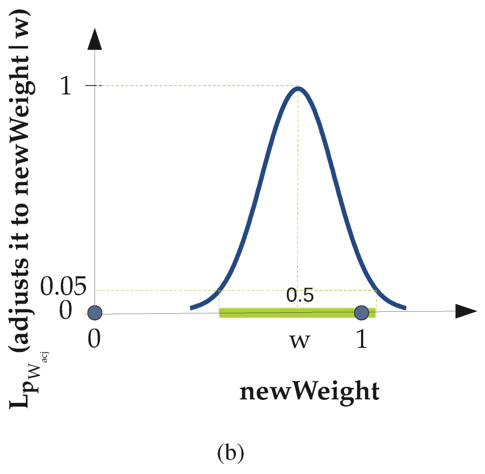


Figure 5.14.: Likelihood functions $L_{pW_{ac_j}}(\textit{adjusts weight of } ac_j|w)$ (a) and $L_{pW_{ac_j}}(\textit{adjusts it to newWeight}|w)$ (b)

To model the fact that, in case of adjusting an attribute condition's weight, it is likely that a user will provide a new weight that is close to the condition's true weight w, the likelihood $L_{pW_{ac_j}}(\textit{adjusts it to newWeight}|w)$ of the user adjusting the weight to *newWeight* is estimated by a Gaussian function of the form

$$L_{pW_{ac_j}}(\textit{adjusts it to newWeight}|w) = e^{-\frac{(newWeight-w)^2}{2s^2}}, \tag{5.41}$$

where the parameter $s > 0$ determines the width of the curve's peak and the likelihood of the user adjusting the weight to the actual weight w is 1. It is depicted in Figure 5.14(b). In our implementation, the parameter s of the likelihood function 5.41 is selected in a way that ensures, that the likelihood of providing a new weight differing from the actual weight w by more than 0.25, is smaller than 0.05. Moreover, we chose $l_{min} = 0.05$ and $l_{max} = 1$.

As discussed before, an appropriate choice of s in Formula 5.41 as well as an appropriate choice of the parameters l_{min} and l_{max} in Formula 5.40 might be user-specific. Moreover, both likelihood functions might depend on the displayed weights of other attribute conditions that have been specified in the request model. Our implementation is based on the likelihood estimations presented above, which do not account for those facts, but are more viable. The evaluation results that will be presented in Chapter 8 indicate, that the request models constructed based on those likelihood functions describe the user's actual service requirements reasonably well and thus are sufficient. A discussion of weight-related model updates caused by the user having chosen a compromise is presented in the subsequent section.

5.9.2. Implicit Model Interactions

Users may indicate desirable service characteristics relative to the properties of a matching service offer that is presented in the results table (cf. Section 5.7). This is done by critiquing the attribute values of the displayed offer and in return accepting a set of compromises on other attribute's values by choosing one of the available tradeoff alternatives determined by the system. System interactions of the described type provide valuable information about both, the involved attributes' importance, i.e. the corresponding attribute conditions' weights in the request, as well as about the user's constraints on potential values of those attributes, i.e. information about direct conditions on the corresponding attribute conditions' target sets. In the remainder of this section, we will explicate how those information are acquired based on the observed critique-and-compromise-interactions and how they are integrated into the corresponding probability distributions by using Bayesian inference.

Having detected a critique-and-compromise-interaction performed by the user based on a given service offer, the type of the visible (in the graphical request model representation introduced in Section 5.5) instance sets that are referenced in the current request model is replaced by that of the corresponding sets of the critiqued offer[22]. This is to account for the fact, that by choosing the critiqued offer as a reference, the user indicated that he is interested in offers of the selected type[23]. In addition to these type adjustments, the weight distributions of the attribute conditions that correspond to the critiqued and the compromised attributes are adjusted according to Formula 5.39. Attribute conditions that have not yet been considered in the request model are created and corresponding weight distributions are initialized. Since it can be expected that a user is likely to critique an attribute that is important to him, i.e. that has a high weight, the likelihood $L_{pW_{ac_j}}(\mathit{critiqued}|w)$ of an attribute a_j being critiqued, if its true weight is w, is chosen to be linearly increasing with w (see Figure 5.15(a)). Similarly, it can be assumed that a user is rather likely to

[22]As already mentioned, these are valid subtypes of the corresponding types in the current request model, since otherwise the critiqued offer would not have matched to the request model.

[23]This is due to the fact, that as mentioned before we do not maintain uncertainty about the type of service attributes.

compromise those attributes, that are not important to him, i.e. that have a low weight, than those that are important to him. Hence, the likelihood $L_{pW_{ac_j}}(\textit{compromised}|w)$ of an attribute a_j being compromised, if its true weight is w, is chosen to be linearly decreasing in w (see Figure 5.15(b)). More specifically, $L_{pW_{ac_j}}(\textit{critiqued}|w)$ is estimated by

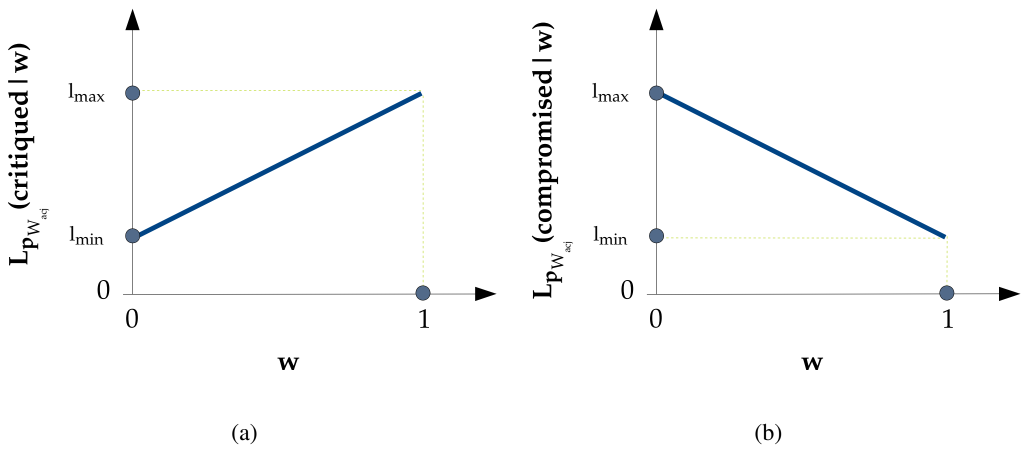


Figure 5.15.: Likelihood functions $L_{pW_{ac_j}}(\textit{critiqued}|w)$ (a) and $L_{pW_{ac_j}}(\textit{compromised}|w)$ (b)

$$L_{pW_{ac_j}}(\textit{critiqued}|w) = (l_{max} - l_{min}) \cdot w + l_{min} \tag{5.42}$$

and $L_{pW_{ac_j}}(\textit{compromised}|w)$ by

$$L_{pW_{ac_j}}(\textit{compromised}|w) = (l_{min} - l_{max}) \cdot w + l_{max}, \tag{5.43}$$

where w is the true weight of the attribute condition ac_j corresponding to the critiqued/-compromised attribute a_j, $l_{min} \geq 0$ is the likelihood of the user critiquing/compromising the attribute, if the true weight of its corresponding attribute condition is lowest/highest possible, and $l_{max} \geq l_{min}$ is the likelihood of the user critiquing/compromising the attribute, if the true weight of its corresponding attribute condition is highest/lowest possible. In our implementation, we chose $l_{min} = 0.05$ and $l_{max} = 1$.

As stated in Section 4.2, DSD provides means to specify nested attribute(s) (conditions), which allow for a more and more fine-grained characterization of service effects and their properties. For instance, the printout delivered by a printing service can be described by a nested attribute *printout*, which might be characterized more precisely by nested attribute conditions *location*, *color* and *resolution* specifying the printout location, color and resolution. The latter might in turn be detailed by attribute conditions *value* and *unit* indicating the offered printout resolution(s) as well as the unit in which the resolution is provided.

Critique-and-compromise-interactions also provide information about the importance, i.e. the weight, of those nested attributes, if their constituent attributes have been critiqued or compromised. In particular, it can be expected that a user is likely to critique a constituent of a nested attribute, if the latter is important to him, i.e. has a high weight, and is likely to compromise a constituent of a containing attribute that is not important to him, i.e. that has a low weight. This is attributed to the fact, that the individual facets of a(n) (non-)important attribute are likely to be also (non-)important to the user. However, the validity of this statement is relativized with increasing nesting level of the constituent attribute. The reason for this is, that constituent attributes at a high nesting level refer to a minor part of the requirements encoded by the nested, i.e. compound, attribute and thus are likely to be of no or limited relevance to them. Hence, drawing conclusions about the importance of such an attribute from knowledge about the containing attribute's importance is not meaningful. For instance, considering again the printing service example, it is likely that, if the delivered printout is important to the service requester, e.g. in contrast to the price of the provided service, its constituent attributes *location*, *color* and *resolution* are also important, since they make up the main properties of the delivered printout. However, importance of the *printout*-attribute does not necessarily imply that, e.g. a particular resolution unit is also a feature that is important to the user. To account for that, a critiquing or compromise interaction performed on a given service attribute, not just triggers an update of the affected attribute condition's weight distribution, but also a cascade of updates on the weight distributions referring to the nested attribute conditions containing the attribute condition that corresponds to the critiqued or compromised attribute. For instance, critiquing or compromising the attribute *unit* in our example, results in an update of the weight distribution referring to the *unit*-attribute's corresponding attribute condition as well as in cascading updates of the weight distributions assigned to the nested attributes *resolution* and *printout*'s weight distributions. Thereby, the impact of the update is reduced with increasing distance between the attribute condition, which is affected by the interaction and that, whose weight distribution is updated.

Consider the update of the nested attribute condition ac_j's weight distribution. In compliance with the Formulas 5.42 and 5.43, the likelihood $L_{pW_{ac_j}}(\textit{constituent condition } ac_k \textit{ critiqued}|w)$ of ac_j's constituent attribute condition ac_k being critiqued, if the true weight of ac_j is w, is estimated by

$$\begin{aligned} &L_{pW_{ac_j}}(\textit{constituent condition } ac_k \textit{ critiqued}|w) \\ &= (l_{max,nl(ac_j,ac_k)} - l_{min,nl(ac_j,ac_k)}) \cdot w + l_{min,nl(ac_j,ac_k)}. \end{aligned} \quad \boxed{5.44}$$

The likelihood $L_{pW_{ac_j}}(\textit{constituent condition } ac_k \textit{ compromised}|w)$ of ac_j's constituent attribute condition ac_k being compromised, if the true weight of ac_j is w, is estimated by

$$\begin{aligned} &L_{pW_{ac_j}}(\textit{constituent condition } ac_k \textit{ compromised}|w) \\ &= (l_{min,nl(ac_j,ac_k)} - l_{max,nl(ac_j,ac_k)}) \cdot w + l_{max,nl(ac_j,ac_k)}. \end{aligned} \quad \boxed{5.45}$$

The impact of the update is reduced with increasing nesting level $nl(ac_j, ac_k)$ of the affected attribute condition ac_k with respect to ac_j by successively halving the ratio $\frac{l_{max}}{l_{min}}$ in the utilized likelihood functions. More specifically, we set

$$l_{max,j+1} = \max(\frac{l_{max,j}}{2}, l_{min,j})$$
$$l_{min,j+1} = l_{min,j},$$

$l_{min,0} = l_{min}$ and $l_{max,0} = l_{max}$. Figure 5.16 exemplary illustrates that for the likelihood function $L_{pW_{ac_j}}(\textit{constituent condition } ac_k \textit{ critiqued} \,|w)$. Note, that the Formulas 5.42 and 5.43 are special cases of the Formulas 5.44 and 5.45, respectively, where the affected attribute condition and that condition, whose weight distribution is updated, coincide, i.e. $j = k$.

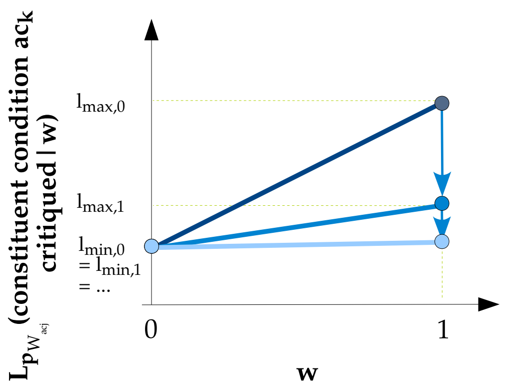


Figure 5.16.: Adjustment of the parameters l_{min} and l_{max}

Critique-interactions do not only provide information about the importance of attributes, but also knowledge about their target set type as well as about the user's constraints on their values. While the former can be modeled by means of proper type conditions, the latter have to be integrated into the request model by adding suitable direct conditions and/or adjusting them appropriately. Thereby, the type condition, that needs to be added to the critiqued attribute's target set, directly emerges from the type of the critiqued attribute. The kind of direct condition related knowledge that can be acquired from a critiquing interaction and thus the kind of updates that are triggered on the model level differs depending on the target type of the critiqued attribute as well as on the kind of critique that has been applied.

Let a given attribute be of a numerical type, such as *Integer* or *Double*, and let there be a critiqued offer taking the value *offValue* with respect to that attribute. Then either a critique of type *more than offValue*, a critique of type *less than offValue* or no critique at all can be provided with respect to that attribute (cf. Section 5.7). All three kinds of interactions convey knowledge about the minimal and the maximal acceptable value of the considered attribute, i.e. information about the range of acceptable instances that are member of the target set I referred to by the attribute condition corresponding to that attribute. It can be modeled by means of an appropriate range condition on the set I and thus triggers updates of the probability distributions p_{Min_I} and p_{Max_I}. If necessary, those distributions are created and initialized first, as described in Section 5.9.1.

As discussed in Section 5.9.1, the posterior probability distribution over possible range minima $p_{Min_I}(\mathit{min}_I|\mathit{interaction})$ taking the evidence provided by an interaction, here a critiquing interaction or the absence of a critiquing interaction on a given attribute, into account is given by Formula 5.34. This analogously holds for $p_{Max_I}(\mathit{max}_I|\mathit{interaction})$, the posterior probability distribution over possible range maxima. Given that the true minimum of the range of acceptable instances that are member of the set I is *min*$_I$, the likelihood of the interaction *more than offValue* is high, if *offValue* is lower than *min*$_I$. It is the higher the larger the distance between *offValue* and the true minimum is. The reason for this is, that whenever an offered attribute value is lower than acceptable, the user has an incentive to indicate that. The incentive is the stronger and thus the likelihood of interaction is the higher, the larger the discrepancy between the offered value and the required minimum level of the critiqued attribute is. If *offValue* is higher than *min*$_I$, the likelihood of the interaction *more than offValue* is low. It is the lower, the larger the distance between *offValue* and the true minimum is. This is due to the fact, that the user has little or no incentive to state that a certain attribute value is too low, if he is willing to accept a lower value. The incentive is particularly low, if the attribute value is much higher than the minimal acceptable value. Those remarks inversely hold for a given maximum of a range and interactions of type *less than offValue*. We use logistic likelihood functions of the form

$$f(\mathit{offValue}; \mu, s) = \frac{1}{1 + e^{-(\mathit{offValue} - \mu)/s}}$$

to model the described behavior, where μ is a location parameter and s is a scale parameter. In particular, given the true minimum *min*$_I$ and the true maximum *max*$_I$, we set the likelihood $L_{p_{Min_I}}(\mathit{more\ than\ offValue}|\mathit{min}_I)$ of the user performing the interaction *more than offValue* on a given attribute of the service offer selected for critiquing to

$$L_{p_{Min_I}}(\mathit{more\ than\ offValue}|\mathit{min}_I) = f(\mathit{offValue}; \mathit{min}_I, \frac{|R_I|}{4 \cdot \ln(\frac{l}{1-l})})$$

and the likelihood $L_{p_{Max_I}}(\mathit{less\ than\ offValue}|\mathit{max}_I)$ of the user performing the interaction *less than offValue* on that attribute and offer to

$$L_{p_{Max_I}}(\mathit{less\ than\ offValue}|\mathit{max}_I) = f(\mathit{offValue}; \mathit{max}_I, \frac{|R_I|}{4 \cdot \ln(\frac{1-l}{l})}).$$

The scale parameter s is chosen in a way ensuring that the likelihood of interaction is lower or equal to l, if $\mathit{offValue} \geq \mu + \frac{|R_I|}{4}$ or $\mathit{offValue} \leq \mu - \frac{|R_I|}{4}$, respectively. If l is set close to 0 (0.05 in our implementation), it is ensured that the likelihood of interaction for offer values lower than $min_I - \frac{|R_I|}{4}$ ($max_I - \frac{|R_I|}{4}$) is close to maximal (minimal) and that of values larger than $min_I + \frac{|R_I|}{4}$ ($max_I + \frac{|R_I|}{4}$) is close to minimal (maximal). The resulting likelihood functions $L_{p_{Min_I}}(\textit{more than offValue}|min_I)$ and $L_{p_{Max_I}}(\textit{less than offValue}|max_I)$ are depicted in Figure 5.17.

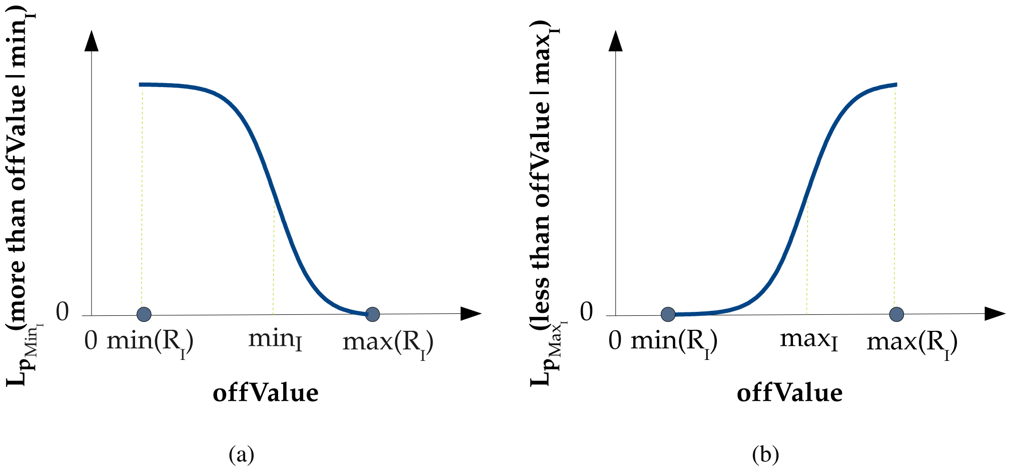


Figure 5.17.: Likelihood functions $L_{p_{Min_I}}(\textit{more than offValue}|min_I)$ (a) and $L_{p_{Max_I}}(\textit{less than offValue}|max_I)$ (b)

Beside of the fact that a critiquing interaction has been performed on a certain attribute of a critiqued offer, also the observation that those interactions have not been carried out on that attribute contributes knowledge about the minimum and the maximum of the range of acceptable values of that attribute. Thereby, the likelihood $L_{p_{Min_I}}(\neg \textit{more than offValue}|min_I)$ of the user not performing the interaction *more than offValue* on a given attribute of a critiqued service offer is given by

$$\begin{aligned} L_{p_{Min_I}}(\neg \textit{more than offValue}|min_I) &= 1 - L_{p_{Min_I}}(\textit{more than offValue}|min_I) \\ &= f(\textit{offValue}; min_I, \frac{|R_I|}{4 \cdot \ln(\frac{1-l}{l})}) \end{aligned} \tag{5.46}$$

and the likelihood $L_{p_{Max_I}}(\neg \textit{less than offValue}|max_I)$ of the user not performing the interaction *less than offValue* on that attribute is given by

$$\begin{aligned} L_{p_{Max_I}}(\neg \textit{less than offValue}|max_I) &= 1 - L_{p_{Max_I}}(\textit{less than offValue}|max_I) \\ &= f(\textit{offValue}; max_I, \frac{|R_I|}{4 \cdot \ln(\frac{l}{1-l})}). \end{aligned} \tag{5.47}$$

Note, that the provided likelihood estimations are based on the assumption that the critiquing behavior of the user is exclusively determined by the value of the considered attribute. However, a user might also decide to not critique a certain attribute value, since a similar critique has been already provided. In such a case, using the likelihood estimations provided above (Formulas 5.46 and 5.47) would lead to the undesirable conclusion that the critique has not been provided, since the offered attribute value was acceptable. This in turn would result in inappropriate updates of the probability distributions p_{Min_I} and p_{Max_I}. Nonetheless, we argue that using the suggested likelihood estimations is still feasible, since it is unlikely that a user is challenged to provide a certain critique or similar critiques more than once. This is due to the fact, that the model update triggered by a former critiquing operation causes offers with similar attribute values to be ranked low.

Beside of the discussed critiques, which refer to numerical attributes, users may also apply critiques of type *!=offValue*, indicating that the instance *offValue* provided by the critiqued offer is not acceptable. The application of this kind of critique is restricted to nominal attributes such as *Color* (cf. Section 5.7). Let there be a critiqued offer taking the value *offValue* with respect to a given nominal attribute. Then the critique *!=offValue* provides knowledge about non-acceptable attribute values with respect to the corresponding attribute condition's target set I. This can be modeled by means of an appropriate in- or not-in-condition on the set I and triggers an update of the involved probability distributions $p_{In^I_{offValue}}$ and $p_{NotIn^I_{offValue}}$, respectively. Given the critique *!=offValue*, we have to distinguish several cases depending on the type of direct condition(s) that have been already specified for I in the request model. If there is an in-condition that has been already defined for the set I, we have to check whether $p_{In^I_{offValue}}(true) \neq 0$ or $p_{In^I_{offValue}}(true) = 0$. In the latter case, nothing needs to be done, since the performed interaction does not provide any additional information. In the former case, we perform a Bayesian update of $p_{In^I_{offValue}}$ according to Formula 5.38 with $P(\textit{!=offValue}|true) = 0.1$ and $P(\textit{!=offValue}|false) = 0.9$. This means, the probability of observing the interaction *!=offValue* is high, if *offValue* is in fact not acceptable and low otherwise. Instead of an existing in-condition, there might have been a not-in-condition already specified with respect to the set I. In case of $p_{NotIn^I_{offValue}}(true) \neq 0$, we perform a Bayesian update analogously to Formula 5.38, with $P(\textit{!=offValue}|true) = 0.9$ and $P(\textit{!=offValue}|false) = 0.1$, i.e. the probability of observing the interaction *!=offValue* is high, if *offValue* is in fact not acceptable and low otherwise. In case of $p_{NotIn^I_{offValue}}(true) = 0$, we set $p_{NotIn^I_{offValue}}(true) = 1$, since in contrast to the former case, we did not yet observe any evidence for $NotIn^I_{offValue} = false$[24]. Finally, if neither a not-in- nor an in-condition has been already specified with respect to I, we create a direct condition of type not-in and proceed as already described.

[24]We know that, since $p_{NotIn^I_{offValue}}(true) = 0$ is the default for not yet considered instances, which due to the choice of $P(\textit{!=offValue}|true)$ and $P(\textit{!=offValue}|false)$ cannot result from a Bayesian update.

6 Modeling, Elicitation and Usage of Consumer Feedback

Enabling well-informed and balanced service selection decisions in an environment where knowledge about service capabilities might be inaccurate and incomplete, requires to make potential service consumers aware of the risk that is associated with the execution of a service. As argued in Section 1.4, a feedback mechanism is able to provide the knowledge that is necessary to perform this task. However, so far we owe an analysis of the conditions under which it will be both, effective in terms of acquiring the demanded information and effective in terms of its ability to support service selection (Objective 2). This chapter, starts with the missing analysis (Section 6.1) followed by a thorough investigation of related research efforts and their relation to the identified requirements as well as an analysis of open research issues (Section 6.2). The remainder and main part of the chapter is dedicated to the presentation of our own collaborative feedback mechanism that is part of our solution and has been designed to fulfill the identified requirements. After outlining the basic idea of the developed solution (Section 6.3), we will introduce its underlying feedback model (Section 6.4) as well as its feedback elicitation approach (Section 6.5). We will also detailedly explain how available feedback is effectively exploited to predict the future performance of a service and thus to assess the risk that is associated with its execution (Section 6.6). Finally, we will describe how those information can be visualized and used to rank available service alternatives in a user-specific way (Section 6.7). The contributions presented in this chapter have been partially published in [KKR08, FK10, KKR10a, KKR10b].

6.1 Requirements

Collaborative feedback mechanisms are an active area of research (see [JIB07] for an overview) and have been widely and successfully used to establish ”stability in otherwise very risky trading environments” [Del02]. However, the diversity and multi-faceted nature of Semantic Web Services imposes special requirements on such a mechanism, if applied to assess the risk that is associated with the execution of inaccurately and incompletely described services. As we will show, these are only partially met by existing solutions

(Section 6.2). In the following, we will specify those requirements. Thereby, the focus of our analysis and also of this thesis is on enabling effective elicitation and usage of truthful feedback. Aspects such as dealing with false or dishonest consumer feedback are orthogonal to our work and out of scope. See [JIB07] for further information and solutions on these topics.

Consumer feedback is subjective, since it reflects a service's performance as observed through a certain consumer's eyes. Hence, feedback is biased by personal expectations and preferences about the invoked service, which are encoded in the service request[1] that was posed (*request* or *feedback provider context*). For instance, a ticket booking service might have been used to buy group tickets for a school class or to buy a single ticket. The subjective performance of the invoked service is likely to differ depending on the kind of request that was posed, e.g. in the first case, the availability of a price discount for groups might have a positive impact on the perceived performance, while, in the second usage scenario, this fact has no influence. Moreover, a service provider's performance is typically dependent on the type of service that is offered (*service (offer) context*) [Del02, Mui02, Sab05]. For instance, a service provider might perform well, when offering to make train reservations, but performs badly when offering to book trips. Effective feedback mechanisms for Semantic Web Services should account for those facts by **taking the request and service context, in which a judgment was made, into account** when using feedback to predict a service's future performance.

To enable its effective usage, feedback has to be **meaningful**. In particular, this means that **the request and the service context underlying an expression of feedback should be clear**. We illustrate this issue with an example. Suppose provider p offers a printing service. Imagine that we are provided with consumer feedback that refers to p and that consists of a single rating, indicating that the service provided a good printout quality. Unfortunately, this information alone will not allow us to infer that p will provide printouts of good quality to another consumer. This is due to the fact, that we do not know enough about the context in which the quality judgment was made, e.g. to which type of printing service the quality judgment refers to, and about the preferences and expectations of the judgment provider. Consequently, to obtain meaningful feedback, we need to model and record the context in which a provided judgment was made. In addition, it should be evident whether and how feedback made under one circumstance can be used to infer about a service provider's performance in another situation or even when providing another service. In short, feedback is meaningful, if it contains all information that are necessary to exploit it effectively.

We would also like to emphasize the necessity of feedback to **be as detailed as possible**, i.e. comprising of judgments referring to various aspects of a service interaction. The reasons for this are manifold. Firstly, feedback, judging the quality of a provided service as a whole, is of limited significance, since as an aggregated judgment it provides not more than a rough estimate of a service's performance. Secondly, risk attitudes might differ

[1]This service request could be either a manually created service request or, in case of our approach, the request that was generated on base of the request model (cf. Section 5.6).

among users. For instance, while one consumer is risk averse with respect to the delivery time of a service and risk neutral with respect to other service aspects, another one is risk averse with respect to the service's quality, but not with respect to its delivery time. Feedback mechanisms that are based on aggregated judgments do not allow to consider risk attitudes at the attribute level by combining and considering feedback information in a user-defined way and thus can only partly adjust to different risk attitudes. Finally, aggregated feedback tends to be inaccurate. The reason for this is that humans are bad at integrating information on different aspects, such as delivery time, price or quality of a service, as they appear in a multi-faceted service interaction, in particular if those aspects are diverse and incomparable [Daw79, Slo72]. As a consequence, one can expect that aggregated feedback referring to the performance of a multi-faceted service interaction will be inaccurate to some degree.

In the context of detailed, i.e. multi-aspect, consumer feedback additional aspects related to the meaningfulness of judgments arise. In particular, **meaningful judgments should refer to the consumer's service requirements**, i.e. the service aspects that have been specified in the service request (model). Otherwise, it would be neither clear which aspects of a service interaction were considered in a judgment nor according to which scale they were judged. This in turn is a prerequisite for the comparability of judgments and thus for the assessment of their relevance when making predictions as well as for the assessment of the risk that is associated with the execution of a service. Moreover, meaningful means, that **the concepts used to describe service aspects and thus the concepts used to describe judged aspects, valid constraints on them as well as valid relationships among them are shared among the system participants**. In doing so, the meaning of judged service aspects is made explicit, consistent among the judgment providers and machine-comprehensible. This property is essential for effective and efficient feedback usage, since it enables the comparability of judgments provided by different users and for different services as well as their automated processing.

Detailed feedback should also be **comprehensive**, which means that **all service aspects that are relevant to a feedback provider should be judged**. This is due to the fact, that inferred judgments based on incomplete information might be incorrect. We illustrate this with an example. Suppose a service provider p offers a shipping service. A consumer who used this service experienced a good performance with respect to the delivery time, but a bad performance with respect to the price, which was quite high. Leveraging just the provided positive delivery time judgment for a service's performance prediction, could possibly lead to undesired conclusions about the performance of the service, e.g. that the service offers fast delivery for a low price. Hence, judgments for both price and delivery time should be elicited. Finally, the judged service aspects should be **appropriate**, i.e. meaningful in the context of the considered service interaction, even if the services and requests (request models) that might be involved in service interactions are diverse and potentially refer to different application domains. For instance, it makes sense to judge an aspect "taste" when referring to a wine selling service, but not when assessing the performance of a ticket booking service. The system should be able to automatically determine those service aspects that are appropriate judgment targets in the context of a specific service interaction.

Another problem we encounter when dealing with consumer feedback on the basis of service interactions is feedback scarcity. Since the services and requests (request models) that are involved in service interactions are diverse, feedback related to a certain request context and a particular service is rare and typically not available at all. Moreover, available feedback is based on a single sample of a service's performance. Hence, **feedback is scarce and has to be exploited effectively**. In particular, service experiences related to different, but similar request contexts and those related to other, but similar, services of a considered provider (a different service (offer) context) have to be leveraged. Due to feedback scarcity and diversity, the **provision of a confidence measure**, indicating how sure the system is about the predicted performance, is indispensable.

Using second-hand feedback also means sharing own feedback as well as interaction-related data with others. Since those information may be used to infer personal knowledge about a consumer, such as his service requirements and preferences, a feedback system should carefully select the information that are propagated to other system participants. **Only necessary information** and those only **in a quality that is required to allow for a desired prediction accuracy**, **should be shared**. Nonetheless, the privacy restrictions of the system participants may be different and may vary depending on the request context. For example, a user might have strong privacy restrictions when using a service to purchase pharmaceuticals, but weak restrictions when using a printing service. Hence, **the quality of the shared information should be adjustable** to account for differing privacy needs.

However, to be able to unfold the full potential of consumer feedback, particularly when using multi-aspect feedback, it is required that users provide meaningful responses. To ensure this, the process of feedback elicitation should not just **take care of elicited feedback being comprehensive, appropriate and meaningful** in the context of a certain service interaction, but **should also consider a consumer's willingness** to provide feedback. This is important, since asking a consumer for a number of judgments he is not willing to provide will result in no or bad quality feedback [JIB07, LK10]. Since the willingness to judge a certain set of service aspects is likely to be context-dependent and user-specific, an effective feedback elicitation mechanism should **flexibly and automatically adjust to different judgment preferences**. Finally, a feedback system **should ensure that all information that are required for the effective exploration of consumer feedback are recorded** (meaningfulness of feedback). This should happen **automatically and transparently** for the user.

Though knowledge about the actual performance of available service alternatives is essential for enabling consumers to make well-informed service selection decisions, it is of little use, if it is not communicated and appropriately presented to the user. In particular, it is necessary to **make the user aware of the risk that is associated with the execution of a service** by presenting feedback information in an effective and intuitive way. Since risk attitudes vary among users and request contexts, the **presentation of those information should be adjustable**.

The identified requirements to consumer feedback, its elicitation, propagation, usage and presentation set the standards for the evaluation (Chapter 9) of the feedback mechanism that will be introduced in the subsequent sections. They can be summarized as follows:

Requirements to Feedback Elicitation

Requirement F.1. (**Feedback quality**) *A feedback mechanism for Semantic Web Services should ensure that elicited feedback is detailed, comprehensive and appropriate in the context of a certain service interaction and that elicited feedback is meaningful, even if the services and requests (request models) that are involved in the service interactions are diverse and potentially refer to different application domains.*

Thereby, meaningful is defined as follows:

Definition 6.1. (**Meaningfulness of consumer feedback**) *Consumer feedback is meaningful, if*

- *the provided judgments refer to the consumer's service requirements,*
- *the concepts used to describe service aspects and thus the concepts used to describe judged aspects, valid constraints on them as well as valid relationships among them are well-defined and shared among the system participants,*
- *the request and the service context underlying an expression of feedback are modeled and recorded and*
- *if it is evident whether and how feedback made in one context can be used to infer about a service provider's performance in another context.*

Requirement F.2. (**Adaptive elicitation**) *The process of feedback elicitation should flexibly and automatically adjust to a consumer's willingness to provide feedback.*

Requirements to Feedback Propagation

Requirement F.3. (**Quality of shared information**) *Shared feedback should only comprise of necessary information being of a quality that is required to allow for a desired prediction quality. The quality of the shared information should be adjustable to account for differing privacy needs.*

Requirements to the Performance Prediction

Requirement F.4. (**Effective exploitation**) *Consumer feedback should be effectively exploited to predict a service provider's future performance, even if available feedback refers to service interactions that are diverse with respect to the services and requests (request models) that were involved.*

Requirement F.5. (**Context dependency**) *A feedback mechanism for Semantic Web Services should account for the context-dependent nature of service performance and service judgments by taking the request and service context, in which a judgment was made, into account when using feedback to predict a service's future performance.*

Requirement F.6. (**Prediction confidence**) *A feedback mechanism for Semantic Web Services should provide a confidence measure for its performance predictions.*

Requirements to the Presentation of Feedback-Derived Information

Requirement F.7. (**Effective presentation**) *Feedback information should be presented in a way that makes the user aware of the risk that is associated with the execution of a service.*

Requirement F.8. (**Adjustable presentation**) *Feedback information should be presented in a way that is adjustable to different risk attitudes.*

6.2 Related Work

In this section, we will analyze related research efforts with respect to the list of requirements that has been compiled in the previous section and identify open research issues. The focus of our analysis is on collaborative filtering systems with an emphasis on multi-criteria rating recommenders [AMK11] (Section 6.2.1), product reviews (Section 6.2.2), trust and reputation systems (Section 6.2.3) and previous efforts on experience-based service provider selection (Section 6.2.4). We conclude the section with a short summary of the analysis results (Section 6.2.5).

6.2.1. Collaborative Filtering

The main intention of collaborative filtering systems (see [Ado05, SFHS07] for an overview) is to recommend promising products or other items to potential consumers. In contrast to

Semantic Web Service retrieval techniques, recommendations are computed indirectly by leveraging consumer-provided item ratings and typically do not rely on explicit models of consumer requirements and item properties. Recommendations are generated by estimating unknown item ratings based on available consumer ratings and then recommending the item(s) with the highest estimated rating(s). Though the mechanisms underlying collaborative filtering (CF) systems differ from those required in our scenario, their main objective, namely predicting an item's or service's suitability based on consumer feedback, is the same and therefore we consider them in this section.

While traditional CF systems base their recommendations on overall ratings, where consumers judge an item as a whole, some preliminary research on multi-criteria rating recommenders (MCRRs) [AMK11], that utilize detailed feedback in terms of multiple criteria ratings, has been done. Though choosing an appropriate set of aspects to judge by the user has been identified as an important and challenging topic [AMK11], research on this issue has not been done yet. MCRRs typically operate on a single domain with predefined aspects, such as "actors", "story" or "special effects" in the domain of movie recommendations. Consumers are asked to judge items based on those aspects. Comprehensiveness of the provided judgments is not enforced. The consumer's willingness to judge certain aspects is not taken into account. In this context, *Lousame et al.* [LS09] propose an algorithm that recommends views, i.e. a set of item features, that might be interesting for a user. However, they do not use the acquired knowledge to support rating elicitation.

A basic assumption of collaborative filtering systems is that consumer ratings are subjective, i.e. depend on a person's personal preferences. When estimating unknown item ratings this fact is taken into account. In (user-based) collaborative filtering, this is done by considering a rating provider's neighborhood to the target user, whose unknown rating has to be predicted. In traditional CF systems, the strength of a neighborhood relationship is determined by the similarity of the users' rating profiles. It is assumed that users who rated the same items similarly, will also produce similar ratings for other items. Neither the context in which a judgment was provided nor the requirements of the user are taken into account. This is problematic in two ways, particularly in domains with heterogeneous items. Firstly, since a user might judge a single item differently when having different expectations and preferences, i.e. in different contexts or having different requirements, two users having provided similar ratings with respect to a certain set of products may not necessarily share the same taste. The other way round, having different rating profiles, i.e. different tastes for a certain set of products, does not necessarily imply different tastes in general. Hence, discarding feedback of users with a dissimilar rating profile, means discarding potentially valuable feedback that could have been exploited.

Adomavicius et al. [Ado05] consider this fact by incorporating contextual information in recommender systems. In their approach, ratings are given in a certain context, that is (explicitly!) modeled adopting the multi-dimensional data model used for OLAP applications in databases. A rating may for example refer to a movie, seen at a specific place with a specific person, where, e.g. place and accompanying person, may be interpreted as usage context. For rating estimation in a certain context, one may then consider only those ratings that have been provided in the same context. However, the approach operates on predefined context attributes in a single domain and does not consider multi-aspect ratings. It only partially addresses the problem of leveraging rating information in one context to estimate a rating in another context. *Berkovsky et al.* [BAH$^+$06, BKR08] made a first attempt to deal with this issue. They suggest an approach that supports cross-domain recommendations as well as inference between judgments that have been provided in different contexts. In a similar vein, several extensions to traditional CF-based recommenders that provide richer models of users and items have been suggested. While the majority of approaches, so-called content-based recommenders [PB07], extract item features by using data mining methods, some more sophisticated solutions that make use of taxonomies or ontologies to describe and categorize items and user interests have also been suggested [CK04, Zan09, SMB10]. In [BKR08] and [JZ09] ontological knowledge is also used to provide cross-domain recommendations. However, those approaches have not been extended to make use of multi-criteria ratings.

This is unfortunate, since multi-criteria rating recommenders have proven to produce more accurate recommendations than traditional CF systems, that are based on overall ratings, [GA07, LWG08, MC07, SKDC06]. Though CF systems perform well in many application scenarios, they have shown to suffer from a number of problems such as the cold-start problem, that arises from the fact, that for new items or users no ratings are available and thus purely CF-based approaches do not provide meaningful recommendations [SFHS07]. Besides providing recommendations, many CF systems also provide a measure indicating their confidence in a recommendation. So-called probabilistic or model-based CF systems [SFHS07, BHK98] maintain a probability distribution over possible rating values for each unknown rating and thus can compute the likelihood of a certain rating value being correct as a very natural measure of confidence. Non-probabilistic or heuristic-based collaborative filtering systems have to fall back on heuristic confidence measures, e.g. taking into account the similarity of neighbors and the number of co-rated items between the user and his neighbors. However, confidence information are typically not communicated to the user. As a consequence, consumers are not able to assess the risk that is associated, e.g. with the purchase of a recommended product.

CF systems typically "just" rely on user provided ratings to make recommendations which are shared with others. Hence, compared to systems that share explicit models of a user's requirements, they propagate information that are of a quality which makes it harder to infer personal information from it. Nonetheless, e.g. *Ramakrishnan et al.* [RKM$^+$01] have shown that a user's rating profile can be used to infer about his identity and to derive personal details when combined with additional information. In case of a context-aware system, even additional explicit information about the context in which a rating was provided are shared. Finally, we are not aware of a CF approach that allows to adjust the quality of the shared information to account for differing privacy needs.

6.2.2. Product Reviews

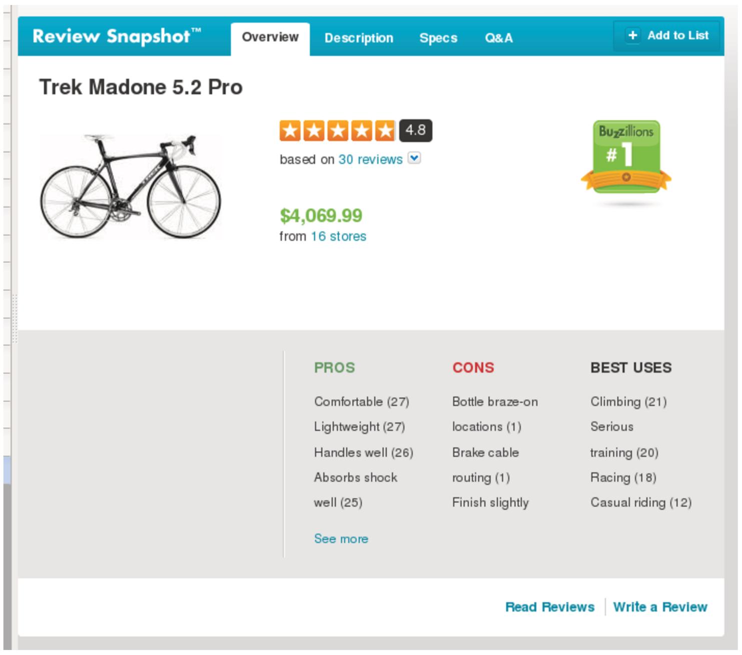
Beside the scientific approaches to recommender systems, there are numerous commercial solutions providing reviewing and recommendation facilities. Those come either as customizable standalone recommendation engines, e.g. Powerreviews.com or Bazaarvoice.com, that can be integrated into existing e-commerce sites, or are specifically designed for a certain online store such as Amazon.com or a commercial product reviewing site such as Epinions.com.

Nearly every online store offers reviewing or rating facilities. However, the capabilities of the underlying recommendation engines do not differ much [2]. Usually, those tools allow the consumer to provide a single overall rating for a given product or service provider, typically on a 5-star scale or, rarely, continuously with a slider. Additionally, some systems allow consumers to judge single aspects of a product or service provider interaction. However, those aspects are either fixed for all products or predefined per product category and hence are either generic, i.e. not product-specific, or not appropriate for all products. For instance, Epinions allows its users to judge movies with respect to the aspects *Action Factor*, *Special Effects* and *Suspense*. Those might make sense for *Action Movies*, but do not fit when considering other movie categories offered by Epinions like *Children Movies*, *Education* or *Comedy*. Epinions also allows to review online stores. They can be rated in terms of the aspects *Ease of Ordering*, *Customer Service*, *On-Time Delivery* and *Selection*. Those aspects are very general and do not allow to judge shop-specific characteristics. In any case, aspect ratings are supplementary in the sense that they do not have any influence on a product's overall rating.

Alternatively, some reviewing engines such as those provided by Bazaarvoice or Powerreviews, offer more flexible reviewing facilities based on tagging. Those systems allow consumers to create tags describing the pros and contras of a given product. These tags can then be reused by other users. As an example consider the bike review taken from the Buzzillions.com web site [3], depicted in Figure 6.1. In the pros and cons columns, one can see which tags and also how often each tag was assigned by other users. Tagging provides a very intuitive and flexible mechanism that allows for product-specific judgments. Users can freely choose the product characteristics they want to judge (and that are appropriate in the given product context) providing as much information as they are willing to provide and spending as much effort as they are willing to spend. Since tags may be reused, a tagging mechanism also introduces some kind of coherence and comparability in the consumer judgments. Moreover, tag re-usage supports the user in the judgment process, since it allows him to choose from a selection of already available tags. However, the high flexibility of the approach is at the cost of the judgments' meaningfulness. This is due to the fact that tags do not have a clear semantics. In particular, the relationship between different tags is unknown and thus makes them incomparable. Moreover, the system does not

[2]However, they do differ much in their ability to provide incentives for feedback provision as well in their ability to avoid and detect dishonest feedback. See *Josang et al.* [JIB07] for more information about this topic.

[3]Buzzillions uses the Powerreviews engine as underlying reviewing tool.



Trek Madone 5.2 Pro

4.8
based on 30 reviews

$4,069.99
from 16 stores

PROS	CONS	BEST USES
Comfortable (27)	Bottle braze-on locations (1)	Climbing (21)
Lightweight (27)	Brake cable routing (1)	Serious training (20)
Handles well (26)	Finish slightly	Racing (18)
Absorbs shock well (25)		Casual riding (12)

See more

Read Reviews | Write a Review

Figure 6.1.: Product review on the reviewing site Buzzillions.com

ensure that all relevant aspects of a product or a service interaction are judged. And again, a product's overall rating is not influenced by the tag-based judgments.

Tagging is also used to indicate a reviewer's expertise in the considered product domain. For instance, in the category *Digital Camera*, Buzzillions offers the expertise levels *Photo Enthusiast*, *Pro Photographer*, *Semi-pro Photographer* and *Casual Photographer*. Available reviews may then be filtered according to this categories. However, reviewer expertise is not considered when aggregating the single consumer ratings to an overall product rating.

Additionally, tagging mechanisms are utilized to capture a reviewer's preferences and expectations on the bought product, i.e. his buying context. For instance, in the category shoes, Buzzillions reviewers may be either *comfort driven* or *style driven* buyers. Reviews may then be filtered with respect to one of these categories. Those categories provide a simple classification of the reviewers with respect to their preferences and expectations about a certain product type. By tagging *Best Uses* of a product, consumers may also indicate their expectations on a certain product. For example, the bike depicted in Figure 6.1 is tagged with the possible uses *Climbing*, *Training*, *Racing* and so on. However, this information may be taken into account by potential buyers, but is not used by the reviewing system to filter available reviews. Usually, users may also indicate the helpfulness of a certain product review on a 4-star scale.

In general, available products are ranked considering the product's overall rating, which is usually the mean of the single overall ratings weighted with their degree of helpfulness, the number and recency of the reviews for that product as well as the trustworthiness of the reviewers who provided them. The latter factors provide a simple measure of the system's confidence in the provided reviews. However, confidence information are not well communicated to the user. For example, the product browsing view on Ebay.com holds only information on whether an offer is provided by a top-rated seller or not. To get detailed information on a sellers rating profile, one has to select a product first and then has to open the sellers rating profile. As a consequence, consumers are not able to assess the risk that is associated with the purchase of a product that is offered by a certain seller. Moreover, the systems are not adjustable to a consumer's risk attitude. For instance on Ebay.com, the only option is to restrict offers to those of top-rated sellers.

None of the approaches mentioned above, allows for effective feedback usage and thus effective rating prediction across domains. Often this is simply not required, such as when predicting the rating for a specific product (e.g. on Buzzillions). In other cases, such as on Ebay, where sellers are rated, the ratings, that typically refer to very different purchase situations, are simply aggregated without considering the context in which they have been provided. In general, most of the information, that are acquired by the mentioned systems, are not considered when ranking products. This is particularly remarkable in consideration of the fact that reviewers reveal various (personal) information, such as what products they purchased, when they purchased it, which aspects they liked and which not. Often, reviewing systems such as Epinions allow other users to freely access a reviewers profile, that provides an overview about all reviews written by that particular reviewer. However, each user decides on his own how much information he is willing to reveal.

6.2.3. Trust and Reputation Systems

Trust and reputation systems gained much research interest over the last years resulting in a multitude of approaches in this area. The purpose of those systems is to assess the trustworthiness of potential transaction partners in a system by aggregating user feedback about their behavior in former transactions. The aim is to support system participants in deciding whether or not to run into a transaction with a considered entity in the future.

Trust and reputation systems typically rely on experiences a party has directly made with a considered transaction partner as well as on second-hand experiences reported by other entities in the system. In addition, information about observed interactions as well as sociological information, such as group membership [SS01], social roles and relationships [SS02], are considered.

Reputation systems typically assume feedback to be objective, i.e. invariant to taste, and provide global reputation scores, reflecting the overall opinion of a community about their members. In contrast, trust systems derive local and subjective trust scores, i.e. different members of a community may derive different trust values for the same entity. Moreover, trust systems often consider transitive trust relationships, while reputation systems normally compute reputation scores based on direct and publicly available feedback from their members [JIB07, JBXC08].

In trust and reputation systems, the performance of a transaction partner is evaluated in terms of ratings, indicating whether or to what degree the outcome of a transaction met the expectations of the user. Those ratings come in several flavors. Often a single binary rating or a set of discrete graded levels is used [ARH00, JH07]. Other approaches take continuous ratings as input [SS01, JLC08]. However, evaluating interactions this way, requires that both parties know and agreed upon a set of actions that has to be performed in the context of a certain transaction, i.e. a contract. Most of the approaches assume that this is implicitly given and do not explicitly model transaction contracts. In those cases the performance of a service provider is typically judged in terms of a single rating assessing the outcome of an interaction as a whole. It remains unclear, what was actually judged and how the judgment relates to the requirements and the contracted aspects. In many scenarios, particularly in multi-agent systems, this is justified, since the actions that may be accomplished by the system participants are simple and very restricted. However, in a scenario with rich service interactions or scenarios where agents perform complex tasks, contracts covering the various facets of a transaction are required and have to be explicitly modeled. Consumer feedback should also account for that fact and judge an interaction with respect to multiple dimensions.

There is some work that is in line with this requirement [SS01, SS02, WV03, Gri05, GDFB06, RRRJ07b, RRRJ07a]. The approaches in [WV03, Gri05, GDFB06] focus on a fixed set of Quality of Service aspects, such as timeliness or download speed, that may be judged after a service interaction. Though this kind of feedback accounts for the multi-faceted nature of service interactions, the aspects to be rated are generic and do not cover non-quality aspects. More sophisticated ideas have been proposed by *Sabater et al.* [SS01, SS02] and *Reece et al.* [RRRJ07b, RRRJ07a]. Both approaches explicitly model contracts as a set of attributes covering several dimensions of an interaction. Those attributes may vary depending on the type of interaction. While in [RRRJ07b, RRRJ07a] attributes do not have a clear semantics, [SS01, SS02] use hierarchically organized attributes defined within an ontology. However, they do not capture the context in which an attribute has been judged. For instance, they do not distinguish between a quality rating given after purchasing a bottle of wine and a quality rating provided after using some network service. Both approaches do not explicitly enforce comprehensive feedback that covers all aspects touched by a contract, but can be easily extended in this way. However, in that case a user has to provide a rating for each attribute covered in the contract. Hence, the approaches may either adapt to the user's willingness to provide ratings, by allowing him to rate just a subset of the contracted attributes or enforce comprehensiveness of the provided feedback, but cannot do both at the same time.

Once a trust or reputation system has acquired sufficient feedback about a certain entity, it computes a trust or reputation score from those information. Using this value, a user should be able to make an informed decision about whether to run into a transaction with this provider or not. However, the concrete semantics and representation of the trust or reputation measure depends on the approach considered. When we refer to trust scores, the following definition of *Gambetta* [Gam88] is often cited: "[Trust] [...] is a particular level of the subjective probability with which an agent assesses that another agent or group of agents will perform a particular action, both before he can monitor such action [...] and in a context in which it affects his own action [...].” Several techniques have been proposed to model trust and/or reputation. The spectrum ranges from measures that simply add or average ratings to more sophisticated approaches like Bayesian-, Belief-, Fuzzy- or Flow Models. See [JIB07] for an overview. As manifold the proposed trust metrics are, as manifold are the proposed measures that indicate how reliable the computed trust score is. Often, heuristic measures considering e.g. the number of experiences, the reliability of the experience providers or the age of the information are considered. There are also more theoretically founded measures. For instance, belief models [Jøs01] represent trust as opinions expressing the relying party’s belief in the truth of a certain statement. Three parameters b, d and $u \in [0, 1]$ are reserved to represent belief, disbelief and uncertainty respectively, where $b + d + u = 1$. Trust models based on probability distributions, e.g. Bayesian models [Jøs02], may use for instance the variance of the distribution as a confidence measure. Fuzzy models such as in [CMD02] may use the inverse of the width of the fuzzy set as a measure of confidence. To the best of our knowledge, the trust-and reputation systems community did not address the issue of how to effectively and intuitively display trust and confidence information. However, in commercially used systems, such as on various question-and-answer, social networking and e-commerce web sites, trust information is typically displayed in form of a trust value (e.g. Stackoverflow.com) or some graphical equivalent (e.g. Expedia.com). Detailed information can be depicted on demand. Sometimes, available interaction partners can be filtered according to their trust score (e.g. Expedia.com). However, often this option is either not supported or allows just for a coarse classification of transaction partners, even if trust plays an important role in the considered environment, such as on crowdsourcing web sites, where people perform paid tasks for others. For instance on Freelancer.com, a work-seeking user can sort offer providers by *Featured*, *Full Time*, *Non-Trial*, *Trial* and *Gold Members Only*, on Amazon Mechanical Turk [4] there is even no option to sort potential employers by their trustworthiness.

Reputation scores are typically aggregations over all experiences and thus represent the general or average opinion across all system participants. However, as we have argued, feedback is subjective and biased by the feedback provider’s expectations and preferences. Hence, when computing a trust score from available feedback, some experiences, namely those from parties that provide ratings that are similar those we would provide in the same situation, are more valuable. Trust systems account for that fact by individually and selectively aggregating feedback to a subjective trust value. There are several strategies that achieve this. Some systems, e.g. [Mar94], solely rely on direct experiences an individual has made and simply do not use second-hand information. Though this relieves us from the task of identifying relevant feedback, it confronts us with the problem of feedback scarcity. Other approaches use second-hand information in addition, but give a higher weight to own experiences. Often researchers, use techniques similar to those applied in collaborative filtering to assess the usefulness of potential feedback providers. For that purpose, they compare the feedback an entity has provided for a certain service with their own experience when actually using this service. The outcome of this comparison indicates the

[4]www.mturk.com

usefulness of a feedback provider and is either used to adjust (e.g. [ARH00]) or to weight (e.g. [WV03]) second-hand experiences accordingly. *Sabater et al.* [SS02] proposed a substantially different approach that analyses the social relationships between system participants to assess their usefulness as experience providers. Usually, the term referral trust is used to indicate a party's usefulness as feedback provider. As the name suggests, it is typically not distinguished whether a feedback provider is not useful, because his taste is different or because he provides dishonest feedback. However, those strategies are only applicable in scenarios, where two parties interact repeatedly. In an e-commerce setting this is typically not the case. To deal with that problem, *Wang et al.* [WV03] proposed to directly compare the internal trust values that two parties assign to other members of the community to assess their similarity when judging other entities. However, this approach raises serious privacy issues.

As has been emphasized by the trust and reputation systems community, trust is not just subjective, but is also related to a certain context or scope [5] [Mar94, ARH97, GS00, Mui02, Sab05, MMS05, JHP06, JIB07]. To say it with the words of *Jøsang et al.* [JHP06], in a trust relationship "the trusted party is relied upon to have certain qualities, and the scope is what the trusting party assumes those qualities to be." For instance, I may trust an entity when purchasing a bottle of wine, but not when buying a car. As Sabater et al. [SS01] mentioned, those qualities could be explicitly agreed upon in a contract, but they could be also qualities the trusting party implicitly assumes the trusted party to have. However as already said, contracted aspects are typically implicitly assumed and not explicitly represented. As *Jøsang* [Jos08] indicated, this is particularly unsatisfying in light of the fact, that a non or partially fulfilled contract is usually associated with some sort of harm to the trusting party. Therefore, it would be desirable to describe more precisely what may potentially go wrong and to what degree.

Though many authors acknowledge the context-dependent nature of trust, little work has actually been done in developing sophisticated multi-context trust models. Often, trust systems are single context systems or deal with a finite set of scopes. For example, it is common to distinguish between referral trust and functional trust, where the first refers to an entity's trustworthiness as a feedback provider and the latter to its trustworthiness as a service provider. Other approaches consider more than two trust categories [ARH97, ARH00, EC01]. However, contexts are typically not explicitly specified and are treated independently, i.e. trust values are maintained per context. The latter point is particularly critical, since consumer feedback is usually scarce and even scarcer per context. Moreover, contexts may be multitudinous. The mentioned approaches do not address the question of how and under which circumstances experiences made in one context might be used to infer about a party's trustworthiness in another context and how context can be precisely described.

Some attempts have been made to answer those questions. Among the first was *Marsh* [Mar94]. He introduced the notion of situational trust and remarked that experiences made

[5] In the literature, among others, the terms trust context, trust scope, trust purpose, subject matter and trust category are synonymously used.

in one situation may be used to infer trust in another, in case those situations are similar or identical. However, he did not exactly determine what he means with situation and similarity.

Mui et al. [MMA$^+$01, Mui02] proposed to describe context as a set of values for a well-defined set of attributes. He noticed that the interest matching approach described in [KM01] may be used to infer about trust relationships referring to different contexts, but did not detail how this could be done.

The REGRET-system suggested by *Sabater et al.* [SS01, SS02] supports contracts that allow to specify and rate various aspects of an interaction. Those atomic aspect ratings may then be combined to obtain judgments for more complex compound aspects in a user-specified way. For instance, a person might be a good seller, if delivery date, product price and product quality were as promised. The focus of the work is on how to build more complex contexts based on atomic contexts, but does not answer the question of how experiences referring to one contract may be used to infer about an entity's behavior in the context of another contract.

Wang et al. [WV03] model trust in the domain of file sharing using a naive Bayesian network. The root of the network represents the overall trust an entity has into another party, while the leaves represent different trust aspects such as the quality or type of a downloaded file. A provider's trustworthiness in a certain context, e.g. with respect to a certain file type and quality, may then be determined by calculating the corresponding conditional probability. For example, a user might be interested in the probability that a party will be trustworthy in providing high quality music files. The described approach is very flexible and effective and allows to use experiences made in one context to infer trust in another context. Unfortunately, it assumes that contexts can be described by a fixed set of attributes and that those attributes are independent.

As part of their attempt to build a general model of trust relationships *Zhao et al.* [ZVB06] define trust context (scope label in their terminology) as a combination of two sets. These are a set of conditions for a certain trust relationship and a set of properties, i.e. the privileges the trust relationship offers to the trusted party. They also define three rules that allow to compare different contexts according to their strictness. However, they do not indicate how those rules may be used to infer about trust relationships referring to different contexts.

Toivonen et al. [TLU06] model trust scope using two sets of attributes, namely a set of quality attributes representing information that are essential for deciding whether to trust or not, and a set of context attributes, that represent additional information. Whether a certain attribute is a quality or a context attribute may differ from scope to scope. For instance, when considering the trustworthiness of a network component, the fact that the component provides encrypted communication may be a quality attribute in the context of a payment application, but a context attribute in a game application. The authors propose to use quality attributes to calculate a trust score using one of the existing trust computation methods and to use context attributes as additional information that either increases or decreases this value by a certain amount. The specific amount is defined by a parameterized function, whose concrete instantiation is user-specific. Toivonen et al. use knowledge about the ontological relationships between context attributes to estimate function parameters that are unknown for a certain user. Though this is an interesting approach, its understanding of trust differs from ours in that it does not evaluate trust based on ratings of a community, but based on certain properties of the trusted party.

Jøsang et al. [JHP06] developed a consistency criterion for trust scopes in trust networks. They argued that a transitive trust relationship between two entities that are connected by a sequence of trust relationships can only be derived, if "there exists a trust scope which is a common subset of all trust scopes in the path. The derived trust scope is then the largest common subset."

Nobarany et al. [NHC08] proposed that people may tag expressions of trust. Those tags may then be used to evaluate trust on a per tag basis. Though tagging provides a simple, but flexible mechanism to describe the context of a trust relationship, it lacks the semantics that is necessary to infer about trust relationships among different contexts.

Recently, *Ries* [Rie09] argued that trust computation in Bayesian trust models may become context-aware by changing parameters of the computation algorithm, such as initial trust value, aging factor, etc., depending on the context. However, they neither indicate how to represent context, nor how to perform inferences between different contexts.

As we have seen, trust and reputation systems have to collect and share user information about past interactions to evaluate the system participants' trustworthiness. Additionally, they require that those data can be related to an identity of an individual. Though, in contrast to collaborative filtering systems, the shared interaction data do not directly contain preference expressions like "I liked this aspect of service A to degree x", they still provide some of those information indirectly, for instance by telling other parties with whom and how often a person ran into a transaction. To mitigate that problem it was proposed to use pseudonyms instead of real world identities. However, that raises the problem that individuals may easily switch pseudonyms and thus may misbehave without paying the consequences. *Friedman et al.* [FR01] argued that trust in pseudonyms is still possible, if starting with a new pseudonym is tied to some costs. This could be achieved for instance by letting newcomers start with a low reputation or by letting participants pay for a new pseudonym. However, even when using pseudonyms, we still risk loosing anonymity, when pseudonyms and their associated user profiles can be matched to the corresponding real world identities. This may for instance happen when two entities communicate with each other. In [KTR05], *Kinateder et al.* proposed a solution for that problem. However, profiling on the basis of pseudonyms is still possible, but necessary to establish trust relationships. In general, the more information we have, the more accurately trustworthiness can be assessed, but the less privacy is left to the entities. Referring to this fact, *Seigneur et al.* [SJ04] argue depending on the outcome of a trust relationship, trusted parties are willing to trade part of their privacy for increased trustworthiness to different degrees. Hence, an adaptive approach to make this trade-off is required. As a solution the authors propose a model based on the linkability of information. In the considered scenario, users may have several pseudonyms, which in turn may have several pieces of information associated with them. If the information available under one pseudonym is insufficient to evidence an entity's trustworthiness, this entity might allow to link more information to this pseudonym in order to improve its trustworthiness and thus its chance to run into a transaction with a desired partner. Additionally, pseudonyms may be linked with each other. This careful disclosure of links enables a fine-grained trade-off process. However, the proposed mechanism solely deals with privacy issues concerning the trusted party, but not with privacy issues that arise, when providing experiences about service interactions. In particular, it remains open how the quality of provided feedback information itself can be adjusted to compromise between a user's privacy issues and the accuracy of the trust assessment.

6.2.4. Experience-Based Service Provider Selection

Evaluating a service's performance based on experiences that consumers made in former service interactions has a long tradition in the research areas discussed in the previous sections. Recently, we observe a growing interest to apply similar techniques in the context of service selection. Often experience-based techniques are used to complement traditional service selection approaches that identify services matching a user's functional requirements by comparing machine readable request and offer descriptions. Typically experience-based approaches assume that a list of services functionally matching a certain request is already available [MP05, WLH07, VHA05, VPHA07, MS02, MS05, Ker06]. The outcome of an experience-based evaluation of these services' performance is then used as an (additional) ranking criterion. Different strategies have been developed to perform this evaluation.

A considerable share of work [VHA05, VPHA07, MS02, MS05, CBGS07, CBGS06, WLH07, YS02, SPg$^+$07] investigated experience-based techniques to evaluate a service's non-functional, i.e. Quality of Service (QoS), properties such as throughput, availability or trust. In this context, only a few approaches [VHA05, VPHA07, MS02, MS05, SPg$^+$07] consider detailed consumer feedback that judges those quality aspects. The collected feedback is utilized to predict a value indicating a service's overall conformance to the advertised QoS in future transactions. For that purpose, different techniques such as linear regression [VHA05, VPHA07] or weighted mean [MS02, MS05] have been employed. *Sensoy et al.* [SPg$^+$07] also consider functional service characteristics. They use a weighted average of the provided qualities as an indicator for the quality that will be provided in future transactions, but also propose to use statistical classification to predict whether a service will be satisfying or not in the future. In this context, they do not exclusively use feedback provided for the considered service, but also utilize feedback made in other, similar, service contexts. Similarity is determined by comparing the service demands that led to a service usage with the subsequent feedback. The similarity measure is customizable and described via SWRL-rules. Unfortunately, their approach requires explicit sharing of service requests to determine relevant feedback. In our mind this divulges too much personal information of the consumers.

The mentioned approaches typically require a user to rate a set of quality attributes. Often the number of the required judgments is considerable. Neither the user's willingness to provide those ratings, nor his ability to do so are considered. However, as already mentioned most of the solutions exclusively consider QoS aspects, which form a subset of service attributes that in general is automatically measurable and where commonly agreed upon measuring methods exist. Hence, monitoring components may and often do automatically examine service quality and thus take on most of the user's burden. This also justifies the assumption that underlies most of the solutions (all except [MP05, WLH07]), namely that consumer feedback is objective. However, we argue that only accounting for non-functional aspects, when judging a service's performance, is not sufficient. Functional service characteristics have to be considered as well. Since those aspects usually cannot be measured automatically, such an extension would involve explicit user-provided feedback and hence mechanisms that cope with the subjectivity inherent in this type of feedback. Moreover, in our mind, (central) monitoring components that collect consumer information including entire requests are problematic in terms of privacy protection. Another point is, that the set of attributes to be rated is typically fixed per service. Since knowledge about relationships between attributes is not available, judgments referring to one service cannot be used to infer knowledge about another service's performance. This is a viable solution as long as we can expect to have a sufficient number of observations per service and thus do not have to rely on feedback referring to similar services of a provider to make accurate predictions of his services' future behavior. However, in an e-commerce setting this assumption no longer holds. *Vu et al.* [VHA05, VPHA07] address part of this problem by defining hierarchical relationships between quality attributes that are considered when integrating judgments referring to different quality attributes. However, they do not consider the context of an attribute judgment, e.g. whether the execution time was judged for a weather service or for a service that offers the latest stock prices.

However, most of the work does not consider detailed consumer feedback at all and relies on single ratings judging the outcome of a service as a whole. As already argued, this will provide only a rough picture of a service's quality. Three different techniques have been applied in this context. *Manikrao et al.* [MP05] and *Wang et al.* [WLH07] for example, proposed to use collaborative-filtering-based approaches in addition to traditional matchmaking to cope with QoS aspects that are not objectively perceivable. While Manikrao et al. employ an item-based approach, Wang et al. propose to use user-based collaborative filtering. More specifically, Manikrao et al. presented a solution that solely relies on own ratings to estimate a user's rating for a certain service. Rating estimation is done by computing the weighted mean of all ratings the user provided for similar services. The weights reflect the degree of similarity of a particular service to the considered one and is computed by comparing the rating profiles of the services. Wang et al. use the weighted average of the ratings that different users provided for a certain service as an indicator for its quality. The weights indicate how similar the rating provider's ratings are to those of the user who provided the ratings. While Manikrao et al. deal with the issue of integrating feedback made in different service contexts, Wang et al. tackle the problem of integrating subjective feedback. Unfortunately, none of the authors considers both issues. Moreover, as collaborative-filtering-based approaches, the mentioned solutions inherit all the disadvantages those types of approaches have. For instance, item-based recommenders, such as proposed by Manikrao et al., require that a user has rated a sufficient number of similar services to achieve a good prediction quality. User-based approaches, such as used by Wang et al., require that the users that should be compared co-rated a sufficient number of services. We also argued already, that two users who produce similar ratings in average, do not necessarily provide similar ratings for a particular service. However, the other way round, there may be users who would produce similar ratings for the considered service, but do not provide similar ratings in average and thus are not considered valuable for the rating estimation. Hence, feedback could have been exploited more effectively.

In addition to collaborative recommendation methods, techniques known from trust and reputation systems have been applied to service selection [CBGS07, CBGS06, BHOC07, YS02]. *Caballero et al.* [CBGS07, CBGS06] define trust for a certain service as the product of the promised degree of QoS satisfaction (the matching degree obtained when comparing the request description with the service advertisement) and the extend to which this promise has been kept in former transactions. For that purpose, the authors propose to describe requests (tasks in their terminology) and service offers in terms of attribute sets and present several ways to compare them to compute a matching degree. The degree of actual user satisfaction is determined using consumer feedback, where the authors borrow from existing trust system solutions, such as REGRET [SS01] and SPORAS [Zac99]. Caballero et al. also consider the problem of having no or little experiences with a given service and suggest to additionally utilize experiences that users made with similar services in those situations. The degree of similarity is determined by comparing the attributes in the corresponding service descriptions using Tversky's measure [RE03, Tve77]. However, as it is the case in the work of Sensoy et al. [SPg$^+$07], this solution requires consumers to share task descriptions, which is not desirable from a privacy perspective. Moreover, the approach is limited in that a service's quality is judged with a single rating. Detailed consumer feedback is not considered. In addition, we believe that the suggested approach could be improved when considering more expressive service descriptions.

In contrast to the discussed solution, *Billhardt et al.* [BHOC07] as well as *Yu et al.* [YS02] propose purely trust-based solutions to service selection. Both take single consumer ratings as input and use them to assess a service provider's trustworthiness. While Billhardt et al. choose a weighted mean approach to combine feedback, Yu et al. propose to use Dempster-Shafer theory of evidence for that purpose. Billhardt et al. also consider ratings made for similar services when evaluating a service provider's trustworthiness. Service similarity is determined by the closeness of the services' categories within a service taxonomy. However, both approaches do not account for the subjective nature of feedback. Moreover, they make not clear how trust is related to the promised service quality. Finally, it should be mentioned that among all approaches considered in this survey, only Caballero et al. as well as Yu et al. provide some kind of a confidence measure for their predictions. As common in trust and reputation systems, Caballero et al. propose a measure based on the amount and variability of experiences, while the belief-based solution of Yu et al. naturally provides a notion of confidence (see Section 6.2.3). However, to the best of our knowledge, none of the approaches mentioned in this section address the issue of how to effectively and intuitively present experience-based information as well as the systems confidence about them.

The approaches mentioned so far, have in common that they leverage consumer feedback that is provided after the service invocation, to evaluate a service's ability to perform a certain task. The service selection strategies proposed by *Kerrigan* [Ker06], *Kokash et al.* [KBD07] and *Averbakh et al.* [AA09, AAS09] differ from those in that they elicit feedback during the (manual) service selection phase and later use it to improve service retrieval. Feedback is either directly elicited by allowing consumers to judge the results provided by the semantic matchmaker [AA09, AAS09] or indirectly by learning from a consumer's service selection decisions in the past [Ker06, KBD07]. All approaches presume a semi-automatic selection process, where consumers manually choose a service from a list of suitable service offers provided by a service matchmaker. For each consumer-goal-preference-constellation, Kerrigan records how often each available service was chosen. Those information are then utilized to rank available services based on their frequency of usage in the context of a given goal-preference-constellation. In our opinion, this approach suffers from a main drawback. It does not consider whether the user was finally satisfied with his selection decision or not and thus the computed service ranking might not reflect the actual suitability of the available services for the considered goal. Moreover, eliciting direct consumer feedback before the actual service invocation presumes that users are able to decide about the suitability of a service being only provided with its formal offer description, which is unlikely. However, the solution accounts for the subjectivity of selection decisions by considering not just the user's goal, but also his preferences when ranking services. Unfortunately, it provides no means for comparing different goals or preferences. This would be desirable, since it is very unlikely that a given task-preference-combination has been already posed in former transactions, whereas similar combinations are more likely to occur. Similar to Kerrigan, *Kokash et al.* protocols information about consumer requests and subsequent manual selections. In opposition to Kerrigan, they consider whether a service was successfully invoked or not and allow for optional consumer feedback. Given a consumer request, the system proposes a suitable service by leveraging this interaction data. The implemented strategy borrows ideas from case-based reasoning. First, the given request is compared with previous requests in the records. Finally, the service chosen for the most similar request is recommended. In this context different similarity measures might be implemented. Unfortunately, the authors just present a solution, where service requests are textual descriptions of the required service functionality and define a similarity measure based on the comparison of term-frequencies. It would be interesting to see how their approach could be enriched by using semantic service descriptions. Kokash et al. also do not explain how their approach could benefit from using direct feedback. Finally, the design of the approach does not allow to acquire detailed feedback and does not consider user preferences. *Averbakh et al.* suggest a hybrid matchmaking approach that considers several matching functions to assess the suitability of service offers. In particular, they use the matching functions provided by the OWLS-MX [KFS06] service matchmaker as well as user feedback in terms of ratings. Those ratings are collected during the process of service selection and indicate the suitability of a retrieved service. The consumer rating for a given service request and offer is determined from existing feedback that was contributed by users that share a similar taste with the target user. Similarity is calculated using standard collaborative filtering techniques. Since as in [Ker06] feedback is acquired before the actual service invocation, this approach inherits the disadvantages that are associated with this fact.

6.2.5. Summary and Open Research Issues

Table 6.1 summarizes the results of our analysis. The entries in the table indicate whether there exists at least one approach in the considered research area that partially (○) or completely (+) fulfills the considered requirement. We also indicate, if none of the approaches in a domain fulfills a certain requirement (-).

As can be seen in the table, the elicitation of multi-aspect consumer feedback is considered in all of the analyzed research areas, which emphasizes its importance. However, more flexible and adaptive mechanisms to elicit and describe this type of feedback, particularly when used across domains such as in Semantic Web Service retrieval, are required. Especially, the question of how to describe multi-aspect feedback meaningfully has been hardly considered. Another issue that has been only partially addressed is that of eliciting comprehensive, appropriate and meaningful feedback. Just a few solutions from the field of product reviews consider the problem of adapting the elicitation process to the user's willingness to provide feedback by offering more flexible tag-based feedback strategies.

Though privacy issues have been discussed in the considered research areas, feedback mechanisms that allow to flexibly trade off the prediction accuracy and the quality of the shared feedback information have not been devised.

The context-dependent nature of consumer feedback and its context-aware usage have been addressed quite well in collaborative filtering systems. It is desirable to apply and adjust the techniques that have been developed in this area to allow for feedback-aware Semantic Web Service retrieval and to extend them by leveraging the powerful semantic description mechanisms that are offered by Semantic Web Service technology. In particular, more attention has to be paid to the question of how to effectively use experiences made in one context to infer knowledge about a service provider's performance in another context.

The strengths of solutions from the fields of trust and reputation systems, collaborative filtering systems and experience-based service provider selection are the effective exploitation of consumer experiences and the accurate prediction of a service's future performance. Approaches from the domain of product reviews lack those capabilities. This is due to the fact, that their predictions are typically based on simple heuristics and non-semantic techniques that are applied to just a small fraction of the elicited data. On the other hand, tag-based solutions from this area are far more flexible with respect to the type of services and service aspects for which detailed feedback can be acquired than those from other

	Exp.-based Service Provider Selection	**Trust & Reputation Systems**	**Collaborative Filtering**	**Product Reviews**
Feedback Elicitation				
elicits appropriate feedback (R F.1)	+	+	-	+
elicits comprehensive feedback (R F.1)	○	+	-	-
elicits detailed feedback (R F.1)	+	+	+	+
elicits meaningful feedback (R F.1)	○	○	○	-
adjusts to the user's willingness (R F.2)	-	-	-	○
Feedback Propagation				
shares information only in required quality (R F.3)	-	○	○	-
quality of shared information is adjustable (R F.3)	-	○	-	○
Performance Prediction				
explores feedback effectively (R F.4)	○	○	+	-
considers the request context (R F.5)	○	○	+	○
considers the service context (R F.5)	○	○	+	-
provides a confidence measure (R F.6)	+	+	+	○
Feedback Presentation				
makes aware of the risk assoc. with a service exec. (R F.7)	-	○	-	-
adjustable to different risk attitudes (R F.8)	-	-	-	-

Table 6.1.: Requirements to the elicitation, propagation, usage and presentation of feedback information

fields. The challenging question is whether it is possible to integrate ideas from both types of approaches into a solution that is both, flexible and effective.

Another aspect we noticed is that most of the solutions aiming at the provision of experience-based support for service selection simply adopt techniques from trust and reputation systems or collaborative filtering systems to complement standard matchmaking and ranking approaches. They typically do not really integrate and adjust existing solutions to meet the special requirements imposed by Semantic Web Services. In addition, we found that though solutions for some of the identified problems exist in other research areas, they have not yet been adopted in the context of Web Service selection. This is especially true for issues such as modeling feedback context and designing flexible description techniques for multi-aspect feedback. While the first aspect has been considered in the context of trust and reputation systems and in collaborative filtering systems, the latter has been successfully addressed in the domain of product reviews.

The issue of how to effectively and intuitively communicate feedback-derived information about the risk that is associated with the execution of a service has not been sufficiently addressed in any of the domains. It also remains open how those information can be presented in a way that accounts for different risk attitudes. These findings are particularly surprising in light of the fact that some of the considered research areas yield powerful and effective prediction algorithms whose full potential remains unused if the information that are produced by them are not appropriately announced to the user.

The most pressing research questions that remain open are:

- How can multi-aspect feedback be described in a flexible and meaningful way that allows for its effective exploitation to make accurate predictions about a service's future performance?
- How can consumers be flexibly supported in providing appropriate, comprehensive and meaningful multi-aspect feedback?
- How can the quality of feedback be reduced, while still allowing for its effective exploitation to make accurate predictions? In particular, how can feedback providers be enabled to flexibly trade off the prediction accuracy and the quality of shared feedback?
- How can the context in which consumer feedback is provided be described and how can those information be leveraged to make effective predictions about a service's future performance? In particular, how to effectively use experiences made in one context to infer knowledge about a service's behavior in another context?
- How can feedback-derived information be effectively communicated to make potential service consumers aware of the risk that is associated with the execution of a service and how can this be done in a way that is adaptive to to different risk attitudes?

6.3 The Feedback Mechanism - Basic Idea

In the subsequent sections, we will introduce a collaborative feedback mechanism that allows to predict a service's performance in future interactions. Since performance judgments are subjective, i.e. dependent on the expectations and preferences of the judgment provider, and since misbehavior in service provision might depend on the type of service that was offered, the performance prediction is done with respect to the given service requirements, i.e. the service request that has been posed, and with respect to the offered service (Requirement F.5). The prediction is based on judgments provided by consumers that interacted with the considered service's provider in the past. For that purpose, consumers may judge the quality of a service interaction after service invocation by providing attribute-specific ratings for selected aspects of the interaction. To foster high-quality feedback, the system supports the user in that process. In particular, it ensures that elicited feedback is comprehensive, meaningful and appropriate in the context of the considered service interaction (Requirement F.1). It also accounts for the consumer's willingness to provide judgments (Requirement F.2). This is done by suggesting a set of service aspects for judgment that is likely to be judged by the consumer (Sections 6.4 and 6.5). The elicited judgments are propagated to other service consumers and are leveraged to infer knowledge about a service provider's future behavior with respect to those aspects (Section 6.6, Requirement F.4). Thereby, the user can flexibly trade off the performance prediction accuracy and the detailedness of the shared feedback information (Requirement F.3). An appropriate confidence measure for the predictions will be derived as part of the evaluation of our approach (Requirement F.6, Section 9.4.6). If desired by the user, feedback-derived information can be used as a supplementary criterion when ranking semantically matching services and is visualized in addition to the ranked service results (Section 6.7, Requirement F.7). Service consumers may personalize the ranking mechanism's outcome according to their individual risk attitude (Requirement F.8).

6.4 Consumer Feedback

In this section, we will demonstrate how semantic service descriptions can be leveraged to create feedback that is detailed, meaningful, comprehensive and appropriate for characterizing a considered service interaction.

6.4.1. Creating Appropriate, Comprehensive and Meaningful Multi-Aspect Consumer Feedback

Our approach is based on the elicitation of multi-aspect consumer feedback, which means that service consumers provide attribute-specific, i.e. detailed, judgments for a service interaction (Requirement F.1). In the context of our work, we presume that those judgments are numerical ratings taking values from the interval $[0, 1]$. Thereby, a rating of 0 means not acceptable at all and a rating of 1 means completely acceptable. Ratings from $(0, 1)$ indicate different grades of acceptance.

By our approach, it is ensured that judged service aspects are appropriate in the context of a considered service interaction (Requirement F.1). As we will see, this is achieved by recruiting potential judgment targets from the service aspects, i.e. the attributes, that are covered by the user's request model. This is reasonable, if we assume, that the request model that led to a service judgment covers all service aspects that are important to the consumer. However, the feedback mechanism is independent of the requirements elicitation procedure and as such is applicable to manually created service requests as well.

Assumption 6.1. (**Accuracy of the request (model)**) *The request (model) that led to a service judgment accurately reflects the requester's actual service requirements and preferences.*

This premise is even stronger than necessary at the moment, but is required later on. It is certainly not true at the beginning of the incremental requirements elicitation process that has been introduced in Chapter 5, but should ideally be valid at the time a service is invoked (and finally judged). Otherwise, the user would have missed to specify a requirement that is crucial to him. As shown in Chapter 8, our user modeling component effectively supports the user in specifying his service requirements and ensures in fact that all service aspects that are important to a user are covered in the system-maintained request model.

Potentially, all service aspects that are considered in the request (model) are relevant to the service consumer and thus are appropriate judgment targets that might be rated. More specifically, a judgment referring to a certain aspect of the request (model) indicates the performance of the invoked service with respect to this aspect as experienced by the judgment provider. In conformance with the semantics of DSD, judgments referring to intermediate, i.e. non-leaf, attributes of the request (model) indicate the service's aggregated performance with respect to those attributes' child aspects (and thus with respect to all aspects covered by the request (model) subtree rooted at the judged aspect). Consider for example the request (model) depicted in Figure 6.2. By providing a rating for the service aspect *productType*, a judgment provider indicates the aggregated performance of the invoked service with respect to the service aspects *battery*, *style*, *phoneType* and *color*.

Letting judgments reference service aspects that are covered in the request (model) is advantageously. By coupling (parts of) the consumer's semantically described requirements with the provided judgments, we provide judgments with a well-defined and commonly agreed upon meaning (Requirement F.1). This is due to the fact that service aspects covered in the request (model) refer to concepts in the service ontologies, which define not only valid service aspects and valid constraints on them, but also valid relationships among them. This ensures that there is an agreed upon meaning of the judgments provided by different service consumers and thus allows for the comparability of judgments provided by different users and for different services. This in turn, is an essential prerequisite for enabling the usage of feedback made under one circumstance, i.e. in one context, to infer about a service provider's performance in another situation or even when providing another service (Requirement F.1).

However, in order to be able to do so, we also need to make sure that the request (model) context in which judgments were made is recorded (Requirement F.1). Moreover, we have to ensure that the provided judgments are comprehensive (Requirement F.1). The challenging question is how the latter can be achieved while at the same time still being flexible in the choice of the service aspects to rate. We propose the concept of a *feedback structure* to deal with those issues.

Definition 6.2. (**Feedback structure**) *A feedback structure is a subtree of the request (model) tree, whose leaves correspond to the service aspects that have to be rated by the user. In contrast to the request (model) tree from which it is derived, a feedback structure does not contain any preference information (e.g. connecting strategies) and no direct conditions.*

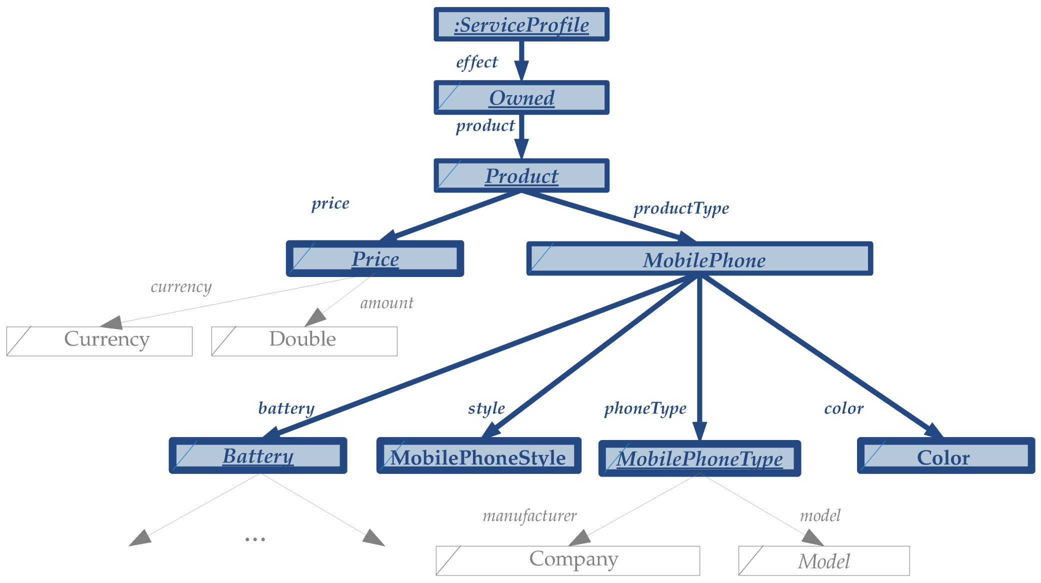


Figure 6.2.: Feedback structure (blue part) and the request (model) where it originates from (blue and gray parts)

Consider the example request (model) depicted in Figure 6.2. The blue part of the tree indicates a possible feedback structure for that request (model), where the aspects *price, battery, style, phoneType* and *color* have to be judged by the consumer. Restricting the judgment space to feedback structure leaves guarantees, that no service aspect is judged twice. This would for instance happen, if a consumer judged both, *productType* and *phoneType*, since the *productType*-judgment indirectly judges *phoneType*. We prohibit multiple judgments for single service aspects, since they are likely to cause inconsistencies. To assure that the provided feedback is comprehensive, the request (model) subtrees rooted at the feedback structure's leaves should cover all leaves of the request (model) tree. This guarantees that all service aspects considered in the request (model), i.e. all aspects that are relevant to the service consumer, are either directly or indirectly (by providing an aggregated rating) judged (comprehensiveness of feedback, Requirement F.1). The feedback structure depicted in Figure 6.2 fulfills this requirement and thus is valid. Omitting, e.g., the aspect *phoneType* would result in an invalid structure, since the aspects *phoneType*, *manufacturer* and *model* would not be judged.

Definition 6.3. (**Validity of a feedback structure**) *Let r be a request (model) and fs a feedback structure derived from it. The feedback structure fs is valid, if and only if the request (model) subtrees of r rooted at the request (model) nodes that correspond*[6] *to the feedback structure's leaf nodes cover all leaves of the request (model) tree r.*

Note, that we are still flexible in the choice of the feedback structure and hence in the choice of the attributes that have to be judged by the service consumer. For example, a consumer might provide a single rating for *productType* instead of judging *battery*, *style*, *phoneType* and *color* separately. Providing just a single overall rating to judge a service interaction as a whole is also a valid option when using this scheme. In this case, the feedback structure comprises of a single node corresponding to the request (model)'s root node. Obviously, there is a trade-off between the detailedness and thus the number of the provided judgments, the rating effort and the user's privacy concerns (Requirements F.1, F.2 and F.3). The more detailed the provided judgments are, the higher is the quality that we can expect of the performance prediction. However, providing detailed judgments imposes a higher judgment effort on the user and reveals more (personal) information about his requirements. A strength of the proposed solution is that, by letting judgment providers freely choose the feedback structure to judge, it enables them to make this compromise according to their personal judgment preferences and privacy restrictions (Requirements F.2 and F.3). However, the effectiveness and flexibility of the approach strongly depends on the quality of the underlying ontology. This is attributed to the fact, that the set of valid feedback structures for a given request (model) is determined by the ontological concepts that can be used to describe service requirements as well as by valid relationships among them.

Besides ensuring the appropriateness and comprehensiveness of judgments, a feedback structure contains most of the information that is required to effectively utilize the provided judgments. In particular, it encodes the paths of the judged service aspects.

Definition 6.4. (**Path of a service aspect**) *Let a be a service aspect that is considered in the request (model) r and let $(a_1, a_2, \dots, a_n)$, $a_1 = root(r)$ and $a_n = a$, be the sequence of nodes, i.e. service aspects, that lie on the request (model) tree r's path from its root node $root(r)$ to a. We define the path $path_r(a)$ of service aspect a in r to be $[name(a_1), type(a_1)].\ [name(a_2), type(a_2)]. \cdots .\ [name(a_n), type(a_n)]$, where $name(a_i)$ is the name of the service aspect a_i and $type(a_i)$ is the type of its target set in r.*

[6] The corresponding node or attribute of a request (model) node is the node of the feedback structure that lies on the same tree path.

As an example, consider the service attribute *color* in Figure 6.2. Its path is $[\textit{effect}, \textit{Owned}]$. $[\textit{product}, \textit{Product}]$. $[\textit{productType}, \textit{MobilePhone}]$. $[\textit{color}, \textit{Color}]$. For the sake of clarity, we omit the name of the referenced request (model), if it is clear. We also omit target set types, if they are not relevant and also (partially) omit the names of the preceding attributes, if the resulting path name of the considered attribute is unique. That is, the expressions *effect.product.productType.color* and *color* are also valid notations for *color*'s path. Recording the paths of the judged service aspects is important, since they provide information about the context in which the judgments were made (Requirement F.1). Consider for instance two price judgments, referring to different service interactions, e.g. an interaction with a ticket booking service and an interaction with a wine selling service. Without having knowledge about the judged aspect's paths, the judgments are meaningless, since we do not know in which context they have been made and thus cannot decide whether they are comparable or not. However, recording just the paths of the judged service aspects is not sufficient, since this means that we would have omitted valuable information about the entire request (model) of the judgment provider, in particular information about his preferences. As we will see, those are required for determining relevant judgments, i.e. those that have been made in a similar context, when leveraging feedback from past service interactions to infer about a service provider's future performance in another context (Section 6.6). However, sharing complete requests (request models) with other system participants is out of question, since this would divulge a massive amount of sensitive personal information such as the consumer's preferences and requirements. Instead, we propose to share indirect information about the judgment provider's request (model) context given by the (partial) matching results for the request (model) and the available and matching offers. Note, that in case of a request model, those matching results are expected matching degrees (cf. Section 5.6).

Definition 6.5. (**Indirect context information**) *Let r be the request (model) of the judgment provider and O_M the set of service offers that matched to r. Presume that one of the services described by the offers in O_M was invoked and finally judged. Let fs be the feedback structure on which the judgment provider's judgments are based. We refer to the set $\{MR_{a_i}(r, O_M) | a_i \in A_{fs}\}$ as indirect context information, where $MR_{a_i}(r, O_M) = \{mv_{a_i}(r, o) | o \in O_M\}$ is a set containing the (partial) matching degrees of the request (model) subtree rooted at the service attribute a_i with the corresponding parts of the offers in O_M and A_{fs} is the set of attributes considered in fs.*

In doing so, distributing service requests (request models) and thus providing easily accessible personal information can be avoided (Requirement F.3) while at the same time sharing information that are sufficient to effectively determine the relevance of feedback items. As we will demonstrate in Section 9.4, collaborative filtering techniques can be successfully applied to achieve this. Details on this approach will be presented in Section 6.6.

Please note, that the required partial matching results are computed anyway during the matchmaking process, when determining suitable, i.e. matching, service offers based on r. Hence, determining indirect context information causes no additional computational effort.

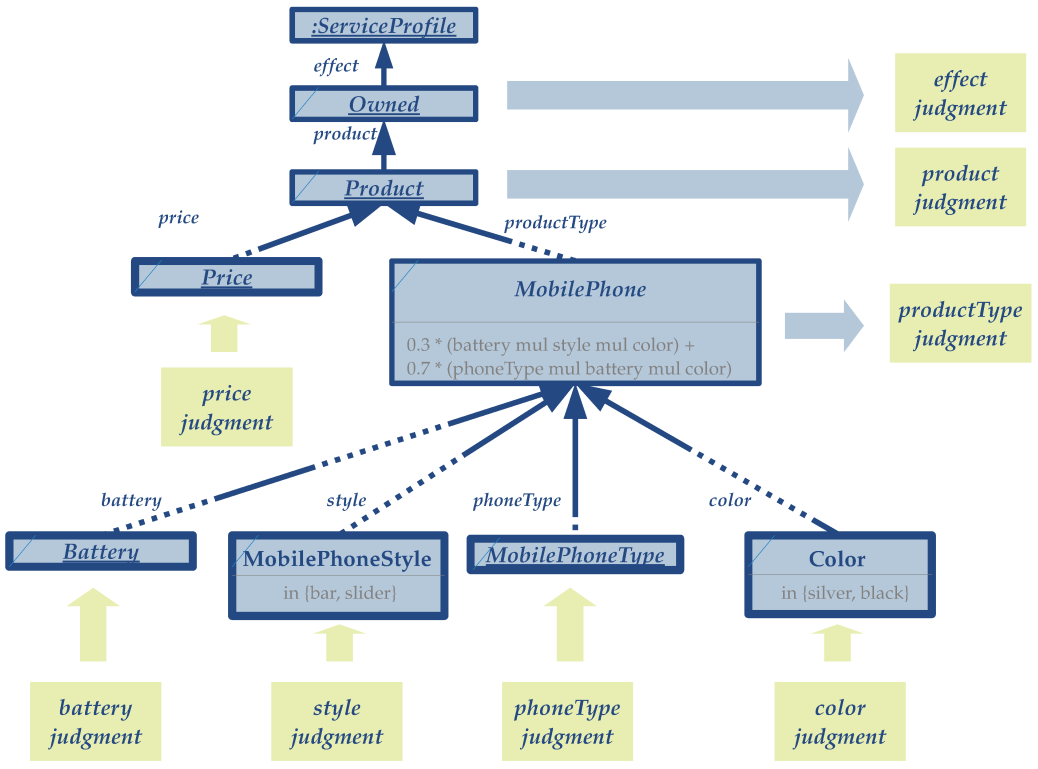


Figure 6.3.: Judgment aggregation using the semantic matchmaker

Once a consumer provided ratings for all service aspects corresponding to the leaves of the selected feedback structure, the semantic matchmaker is used to determine aggregated ratings for all internal nodes of the feedback structure, i.e. all non-leaf aspects. For that purpose, it is supplied with the judgments that have been provided for the feedback structure leaves as matchmaking input for the judged attributes, i.e. as predefined matching values for those aspects, and performs standard matchmaking based on those values. Assuming again, that the consumer's service request accurately reflects his actual requirements (Assumption 6.1), the resulting judgments will correspond to the aggregated judgments the user would have provided for the feedback structure's internal nodes, if asked. In fact, it is likely that the aggregated judgments, if provided by the consumer, would be inconsistent with the ratings provided for their subaspects. As already argued, this is due to the fact that humans are bad at integrating information on different aspects, as they appear in a multi-faceted service interaction. By automatically aggregating attribute-specific ratings, we mitigate that problem. Figure 6.3 illustrates the aggregation procedure based on a sample request.

The feedback structure accompanied with the judgments provided by the consumer as well as the indirect context information are propagated to other consumers as a *feedback item* and can be used to infer knowledge about a considered service's future behavior.

Definition 6.6. (**Feedback item**) *We refer to the entirety of information that characterize a service interaction as feedback item. Those information are passed to other service consumers to infer knowledge about a considered service's performance in future interactions. They include*

- *the provided judgments,*
- *the feedback structure on which the judgments are based,*
- *the (automatically) aggregated judgments for the feedback structure's intermediate nodes,*
- *the indirect context information related to the service interaction,*
- *the identifier of the judged service, its offer description and the identifier of its provider as well as*
- *the starting date and time of the service interaction.*

Starting date and time of the service interaction are required to discount the influence of older and thus possibly out-dated judgments on the rating prediction[7].

6.5 Effective Elicitation of Consumer Feedback

In the previous section, we discussed how consumer feedback can be described in a way that ensures its comprehensiveness, appropriateness and meaningfulness and identified necessary constraints to it. However, so far we owe to explicate how service consumers can be assisted in actually providing the desired judgment information. In particular, we did not explain how feedback quality can be ensured by accounting for a consumer's judgment preferences, i.e. his willingness to provide certain ratings, when eliciting judgment information and how such a process can flexibly and automatically adjust to different judgment preferences (Requirement F.2). In this section, we will suggest an elicitation mechanism that satisfies those requirements (Section 9.3).

Assume, that given a certain request (model), an appropriate service was selected and invoked and now its performance shall be judged by the consumer. In a first step, we utilize the provided request (model) to determine valid feedback structures as defined in the previous section. Subsequently, the structure that is most likely to be judged by the user, i.e. in the context of the given request (model), fits best to the consumer's personal judgment preferences, is selected and graphically displayed to the user (Figure 6.4). The required knowledge about the user's judgment preferences is learned from his behavior in previous judgment sessions. The presented feedback structure represents a careful compromise

[7]This aspect is not considered in our work, since it has been sufficiently addressed in other research areas, such as trust and reputation systems (cf. [JIB07]).

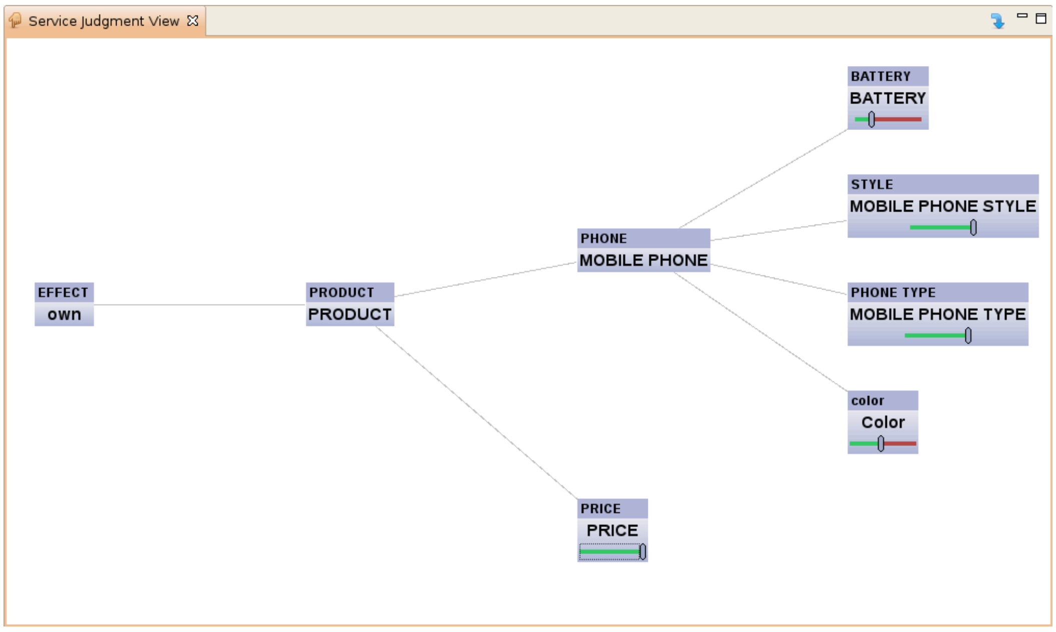


Figure 6.4.: Judgment view

between the consumer's competing judgment preferences and as such typically cannot perfectly meet all his judgment requirements. It is just a recommendation to the user and thus, if required, might be adjusted by him to match his actual judgment needs. This can be done by expanding and/or collapsing feedback structure nodes. Nodes are expanded level-wise by simply clicking on the considered attribute node and collapsed completely upon clicking on an expanded node. Thereby, maximally expanding the displayed feedback structure will result in the attribute structure given by the request (model) tree. As an example consider the feedback structure depicted in Figure 6.4. The user might expand the node *phoneType* to judge its subaspects *manufacturer* and *model*. He might also collapse the root attribute, to provide just a single overall judgment for the service. After having customized the recommended feedback structure according to his needs, the user finally judges all leaf attributes of the structure by providing a rating. This is done by simply moving the slider that is shown for those attributes (Figure 6.4). The default rating is 1.0 ("requirement completely fulfilled"). This is to encourage the user to adjust the rating, if it does not comply with the actual situation. In addition to the slider position, the value of the provided rating is color-encoded in the slider bar. While the fraction of the bar that corresponds to the rating value is colored green, i.e. positive, the fraction that is missing to a rating of 1.0 is colored red. As an example, consider again the feedback structure depicted in Figure 6.4. The user indicated that the service performance with respect to the attributes *price*, *phoneType* and *style* was as required (rating of 1.0), whereas *battery* and the *color* were not as desired (rating lower than 1.0). Once, the consumer submits his judgments, the system takes care of storing all relevant feedback information and session data for future recommendations. In particular, it is recorded which and how many service aspects were judged by the consumer and which request (model) led to the judgment. The acquired information are used later on to identify the feedback structure that is most likely to be judged by the user in future judgment sessions.

As already mentioned, there exist typically numerous valid feedback structures for a given request (model). However, usually they do not fit equally well to the consumer's judgment preferences. Hence, beside a procedure that allows to identify valid feedback structures, a mechanism that assesses the likelihood of a certain structure to be judged by the user and thus enables the identification of the/a feedback structure among the possible structures that best fits to the user's judgment preferences is required. In this context, we like to note that it is not necessary to assess the actual likelihood of being judged for each valid feedback structure. It is sufficient to be able to sort feedback structures according to their likelihood and thus to be able to find the one that is most likely. While the problem of identifying valid feedback structures is addressed in the subsequent section (Section 6.5.1), the aspect of determining a feedback structure's likelihood to be judged is considered in Section 6.5.2.

6.5.1. Determining Valid Feedback Structures

In this section, we describe a procedure that allows to determine all valid feedback structures that can be derived from a given request (model). The algorithm is outlined in Listing 6.1. It recursively constructs the set of valid feedback structures for the request

```
FeedbackStructures determineValidStructures(RequestModelAttribute a) {
    /* add a feedback structure comprising just of your own path */
    feedbackStructures.add(new FeedbackStructure(path(a)))

    /* if this is not a leaf attribute of the request (model) tree */
    if(!a.isLeaf()) {
        /* determine valid feedback structures for the subtrees rooted at a's children */
        for(child attribute a' of a) {
            possChildStructures.add(determineValidStructures(a'))
        }

        /* determine valid feedback structures based on valid child structures */
        feedbackStructures.add(join(possChildStructures))
    }
    return feedbackStructures
}
```

Listing 6.1: Determining valid feedback structures for the request (model) subtree rooted at the attribute a (feedback structures are identified by the paths of their leaf attributes)

(model) subtree rooted at the attribute provided as input. Hence, calling the algorithm with the root attribute of a given request (model) r as an argument will deliver valid feedback structures for the request (model). Thereby, feedback structures are uniquely described by means of their leaf attributes each identified by its path (Definition 6.4). For example, the set $\{product.price, product.productType\}$ of leaf aspects, uniquely identifies the blue-colored feedback structure in Figure 6.5. The feedback structure construction algorithm works as follows. It first adds the superficial feedback structure comprising just of the input attribute itself to the output set. In case of a non-leaf attribute, it then recursively determines valid feedback structures for the subtrees rooted at the children of the input attribute in r. Let a be the input attribute and $FS_r(a_1), \ldots, FS_r(a_n)$ the sets of valid feedback structures constructed for the subtrees rooted at its child attributes $a_1, \ldots, a_n$. The algorithm identifies non-superficial feedback structures for the request (model) subtree rooted at a by determining $FS_r(a) = \{fs_1 \cup \ldots \cup fs_n | fs_i \in FS_r(a_i)$ for all $1 \leq i \leq n\}$. Recursion stops at the leaf attributes of the subtree rooted at the input attribute. Finally, we end up with an output set comprising of all valid feedback structures of the subtree rooted at the input attribute.

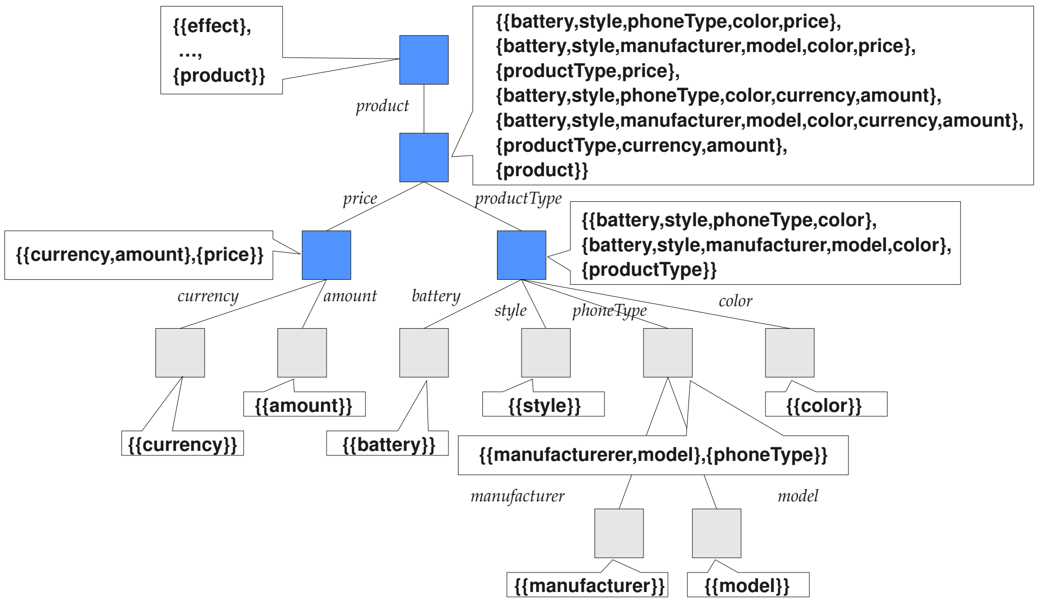

{{effect},
...,
{product}}
product
{{battery,style,phoneType,color,price},
{battery,style,manufacturer,model,color,price},
{productType,price},
{battery,style,phoneType,color,currency,amount},
{battery,style,manufacturer,model,color,currency,amount},
{productType,currency,amount},
{product}}
price
productType
{{currency,amount},{price}}
{{battery,style,phoneType,color},
{battery,style,manufacturer,model,color},
{productType}}
currency
amount
battery
style
phoneType
color
{{amount}}
{{style}}
{{color}}
{{currency}}
{{battery}}
{{manufacturerer,model},{phoneType}}
manufacturer
model
{{manufacturer}}
{{model}}

Figure 6.5.: Determining valid feedback structures

Figure 6.5 illustrates how the algorithm works, exemplary for the request (model) depicted in Figure 6.2. Starting from the request's (request model's) root attribute (*effect*), the algorithm first adds the feedback structure comprising just of the root attribute to the output set. It then recursively computes the set of feedback structures for the subtree rooted at its child attribute *product*, which in turn is determined by creating all possible combinations of an element from the feedback structure set determined for the subtree rooted at *price* and an element from the feedback structure set determined for the subtree rooted at *productType* and so on.

6.5.2. Feedback Structure Suitability

Users might have various reasons for preferring to judge certain service aspects to others, e.g. one might be willing to judge aspects that are important and might not be willing to judge aspects that are unimportant or private. Though those issues might be interesting from a psychological perspective, we only marginally consider them here (see Section 9.3.4). In fact, our approach to feedback structure recommendation is purely behavioral, i.e. it considers the effect of a consumer's judgment preferences, namely the resulting judgment behavior, but does not try to explain it. In particular, we assess the likelihood of a certain feedback structure to be judged by a user by considering his judgment behavior in past judgment sessions. The likelihood is estimated based on two behavioral indicators, namely the number of service aspects and the kind of service aspects that have been judged in the past. We do not consider other factors that might influence the likelihood of a feedback structure to be judged (see Section 9.3.4 for a short discussion on this topic). As already argued, a consumer's judgment preferences might vary for different service requirements, e.g. the user's judgment preferences after having booked a flight might differ from those after having purchased a book. We account for that fact by considering not just a user's past judgment behavior, but also the semantic similarity of the current request (model) and the requests (request models) that led to the past judgments. In this context, we presume, that a user's future judgment behavior will be similar to its past, if it refers to a similar judgment situation, i.e. a similar request (model). More specifically, we make the following two assumptions.

Assumption 6.2. (**Number of judged service aspects**) *Let r' be a past request (model) that led to the judgment of feedback structure fs' and let r be a future request (model) that is semantically similar to r'. Since the user was willing to judge as many service aspects as required by fs', it is likely that he is willing to judge a similar number of service aspects after the service interaction that is based on r. As a consequence, it is likely that he is willing to judge any feedback structure fs that is based on r and requires him to judge a number of service aspects that is similar to the number of aspects that had to be judged according to fs'. The similarity of the number of service aspects that had to be judged according to fs' and the number of aspects the user is willing to judge after a service interaction that is based on r is the higher, the more similar the involved requests (request models) are.*

Assumption 6.3. (**Type of judged service aspects**) *Let r' be a past request (model) that led to the judgment of feedback structure fs' and let r be a future service request (model) that is semantically similar to r'. Since the user was willing to judge the kinds of service aspects required by fs', it is likely that he is willing to judge similar kinds of service aspects after the service interaction that is based on r. As a consequence, it is likely that he is willing to judge any feedback structure fs that is based on r and requires him to judge service aspects that are similar to those that had to be judged according to fs'. The similarity of the kinds of service aspects that had to be judged according to fs' and the kinds of aspects the user is willing to judge after a service interaction that is based on r is the higher, the more similar the involved requests (request models) are.*

We think that those assumptions are reasonable and valid in the context of our scenario. This particularly holds for the latter assumption, since similar requests (request models) induce similar service attributes of interest and thus a similar set of service aspects that might be potentially judged. As we will see, our evaluation results will support this hypothesis (see Section 9.3.3). We do not consider other situational aspects that might have an impact on a user's judgment behavior. By doing so, we make the implicit assumption that the reasons for a service consumer's judgment behavior exclusively lie in his service requirements, which might not be true (see again Section 9.3.4 for a discussion on this topic).

In the remainder of this section, we will provide details on how the likelihood of a given feedback structure to be judged by the user is determined. We start with a discussion on how the probability that the user is willing to judge as many service aspects as required by a given feedback structure can be assessed, followed by remarks on how the likelihood that the user is willing to judge the kinds of service aspects as required by a given feedback structure can be estimated. Finally, we will explicate how those probabilities can be leveraged to assess the likelihood of a certain feedback structure to be judged by the user.

Likelihood with Respect to the Number of Service Aspects

Consider a user having posed the request (model) r and having invoked an appropriate service s after the retrieval of matching offers based on r. Let FS_r be the set of valid feedback structures that can be derived from r and that might be used to judge the performance of s with respect to r. Let further $E = \{(r',fs')|\, fs'$ was judged after having posed $r'\}$ be the set of information on the user's past judgment behavior, each characterized by the feedback structure fs' that was judged and the request (model) r' that led to the judgment. We refer to the probability distribution, indicating for each feedback structure $fs \in FS_r$ the likelihood of the user being willing to judge as many service aspects as is required by fs as $p_{num}(r,fs)$ (for the sake of clarity, we omit an index referring to FS_r). It is determined by Bayesian inference using the evidence provided in E, i.e. the information about the user's behavior in past judgment sessions. We start with a uniform prior distribution, i.e. when having no past judgment information, we simply assume that all feedback structures $fs \in FS_r$ are equally likely to be judged. The posterior probability distribution $p_{num}(r,fs|r',fs')$, taking the past judgment information $(r',fs') \in E$ into account, is given by

$$p_{num}(r,fs|r',fs') = c_{num} \cdot P_{num}(r',fs'|r,fs) \cdot p_{num}(r,fs), \tag{6.1}$$

where $p_{num}(r,fs)$ is the prior distribution before taking the observation (r',fs') into account and c_{num} is a normalizing constant chosen in a way ensuring that $p_{num}(r,fs|r',fs')$ is in fact a probability distribution. The probability $P_{num}(r',fs'|r,fs)$ indicates the likelihood of the user being willing to judge as many service aspects as required by the feedback structure fs' derived from r', if fs derived from r would have already been judged. It can be estimated using Assumption 6.2.

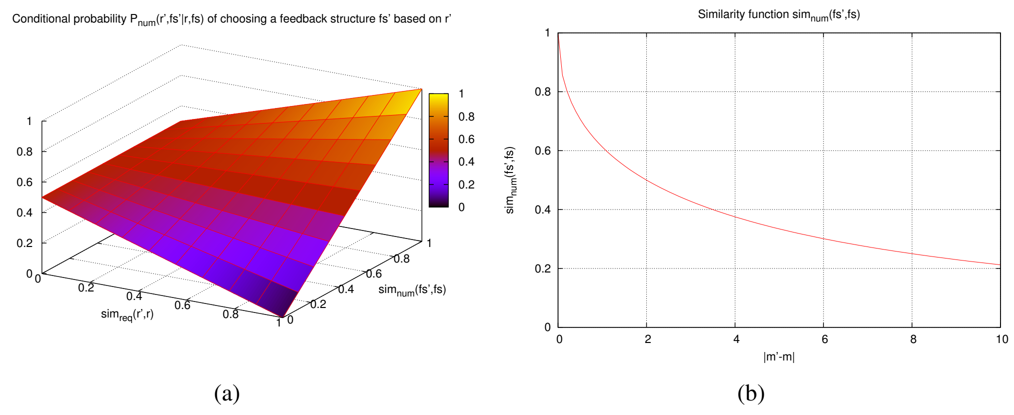


Figure 6.6.: Conditional probability $P_{num}(r',fs'|r,fs)$ of choosing a feedback structure fs' based on r' (a) and similarity $sim_{num}(fs',fs)$ of two feedback structures with respect to the numbers m' and m of service aspects they require to be judged (b)

Let fs be a feedback structure that was judged by a user in a past judgment session after having posed request r and having invoked an appropriate service. Let further be r' the current request (model) and fs' a valid feedback structure that might be judged by the user. Let $sim_{req}(r',r) \in [0,1]$ denote the semantic similarity of the requests (request models) r' and r, indicating how similar the service requirements encoded in the request (model) r' are to those in the request (model) r. A similarity value of 1 indicates that the two requests (request models) are semantically identical, while values lower than 1 indicate decreasing semantic similarity. A detailed discussion on how to compute this similarity will be provided in Section 6.5.3. Let further $sim_{num}(fs',fs) \in [0,1]$ be the similarity of the feedback structures fs' and fs with respect to the numbers m' and m of service aspects they require to be judged. A similarity value of 1 indicates that the values m' and m are identical, while values lower than 1 indicate increasing distance between the two values. Then, according to Assumption 6.2, the following holds. The likelihood $P_{num}(r',fs'|r,fs)$ of the user being willing to judge as many attributes as required by the feedback structure fs' provided that he already judged fs is 1, if the requests (request models) r and r' are semantically identical, i.e. $sim_{req}(r',r) = 1$, and the number m' of service aspects that has to be judged according to fs' is equal to the number of service aspects m that has been judged based on fs, i.e. if $sim_{num}(fs',fs) = 1$. If the numbers m and m' maximally differ, i.e. $sim_{num}(fs',fs) = 0$, then $P_{num}(r',fs'|r,fs) = 0$. This does not hold, if the two requests (request models) r and r' totally differ, i.e. $sim_{req}(r',r) = 0$. In this case, we cannot draw any conclusions about the user's willingness to judge as many service aspects as required by fs' from the fact that he judged fs. Hence, $P_{num}(r',fs'|r,fs)$ is set to 0.5 to indicate that both, judging fs' and not judging fs' is equally likely. The more similar the requests (request models) r and r' are, the more similar the user's feedback structure selection behavior will be to that exhibited in the past judgment session, i.e. when judging *fs*. The following estimation of the probability $P_{num}(r',fs'|r,fs)$ is in compliance with those considerations and depicted in Figure 6.6(a).

$$P_{num}(r',fs'|r,fs) = 0.5 + ((sim_{num}(fs',fs) - 0.5) \cdot sim_{req}(r',r)) \tag{6.2}$$

A natural measure for the similarity $sim_{num}(fs',fs)$ of two feedback structures with respect to the numbers m' and m of service aspects they require a user to judge is based on the distance $|m'-m|$ between m' and m. As argued, the similarity should be 1 for $|m'-m| = 0$ and should decrease to 0 with increasing distance, i.e. $\lim_{|m'-m|\to\infty} sim_{num}(fs',fs) = 0$. In our implementation, we use the following similarity measure that fulfills those requirements:

$$sim_{num}(fs',fs) = \frac{1}{a^{\sqrt{|m'-m|}}}. \tag{6.3}$$

The real number a can be freely chosen and determines how fast the similarity value decreases with increasing distance. For example, choosing $a = 1.6325$ causes the similarity value to be halved for a distance of 2. The resulting similarity function is depicted in Figure 6.6(b).

Note, that we do not necessarily have to consider all information in E to determine the probability distribution $p_{num}(r,fs|r',fs')$. Instead, we might restrict ourselves to the past judgment experience(s) that is/are most similar to the current judgment situation in terms of the request (model) that was posed, i.e. to $\{(r',fs') \in E | \neg\exists(r'',fs'') \in E$ with $sim_{req}(r,r') < sim_{req}(r,r'')\}$. This is particularly reasonable, if it would turn out that the Assumptions 6.2 and 6.3 just hold for very similar judgment situations. In our evaluation (Section 9.3), we considered the latter variant, since it delivered more accurate results than the former.

Likelihood with Respect to the Kinds of Service Aspects

Consider again a user having posed the request (model) r and having invoked an appropriate service after the retrieval of matching offers based on r. Let FS_r be the set of valid feedback structures that can be derived from r and let E be the set of information on the user's past judgment behavior. We refer to the probability distribution, indicating for each feedback structure $fs \in FS_r$ the likelihood of the user being willing to judge the kinds of service aspects as required by the feedback structure *fs* as $p_{attr}(r,fs)$ (again, we omit an index referring to FS_r for the sake of clarity). It is also determined by Bayesian inference using the evidence provided in E. It is initialized as a uniform distribution assigning equal probabilities to all valid feedback structures $fs \in FS_r$. The posterior probability distribution $p_{attr}(r,fs|r',fs')$, taking the past judgment information $(r',fs') \in E$ into account, is given by

$$p_{attr}(r,fs|r',fs') = c_{attr} \cdot P_{attr}(r',fs'|r,fs) \cdot p_{attr}(r,fs), \tag{6.4}$$

where $p_{attr}(r,fs)$ is the prior distribution before taking the observation (r',fs') into account and c_{attr} is a normalizing constant chosen in a way ensuring that $p_{attr}(r,fs|r',fs')$ is in fact a probability distribution. The probability $P_{attr}(r',fs'|r,fs)$ indicates the likelihood of the user being willing to judge the kinds of service aspects as required by the feedback structure fs' derived from r', if fs derived from r would have already been judged. It can be estimated similar to $P_{num}(r',fs'|r,fs)$ by making use of Assumption 6.3.

$$P_{attr}(r',fs'|r,fs) = 0.5 + ((sim_{attr}(fs',fs) - 0.5) \cdot sim_{req}(r', r)) \tag{6.5}$$

The value $sim_{attr}(fs',fs) \in [0,1]$ indicates the similarity of the set of attributes $A_{fs'}^{judged}$ that has to be judged according to the feedback structure fs' and the set of attributes A_{fs}^{judged} that has to be judged according to the feedback structure fs. As a measure for $sim_{attr}(fs',fs)$, we use Jaccard's similarity coefficient [Jac01] that is often used for comparing sample sets with respect to their elements. In particular, we define

$$sim_{attr}(fs',fs) = \frac{|A_{fs'}^{judged} \cap A_{fs}^{judged}|}{|A_{fs'}^{judged} \cup A_{fs}^{judged}|}. \tag{6.6}$$

The similarity value is 0, if the two attribute sets $A_{fs'}^{judged}$ and A_{fs}^{judged} do not share any attributes, and increases with increasing number of shared attributes up to 1 for sets that contain the same attributes.

Again, we do not necessarily have to consider all information in E to determine the probability distribution $p_{attr}(r,fs|r',fs')$, but might restrict ourselves to the past judgment experience(s) that is/are most similar to the current judgment situation in terms of the request (model) that was posed. In our evaluation (Section 9.3), we considered the latter variant.

Likelihood of a Feedback Structure to Be Judged

Let r be the request (model) encoding the user's requirements. We define the probability distribution $p_{judges}(r,fs;\alpha)$, that indicates the likelihood of the feedback structures $fs \in FS_r$ to be judged, to be

$$p_{judges}(r,fs;\alpha) = \alpha \cdot p_{num}(r,fs) + (1-\alpha) \cdot p_{attr}(r,fs). \tag{6.7}$$

The parameter α with $\alpha \in [0,1]$ determines the influence of the probabilities $p_{num}(r,fs)$ and $p_{attr}(r,fs)$, respectively. An appropriate value for α might vary from one user to another. In Section 6.5.4, we will demonstrate how it can be learned from a consumer's past judgment behavior.

Valid feedback structures $fs \in FS_r$ of r can now be compared with respect to $p_{judges}(r,fs;\alpha)$, i.e. their likelihood of being judged by the user. The most likely feedback structure is selected and presented to the user. Listing 6.2 summarizes the overall recommendation procedure.

```
FeedbackStructure recommendStructure(Request(Model) r, Experiences E) {
    /* determine all valid feedback structures for r */
    validFeedbackStructures = determineValidStructures(root(r))

    /* retrieve the judgment experience that is most similar w.r.t. to the request (model) it is based on */
    experience = retrieveMostSimilarExperience(E,r)

    /* determine the likelihood of being judged for each valid feedback structure */
    pNum.initialize(validFeedbackStructures)
    pNum.bayesianUpdate(experience)

    pAttr.initialize(validFeedbackStructures)
    pAttr.bayesianUpdate(experience)

    pJudges = experience.getAlpha · pNum + (1−experience.getAlpha) · pAttr

    /* return the feedback structure that is most likely to be judged by the user */
    return getMostLikelyFeedbackStructure(pJudges)
}
```

Listing 6.2: Determining the feedback structure that is most likely to be judged by a user having posed the request (model) r when being provided with the experiences E about the user's past judgment behavior

6.5.3. Request Model Similarity

As mentioned earlier, a consumer's judgment preferences might depend on his service requirements encoded in his request (model). To allow for a comparison of the service requirements that underlie a set of judgments and thus to allow for determining the relevance of a user's judgment behavior exhibited in a past judgment session for the prediction of its current judgment preferences, a measure for the semantic similarity of two requests (request models), i.e. the similarity of the service requirements they encode, is required. In this section, we will propose such a measure. Thereby, we restrict ourselves to the comparison of request models. A similarity measure for DSD request descriptions (cf. Section 4.2) has been also proposed and has been published in [FK10].

The semantic similarity $\mathit{sim}_{req}(r, r')$ of two request models r and r' is recursively defined. It is computed by determining the similarity $\mathit{sim}_{req}(r_{root(r)}, r'_{root(r')})$ of the request model subtrees rooted at the involved model's root attributes $\mathit{root}(r)$ and $\mathit{root}(r')$. More specifically, the similarity $\mathit{sim}_{req}(r_a, r'_{a'})$ of two request model subtrees r_a and $r'_{a'}$ rooted at the attributes a and a' in r and r', respectively, is calculated by computing the similarity $\mathit{sim}_{type}(r_a, r'_{a'})$ of their root attribute's ontological type (the *type similarity*), the similarity $\mathit{sim}_{dc}(r_a, r'_{a'})$ of their root attributes with respect to the direct conditions they have specified (the *direct condition similarity*) and the aggregated similarity $\mathit{sim}_{sub}(r_a, r'_{a'})$ of the request model subtrees rooted at their root nodes' child attributes (the *subtree similarity*). More specifically, we define $\mathit{sim}_{req}(r_a, r'_{a'})$ to be the weighted mean of these three values, i.e.

$$\mathit{sim}_{req}(r_a, r'_{a'}) = \alpha \cdot \mathit{sim}_{type}(r_a, r'_{a'}) + \beta \cdot \mathit{sim}_{sub}(r_a, r'_{a'}) + \gamma \cdot \mathit{sim}_{dc}(r_a, r'_{a'}), \qquad (6.8)$$

where the values α, β and γ with $\alpha + \beta + \gamma = 1$ determine the weights and thus the influence of the three similarity terms on the (overall) similarity of two request model subtrees. In the remainder of this section, we will explain the rationale behind the three involved similarity measures and particularize on how they are determined. Possible similarity values $sim_{req}(r_a, r'_{a'})$ are from the interval $[0, 1]$, where a similarity value of 0.0 means "not similar at all" and a value of 1.0 means that the service requirements encoded by the two request models are semantically identical. Values lying inbetween indicate different degrees of semantic similarity.

Determining the type similarity The type similarity $sim_{type}(r_a, r'_{a'}) \in [0, 1]$ of two request model subtrees r_a and $r'_{a'}$ indicates how similar their root attributes a and a' are with respect to the ontological type of their target sets I_a and $I'_{a'}$. To measure this similarity, we adopt again the idea of Jaccard's index [Jac01]. In particular, we define

$$sim_{type}(r_a, r'_{a'}) = \frac{|A_{I_a} \cap A_{I'_{a'}}|}{|A_{I_a} \cup A_{I'_{a'}}|}, \tag{6.9}$$

where A_{I_a} is the set of attributes that the service ontology defines for the type of a's target set in r_a and $A_{I'_{a'}}$ is the set of attributes that it defines for the type of a''s target set in $r'_{a'}$. The rationale behind this choice is, that two ontological types are the more closely related, the more attributes they share.

As an example, consider Figure 6.7. Presuming that *battery*, *phoneType* and *color* are all attributes that are defined for the ontological type *Phone* and the attributes *battery*, *phoneType*, *color* and *style* are all attributes that are defined for its subtype *MobilePhone*, the type similarity of the subtrees rooted at the *productType*-attributes of the depicted request model subtrees $r_{productType}$ and $r'_{productType}$ is $\frac{|\{battery, phoneType, color\}|}{|\{battery, phoneType, color, style\}|} = 0.75$.

Determining the direct condition similarity The direct condition similarity $sim_{dc}(r_a, r'_{a'}) \in [0, 1]$ of two request model subtrees r_a and $r'_{a'}$ indicates how similar their root attributes a and a' are with respect to the uncertain direct conditions that have been specified on their target sets. In Section 5.4, we introduced three types of uncertain direct conditions, namely in-conditions, not-in-conditions and range conditions, for request model attributes, that allow to restrict acceptable values of a service attribute. A natural measure for the similarity of two attributes with respect to those conditions is their similarity in terms of the values they consider (not) acceptable. However, such a measure should account for the uncertainty that is associated with direct conditions on request model attributes, i.e. should take the probability with which a certain direct condition is fulfilled into account. Hence, as a measure for the similarity of two attributes with respect to their direct conditions we choose the expected probability of a valid attribute value being either acceptable with respect to both attributes or not acceptable with respect to both attributes.

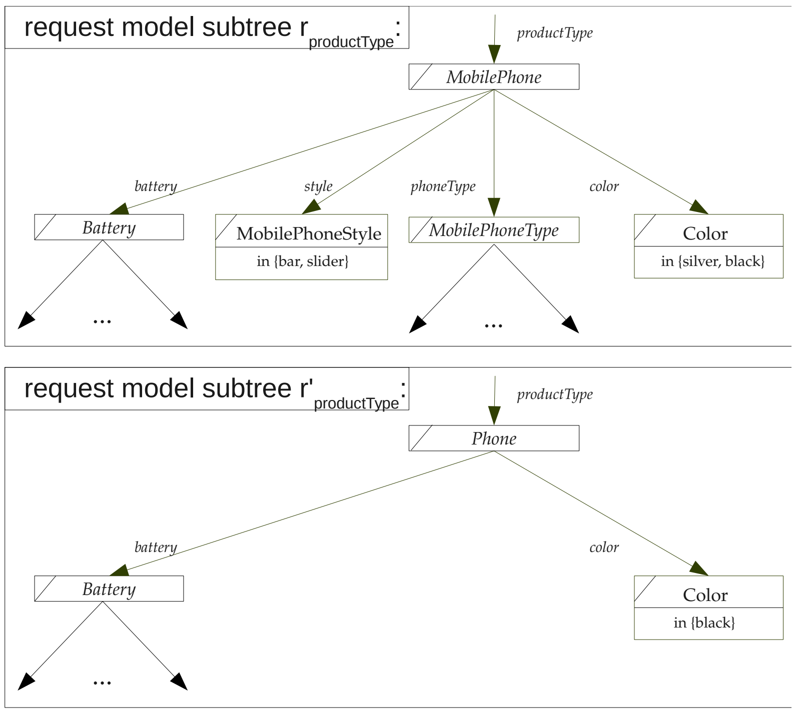


Figure 6.7.: Determining the semantic similarity of two request model subtrees rooted at the attribute *productType*

Thereby, we assume that attribute values occur with equal probability in the available service offers[8]. We do not consider preferences over attribute values. More specifically, we define

$$\begin{aligned} sim_{dc}(r_a, r'_{a'}) &= \mathbb{E}[P^{acc}_a(v) \cdot P^{acc}_{a'}(v) + (1 - P^{acc}_a(v)) \cdot (1 - P^{acc}_{a'}(v)] \\ &= \frac{\sum_{v \in V_a \cup V_{a'}} [P^{acc}_a(v) \cdot P^{acc}_{a'}(v) + (1 - P^{acc}_a(v)) \cdot (1 - P^{acc}_{a'}(v)]}{|V_a \cup V_{a'}|}, \end{aligned} \tag{6.10}$$

where V_a and $V_{a'}$ are the sets of valid values that a and a' might take and $P^{acc}_a(v)$ and $P^{acc}_{a'}(v)$ are the probabilities of a certain attribute value v being acceptable with respect to the direct condition(s) that have been specified for the target sets I_a and $I'_{a'}$ of a and a', respectively.

The calculation of this expected value differs depending on the type of direct condition(s) that are involved. If the direct condition that has been specified for the target set I_a of

[8]The estimation of the expected probability would be more precise if we would consider only those attribute values that actually occur in one of the available service offers. This would also allow for a better estimation of the probability of occurrence of a certain attribute value. However, it would introduce additional computational effort for determining occurring attribute values, which in light of the accuracy of the described judgment recommendation algorithm is unnecessary (cf. Section 9.3).

a certain attribute a is an in-condition, the probability $P_a^{acc}(v)$ is given by $P_a^{acc}(v) = P(\mathit{In}_v^{I_a} = true)$. If the specified direct condition is a not-in-condition, $P_a^{acc}(v)$ is given by $P_a^{acc}(v) = 1 - P(\mathit{NotIn}_v^{I_a} = true)$ (cf. Section 5.4). Note, that the computational effort for calculating the expected value only depends on the number of values that have been either considered in the direct condition on I_a or $I'_{a'}$. This is due to the fact that $P_a^{acc}(v) = P(\mathit{In}_v^{I_a} = true) = 0$ for attribute values v that have not been specified in an in-condition and $P_a^{acc}(v) = 1 - P(\mathit{NotIn}_v^{I_a} = true) = 1 - 0$ for attribute values that have not been specified in a not-in-condition. Hence, valid attribute values that have not been considered in the direct conditions have to be simply counted and jointly considered to compute the expected value. For example, having two attributes a and a' whose target sets I_a and $I'_{a'}$ both specify an in-condition, the direct condition similarity $\mathit{sim}_{dc}(r_a, r'_{a'})$ is given by

$$\frac{\sum_{v \in V_{a,a'}^{spec}} [P_a^{acc}(v) \cdot P_{a'}^{acc}(v) + (1 - P_a^{acc}(v)) \cdot (1 - P_{a'}^{acc}(v))] + |(V_a \cup V_{a'}) \setminus V_{a,a'}^{spec}|}{|V_a \cup V_{a'}|},$$

where $V_{a,a'}^{spec}$ is the set of attribute values that have been considered in the direct condition specified for I_a or $I'_{a'}$. This similarly works if both target sets specified a not-in-condition, if one target set specified an in- and the other a not-in-condition and if one target set specified an in- or not-in-condition and the other did not specify any direct condition.

As an example, consider again the request model subtrees depicted in Figure 6.7. Assume that valid values for the attribute *color* include *silver, black, anthracite* and *white* and that each of the attribute values *silver* and *black*, that have been considered in the in-conditions is acceptable with probability 0.5. Then, the direct condition similarity for the subtrees rooted at the *color*-attributes in the depicted request model subtrees $r_{\mathit{productType}}$ and $r'_{\mathit{productType}}$ is $\frac{(0.5 \cdot 0 + 0.5 \cdot 1) + (0.5 \cdot 0.5 + 0.5 \cdot 0.5) + 2}{4} = 0.75$.

We also consider the calculation of the direct condition similarity for target sets that both specify a range condition. Let R_{I_a} and $R_{I'_{a'}}$ be the ranges from which values of I_a and $I'_{a'}$ can be taken and let $|R_{I_a} \cup R_{I'_{a'}}|$ be the length of the range that results by joining the ranges R_{I_a} and $R_{I'_{a'}}$. Then analogously to Formula 6.10, the direct condition similarity $\mathit{sim}_{dc}(r_a, r'_{a'})$ is given by

$$\frac{\int_{v \in R_{I_a} \cup R_{I'_{a'}}} (p_a^{acc}(v) \cdot p_{a'}^{acc}(v) + (1 - p_a^{acc}(v)) \cdot (1 - p_{a'}^{acc}(v)))dv}{|R_{I_a} \cup R_{I'_{a'}}|}, \qquad (6.11)$$

where $p_a^{acc}(v)$ is a probability distribution indicating for each attribute value v its probability of being acceptable with respect to the direct condition(s) that have been specified for the target set I_a and $p_a^{acc}(v)$ is given by $p_a^{acc}(v) = \mathit{Prob}(\mathit{Min}_{I_a} \leq v < \mathit{Max}_{I_a}) = \int_{z=\min(R_{I_a})}^{v} p_{\mathit{Min}_{I_a}}(z)dz \cdot (1 - \int_{z=\min(R_{I_a})}^{v} p_{\mathit{Max}_{I_a}}(z)dz)$[9] (cf. Section 5.4). This similarly works, if one of the target sets specifies a range condition and the other one does not specify a direct condition.

[9]Since in our implementation continuous distributions are discretized, the computation of the integral can be done with reasonable effort.

If none of the target sets I_a and $I'_{a'}$ specifies a direct condition, the direct condition similarity is defined to be 1.0. For instance the request model subtrees depicted in Figure 6.7 both do not specify conditions on the attribute *battery*. Hence, the direct condition similarity for the subtrees rooted at the *battery*-attributes in the depicted request models is 1.0. At this time, similarity calculation for pairs of subtrees, one of which's root attribute's target set specifying a range condition and the other specifying an in- or not-in-condition, as well as for pairs where target sets specify more than one direct condition is not supported.

Determining the subtree similarity The similarity value $sim_{sub}(r_a, r'_{a'}) \in [0,1]$ of two request model subtrees r_a and $r'_{a'}$ indicates how similar their root attributes a and a' are with respect to their child trees and allows for the recursive calculation of the request model similarity. Let A be the set of attributes defined in the service ontology, either for the type of a, the type of a' or for both types and let $\{sim_{req}(r_{a_i}, r'_{a_i}) | a_i \in A\}$ be the set of request model similarity values for the corresponding subtrees of r_a and r'_a rooted at the attributes $a_i \in A$. Inspired again by Jaccard's index, the subtree similarity, i.e. the aggregated similarity of two request model subtrees with respect to the subtrees rooted at their root attributes is defined as the weighted sum of the request model similarity values divided by the weighted sum of the maximal similarity values that can be achieved for each pair of corresponding subtrees (1.0).

$$sim_{sub}(r_a, r'_{a'}) = \frac{\sum_{a_i \in A} w_{a_i} \cdot sim_{req}(r_{a_i}, r'_{a_i})}{\sum_{a_i \in A} w_{a_i}} \tag{6.12}$$

Thereby, the weights w_{a_i}, $a_i \in A$ are defined to be

$$w_{a_i} = max(w_{r_a, a_i}, w_{r'_{a'}, a_i}), \tag{6.13}$$

where w_{r_a, a_i} and $w_{r'_{a'}, a_i}$ are the expected (relative) weights of a_i in the request model subtrees r_a and $r'_{a'}$ rooted at the attributes a and a' as defined in Section 5.6 (Formula 5.19). The rationale behind that choice is that the similarity of two corresponding attribute subtrees is the more important and thus should have the more influence on the subtree similarity, the more important (indicated by a high weight) the attribute is in at least one of the request model subtrees r_a and $r'_{a'}$. Consider for example an attribute *price*, if the *price*-weight is low with respect to both compared request model subtrees, then the similarity of the corresponding subtrees rooted at the *price*-attribute is rather unimportant.

Since attributes in A are not necessarily defined for both, the type of the target set I_a and the target set $I'_{a'}$, we set $sim_{req}(r_{a_i}, r'_{a_i}) = 0.0$, if the attribute $a_i \in A$ is not defined for one of the types. Attributes in A might also not be specified for one or both of the target sets I_a and $I'_{a'}$. If an attribute $a_i \in A$ is not specified for both target sets, we set $sim_{req}(r_{a_i}, r'_{a_i}) = 1.0$, else, if a_i is specified for just one of the target sets, $sim_{req}(r_{a_i}, r'_{a_i})$ is defined to be $sim_{req}(r_{a_i}, t')$ or $sim_{req}(t, r'_{a_i})$, respectively, where t is a request model tree comprising of a single target set, having the most generic type defined for a_i's target set in the service ontology.

We illustrate the recursive computation of the subtree similarity by means of the *productType*-subtrees depicted in Figure 6.7. The type of the *productType* attribute in $r_{\textit{productType}}$ is *MobilePhone*, while that in $r'_{\textit{productType}}$ is *Phone*. Assume, that the ontology defines the attributes *battery*, *phoneType* and *color* for the type *Phone* and an additional attribute *style* for the type *MobilePhone*, which is assumed to be a subtype of *Phone*. The subtree similarity of the two corresponding *productType*-subtrees is determined by the similarity of their root attributes' corresponding child trees for the attributes $A = \{\textit{battery, phoneType, color, style}\}$. The subattributes *battery* and *color* are specified in both depicted subtrees. Hence, the similarity values $sim_{req}(r_{\textit{battery}}, r'_{\textit{battery}})$ and $sim_{req}(r_{\textit{color}}, r'_{\textit{color}})$ can be computed by determining the request model similarity for the corresponding request model subtrees rooted at the *battery*-attributes and those rooted at the *color*-attributes. The attribute *style* is only defined for the type *MobilePhone*, hence $sim_{req}(r_{\textit{style}}, r'_{\textit{style}}) = 0.0$. The attribute *phoneType* is defined for both types, *MobilePhone* and *Phone*, but is only specified in $r_{\textit{productType}}$. Hence, $r'_{\textit{phoneType}} := t'$, where t' is a subtree comprising of a single node having the most generic type defined for the target set of attribute *phoneType*. Thus the required request model similarity $sim_{req}(r_{\textit{phoneType}}, r'_{\textit{phoneType}})$ is determined by computing $sim_{req}(r_{\textit{phoneType}}, t')$, where $r_{\textit{phoneType}}$ is the subtree of $r_{\textit{productType}}$ rooted at the node of type *MobilePhoneType*.

6.5.4. Dynamically adjusting α

As discussed earlier, the parameter α that weights the influence of the probabilities $p_{num}(r,\mathit{fs})$ and $p_{attr}(r,\mathit{fs})$ on the (overall) likelihood $p_{judges}(r,\mathit{fs};\alpha)$ of a feedback structure being judged by a user, might vary from user to user (cf. Section 6.5.2). In this section, we will demonstrate how this value can be learned from a consumer's past judgment behavior.

Initially, i.e. without having information about a user's previous judgment behavior, we do not know anything about the parameter's value, so α could be any value from the interval $[0, 1]$. Hence, for the purpose of computing the probability distribution $p_{judges}(r,\mathit{fs};\alpha)$ that indicates the likelihood of being judged for the valid feedback structures fs of r, we equally weight the probabilities $p_{num}(r,\mathit{fs})$ and $p_{attr}(r,\mathit{fs})$, i.e. we set $\alpha = 1 - \alpha = 0.5$. Once having determined the feedback structure fs_{rec} that is most likely to be judged, we present it to the consumer, who has the opportunity to change it by expanding and/or collapsing attributes. Finally, the consumer provides judgments for the resulting feedback structure's leaf attributes. Let the feedback structure that was finally judged by the user be fs_{judged}. Inspired by the idea of maximum-likelihood estimation, we determine how α should have been chosen to make the judgment of feedback structure fs_{judged} most likely according to $p_{judges}(r,\mathit{fs};\alpha)$. This can be easily done using knowledge about the set of valid feedback structures FS_r that can be derived from r and knowledge about the feedback structure fs_{judged} that was actually judged. More specifically, we know that for each unjudged feedback structure $\mathit{fs} \in FS_r \setminus \{\mathit{fs}_{judged}\}$, the inequality $p_{judges}(r,\mathit{fs};\alpha) \leq p_{judges}(r,\mathit{fs}_{judged};\alpha)$ must hold. Using Formula 6.7, we find that

$$\alpha \leq \frac{p_{judges}(r,\mathit{fs}_{judged};\alpha) - p_{attr}(r,\mathit{fs})}{p_{num}(r,\mathit{fs}) - p_{attr}(r,\mathit{fs})} \tag{6.14}$$

for $p_{num}(r,fs) > p_{attr}(r,fs)$ and

$$\alpha > \frac{p_{judges}(r,fs_{judged};\alpha) - p_{attr}(r,fs)}{p_{num}(r,fs) - p_{attr}(r,fs)} \tag{6.15}$$

for $p_{num}(r,fs) < p_{attr}(r,fs)$. Using those information, we can adjust, i.e. shrink the range of α correspondingly. For example, if $\alpha \leq 0.8$ holds, we adjust the interval to $[0, 0.8]$. Redundant information, such as if $\alpha \leq 0.8$ holds, when already having $\alpha \in (0.5, 0.7]$, are simply ignored. Information about the range that has been determined for α is stored with the other information about the judgment session and can be leveraged to recommend a suitable feedback structure in a future session. This is done as follows. Given the request model r_{curr}, for which a feedback structure shall be recommended, we retrieve information about the past judgment session that is based on a request model r_{past} that is most similar to r_{curr}, i.e. for all request models r that led to past judgment sessions, the inequality $sim_{req}(r, r_{curr}) \leq sim_{req}(r_{past}, r_{curr})$ holds. If more than one past judgment session with this property exists, we select the most recent one. The parameter α that has been determined based on the user's judgment behavior in the selected session (the midpoint of the determined range) is used in the recommendation process of the current judgment session.

6.6 Utilizing Feedback to Predict a Service's Future Performance

The focus of this section is on how to effectively use elicited consumer feedback to infer knowledge about a service's future performance and thus about the risk that is associated with its execution. Before elaborating on the prediction procedure itself, we will define our notion of service performance more precisely and will exactly specify the goal of the performance prediction.

Definition 6.7. (**Service Performance**) *Service performance is a subjective term that indicates to what degree a given service's outcome fulfills the service requirements of a user. It is measured in terms of (a) rating(s) the user provides after service execution.*

Imagine, a service consumer c has posed a service request (which is derived from a request model, if the requirements elicitation procedure described in Chapter 5 has been used) for which a number of matching service offers have been retrieved. The goal of the performance prediction is to anticipate the actual performance of those services with respect to the consumer's service requirements. More formally, given a matching service offer for service s, we would like to know how c would judge the performance of s (possibly with respect to several service aspects) when actually executing it. Let p be the provider of service s. We propose to leverage feedback provided by consumers that interacted with services of p in the past to predict the required judgment. Since feedback referring to a particular service is typically scarce and since it is very unlikely that if such feedback exists, it was made in the context of a request that is equal to ours, we suggest to consider not just feedback that is related to the considered service and request, but any feedback related to services offered by provider p (Requirement F.4).

However, as argued before, feedback items are of different value for the judgment prediction. In particular, judgments referring to a service that offers a functionality which is similar to that offered by s, i.e. that have been made in a similar *service (offer) context*, are more valuable for the prediction than those that refer to a completely different kind of service. This is due to the fact that a service provider is likely to behave similarly when offering a similar service, i.e. when being in a similar service (offer) context. Hence, we can infer from a service provider's behavior in one service (offer) context about his behavior in a similar offer context. The more similar the offered service functionality, the more similar the behavior we can expect. Moreover, judgments which are made by consumers that had similar service requirements as c, i.e. that posed a similar request and thus were in a similar *request context*, are more valuable for the prediction than those that are based on completely different requirements. The reason for this is that users with similar requirements are likely to judge the same service interaction similarly. Hence, we can infer from a user's judgment made in one request context about another user's judgment in a similar request context. The more similar the users' service requirements and thus the posed requests, the more similar the judgment behavior we can expect. Hence, a feedback item is the more relevant and thus the more valuable for the prediction of a given judgment, the more similar the request and the service (offer) context it was made in are to the given request and service (offer) context. The relevance of a feedback with respect to the request and the service (offer) context in which it was made is covered by the notions of *feedback provider* and *service (offer) similarity*.

Definition 6.8. (**Feedback provider and service (offer) similarity**) *Let r be a service request that was posed and s of service provider p a service whose offer matches to r. Consider the rating prediction for s. The service (offer) similarity sim$_{service}(f, s)$ of a certain feedback item f (referring to an interaction with p) and the service s indicates the similarity of the functionality offered by the service that was judged in f and the functionality offered by s. The feedback provider similarity sim$_{provider}(f, r)$ of f and the request r indicates the similarity of r and the request on which f is based.*

Hence, the higher the feedback provider and the service (offer) similarity of a feedback item, the higher its relevance for the prediction of a considered judgment. Please note, that by defining feedback provider similarity in the specified way, we implicitly assume that the service request (and as a prerequisite the request model from which it is derived) accurately reflects a service consumer's service requirements and in particular his preferences (Assumption 6.1). More specifically, we presume that the feedback provider similarity, which is based on the similarity of requests, indicates the similarity of the service requirements the providers of those requests have. As already argued, we cannot presume that this assumption is entirely fulfilled. Hence, in Chapter 9, we will analyze how a relaxation of the required assumption impacts the quality of a service's performance prediction.

Having introduced the terms service (offer) and feedback provider similarity as well as feedback relevance, we will now detail on how similarity and relevance can be actually measured and determined and finally considered when making a prediction.

6.6.1. Determining Feedback Relevance

As discussed in Section 6.4, we do not determine feedback relevance by directly comparing service requests or request models, since to protect personal information, consumer feedback does not contain request (model) descriptions (Requirement F.3). Relevance computation is rather based on indirect context information (Definition 6.5) that are delivered with each feedback item. We propose to apply techniques known from collaborative filtering (CF) systems [SFHS07] to approach this issue. Those systems aim at predicting a consumer's unknown rating for a product or item by utilizing ratings of other users that share a similar taste. Those neighboring users are identified by comparing the ratings they provided for different purchased items with those of the target user. The underlying assumption is that the more similar the ratings are, the more similar are the users' tastes and thus the more similar their ratings for other products will be. We illustrate this technique,

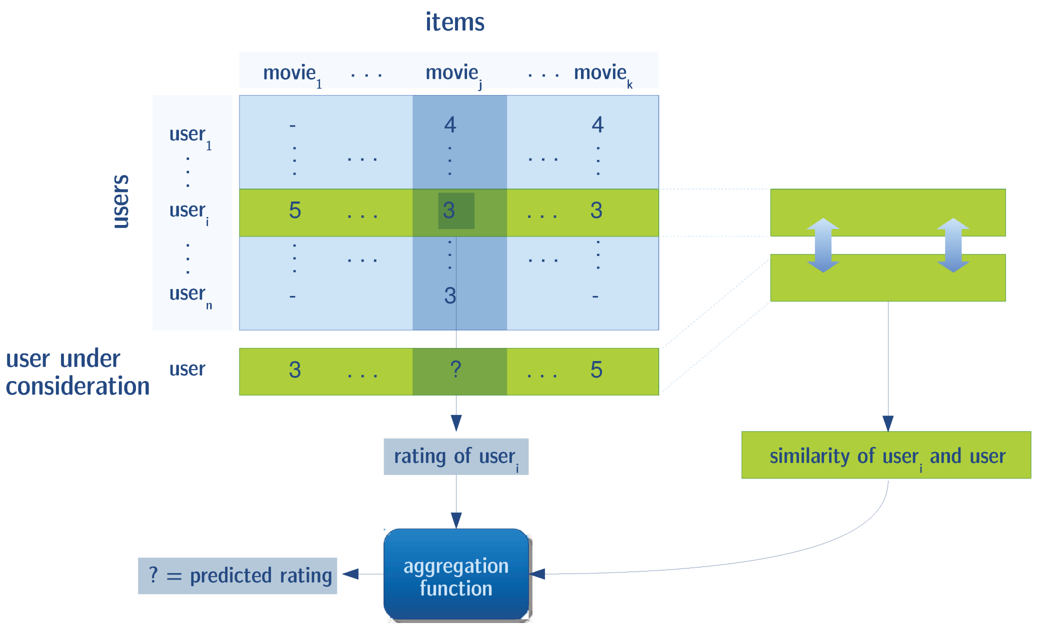


Figure 6.8.: User-based collaborative filtering

which is known as user-based collaborative filtering [SFHS07], with an example. Imagine an online video rental service, such as Netflix [10], which offers about $100,000$ movies and other titles on DVDs. The provider of such a service is interested in ensuring customer

[10]http://www.netflix.com

satisfaction, since it wants to retain its customers. To achieve this, the video rental service might allow its customers to rate watched movies on a 5-star-scale and might offer personalized movie recommendations based on those ratings. Recommendations for a certain customer could e.g. be generated by predicting the considered customer's unknown ratings for each movie and then recommending the 10 highest rated movies to him. The required unknown movie ratings could be determined by using collaborative filtering techniques. Figure 6.8 demonstrates how this works. It shows an example matrix (top) containing the movie ratings of the rental service users and the ratings provided by the target customer, whose rating shall be predicted (middle). Imagine, we attempt to predict the customer's rating for the movie $movie_j$. To accomplish this, we extract all users that rated the movie $movie_j$ in the past. As a prediction for the unknown movie rating, we can use some aggregate, e.g. a weighted mean, of the ratings those user's provided for movie $movie_j$. To personalize the prediction, we should assign higher weights to users whose taste is similar to that of the target user. In user-based collaborative filtering, the weight for each user's rating is determined by comparing this user's and the target user's judgments for those items that have been co-rated by them. In the literature, various similarity measures have been proposed to perform this comparison (examples will follow below).

How can this principle be applied to determine the relevance of a feedback item? Let again r be a service request that was posed and s of service provider p a service whose offer matches to r. Consider the rating prediction for s. As already argued, a feedback item referring to a service interaction with p is the more relevant, the more similar the service requirements and thus the service request of the feedback provider is to that of the target user (high feedback provider similarity) and the more similar the functionality offered by the service that was judged is to that offered by s (high service similarity). We suggest to apply user-based collaborative filtering to determine the required feedback provider similarity. More specifically, the similarity of the feedback provider's and the target user's service requirements is determined by comparing the ratings they would provide for the available service offers O_M that match to both the feedback provider's and the target user's request. However, we do not have those ratings at our disposal. To solve this problem, we make use of the fact, that we have explicit models of both, the service requirements of the feedback provider and the service requirements of the target user, namely the service requests they have posed, as well as explicit models of the matching service offers. Hence, we can employ semantic matchmaking to determine those judgments. Let r be the service request of the target user and r_f be the past request of the feedback provider. To determine how similar the requirements of the feedback provider are to those of the target user, we compare the matching values of the service offers in O_M calculated with respect to r_f with those that have been computed with respect to r. A similar technique, item-based collaborative filtering [SFHS07], is applied to determine the similarity of the functionality offered by the service that was judged by the feedback provider and that offered by service s, whose future performance shall be predicted. For that purpose, we have to compare the matching values provided for the two services' offers with respect to different requests. The more similar those values are, the more similar is the functionality offered by the two services. Note, that we already have all the required matching information available, since they are part of the indirect context information (Definition 6.5) that are delivered with each feedback item. However, we must remark again, that we buy the advantage of automation by assuming that the requirements captured in the service request (model) reflect the actual consumer requirements (Assumption 6.1). In the remainder of this section, we will detail on how the comparison of the involved matching values is actually performed.

Let again r be a service request that was posed and s of service provider p a service whose performance with respect to r shall be predicted. Assume, that the set of feedback items F, that contains judgments that refer to past service interactions with services of p, is provided to perform this task. In a first step, we create a feedback matrix FM_F with a row for each available feedback item $f \in F$ and a column for each service offer $o \in O_F$, where O_F is the set of offers o for which at least one of the feedback items in F provides matching information (Figure 6.9). Suppose that the feedback item f was based on the request r_f. Then the matrix entry $FM_F(f, o)$ stores the matching value $mv(r_f, o)$ of r_f and o (if this value is not known this is indicated).

Definition 6.9. (**Feedback matrix**) *We define the feedback matrix that is derived from a set of feedback items F by*

$$FM_F = [FM_F(f, o) | f \in F \wedge o \in O_F \wedge FM_F(f, o) = mv(r_f, o)], \quad (6.16)$$

where $O_F = \bigcup_{f \in F} O_f$ and O_f is the set of service offers that match to r_f, i.e. the set of service offers for which f provides matching information. If a matching degree $mv(r_f, o)$ is not known, this is indicated.

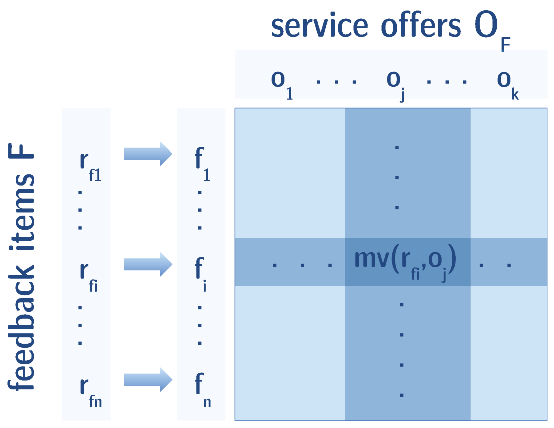


Figure 6.9.: Feedback matrix FM_F

Let $\vec{O}_F$ be the vector of offers from O_F as sorted in the feedback matrix and let $\overrightarrow{MR}(r, \vec{O}_F)$ be the vector of matching results that we obtain for the request r and the offers in $\vec{O}_F$. To determine the feedback provider similarity $sim_{provider}(f, r)$ of f and r, i.e. the similarity of the ratings the provider of f and the target user would give for the same services, we have to compare the row $FM_F(f,*)$ of the feedback matrix with $\overrightarrow{MR}(r, \vec{O}_F)$ (Figure 6.10), i.e.

$$sim_{provider}(f,r) = simMeasure(FM_F(f,*), \overrightarrow{MR}(r, \vec{O}_F)), \tag{6.17}$$

where *simMeasure* is a similarity measure and unknown (due to missing matching information) entries are ignored.

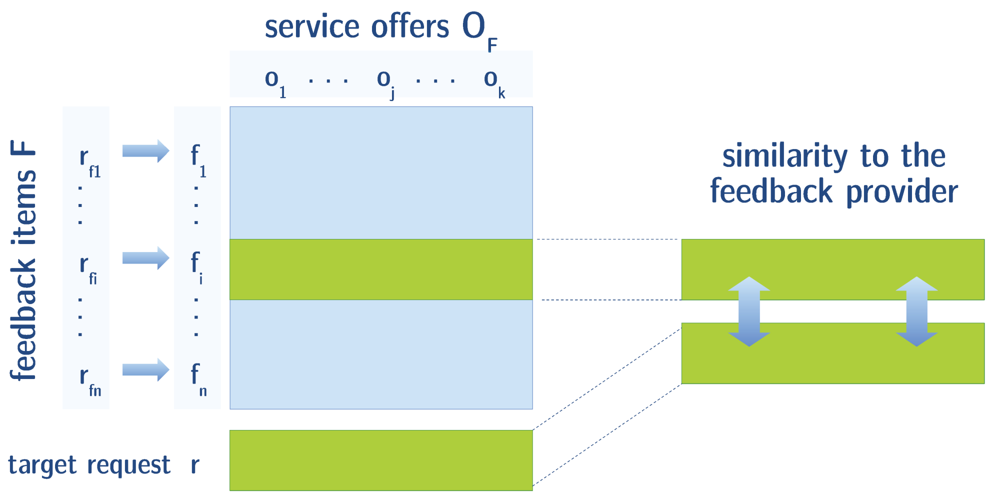


Figure 6.10.: Determining the feedback provider similarity

Similarly, to determine the service (offer) similarity $sim_{service}(f,s)$ of f and s, i.e. the similarity of the functionality offered by the rated service s_f and that of service s, we are interested in, we compare the column for the offer o_{s_f} of s_f and the column for the offer o_s of s, i.e.

$$sim_{service}(f,s) = simMeasure(FM_F(*, o_{s_f}), FM_F(*, o_s)). \tag{6.18}$$

Again, unknown entries are ignored. If similarity information for a given feedback item $f \in F$ cannot be obtained, we assume the similarity to be 0.

In the literature on collaborative filtering systems a number of similarity measures have been proposed. We implemented our approach with two measures, that have proven to deliver good results in many domains, namely a slightly modified version of the *correlation-based similarity measure* proposed in [RIS+94]

$$\begin{aligned} corr(\vec{x},\vec{y}) &= \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2\sum_{i=1}^{n}(y_i-\bar{y})^2}} \\ simMeasure(\vec{x},\vec{y}) &= \begin{cases} 0 & \text{if } corr(\vec{x},\vec{y}) < 0 \\ corr(\vec{x},\vec{y}) & \text{otherwise} \end{cases} \end{aligned} \tag{6.19}$$

and the *cosine-based similarity measure* introduced in [BHK98]

$$simMeasure(\vec{x},\vec{y}) = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}} \tag{6.20}$$

and evaluated how well they perform in the context of our approach (Section 9.4). Both measures take values from the interval $[0,1]$, where a similarity value of 0 means not similar at all and a value of 1 means equal. Similarity values from $(0,1)$ indicate different grades of similarity. Once the similarity information for each feedback item in F have been calculated, the future performance of the considered service s, possibly with respect to several service aspects, can be predicted by leveraging the similarity information as well as the attribute judgments provided by each feedback item. In the subsequent section we will detailedly describe this process. In particular, we will answer the challenging question of how to integrate feedback items that are based on various feedback structures to infer knowledge about a service's future performance.

6.6.2. Rating Prediction Based on Coarse-Grained Similarity Information

Let r, depicted in Figure 6.11 left, be the service request (request model) that was posed and s the service for which attribute ratings shall be predicted. Consider a single feedback item f, which is based on the feedback structure *fs*, depicted in Figure 6.11 right. Imagine, we would like to know how s, if executed, would perform with respect to the service aspect *product* according to the judgments provided in f. To answer this question, we have to check whether the feedback structure *fs* contains the aspect *product* within the same context, i.e. the same path (Definition 6.4), as the aspect *product* in r. Thereby, we consider the paths of the two service aspects to be the same, if they share the same attribute names, but not necessarily the same types. Though this might mean to infer a service aspect's performance from a judgment that refers to a different service aspect, it is ensured that those aspects are closely related. The strength of this relationship is accounted for by considering the feedback provider similarity of the feedback item f and the request (model) r. If *fs* would not contain the aspect *product* within the same path, *fs* would not provide any information about this aspect. Fortunately, the opposite is true. However, the judgment provider did not directly rate the service aspect *product*. Hence, we have to rely on the aggregated *product*-rating stored in *fs*. As can be seen in the feedback structure, this judgment does not consider the service aspect *style*, which has been specified in r. However, this fact is again accounted for by considering the feedback provider similarity of the feedback item f and the request (model) r.

The described procedure is repeated for all service aspects that are considered in a given feedback structure and for all feedback items $f \in F$ that are available and relate to service interactions with the provider of s. Ultimately, we end up with a set of attribute ratings for each service aspect that was specified in the request (model). Those ratings stem from

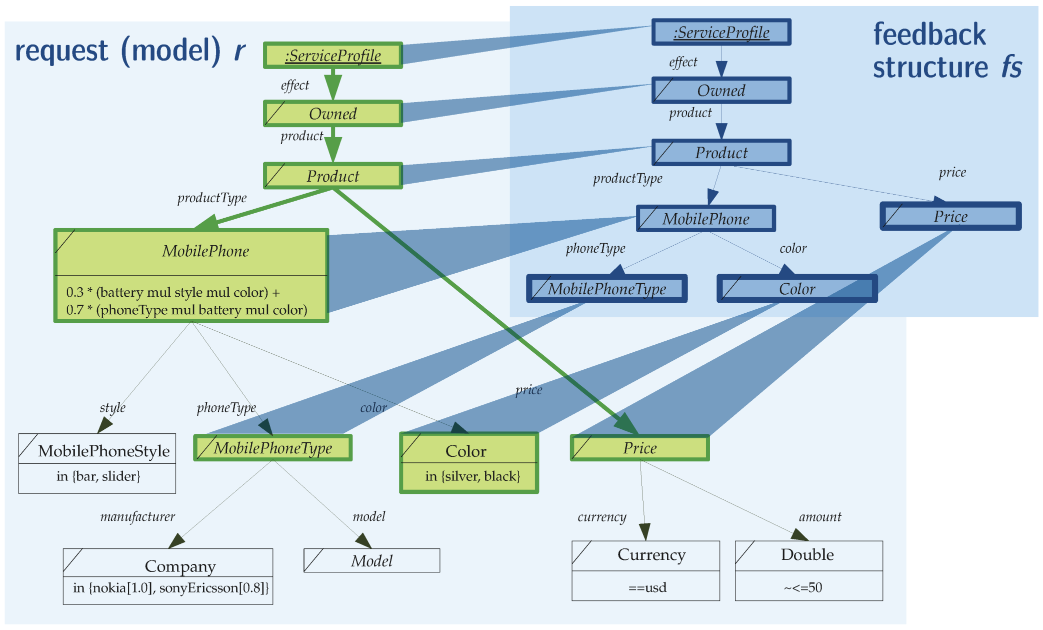


Figure 6.11.: Feedback extraction

the available feedback items. In a final step, the unknown rating for each service aspect in r with respect to the considered service s is estimated as an aggregate of the ratings for this attribute as contributed by the single feedback items. Obviously, the choice of the aggregation function that is applied for this purpose has a great impact on the quality of the prediction (see also Section 9.4). We implemented and tested our approach with the aggregation function *adjusted weighted mean* (Formula 6.21), which is known from collaborative filtering systems and seems to deliver good prediction results in many application domains [Ado05]. It provides a prediction for the unknown rating $\widetilde{rat}_a(r,s) \in [0,1]$ of a certain service aspect a of r with respect to the service s and the posed request r. The prediction is based on a's judgments derived from the feedback items $F_a \neq \emptyset$ that refer to service interactions with the service provider of s and that contribute a rating for a. Using adjusted weighted mean as an aggregation function, the predicted rating $\widetilde{rat}_a(r,s)$ is given by

$$\widetilde{rat}_a(r,s) = \overline{rat}_a + k \sum_{f \in F_a} w_f(r,s) \cdot (rat_a(f) - \overline{rat}_a), \tag{6.21}$$

where $rat_a(f)$ is the rating for a contributed by feedback item f, $\overline{rat}_a = \frac{\sum_{f \in F_a} rat_a(f)}{|F_a|}$ and $k = \frac{1}{\sum_{f \in F_a} w_f(r,s)}$ is a normalizing factor. The weight $w_f(r,s)$ of a certain feedback item $f \in F$ indicates its relevance for the rating prediction and is determined as the product of the feedback provider and the service similarity of this feedback item.

$$w_f(r,s) = sim_{provider}(f,r) \cdot sim_{service}(f,s), \tag{6.22}$$

where r is the service request (request model) for which ratings shall be predicted and s is the service under consideration. Please note, that the similarity information, that is used for the rating prediction, is the same for all aspect-specific judgments of a feedback item. Also note, that for aspects which are not considered in any of the feedback item's feedback structures, no ratings are available and thus no rating prediction can be performed.

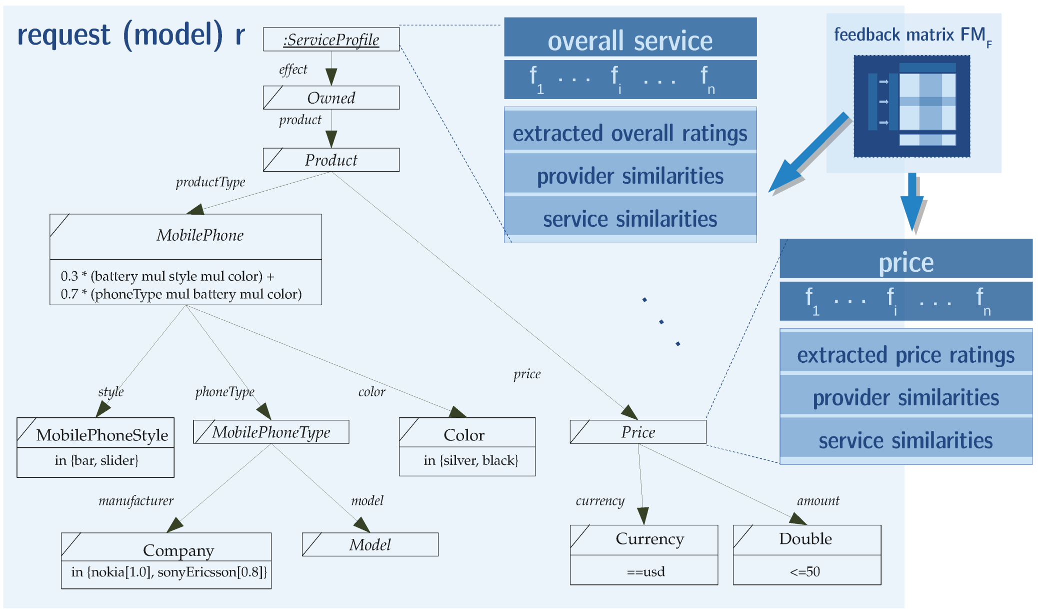


Figure 6.12.: Feedback aggregation based on coarse-grained similarity information

Listing 6.3 and Figure 6.13(a) summarize the introduced procedure for feedback extraction and rating prediction. Figure 6.12 exemplary shows the data that are involved in this process. The suggested procedure can be repeated for all service offers that match to a given request to predict their performance. Section 6.7 will provide details on how those information can be presented to the user and how they can be leveraged to produce a feedback-aware service ranking.

6.6.3. Rating Prediction Based on Fine-Grained Similarity Information

The prediction algorithm introduced above, assigns the same weight to each attribute rating that has been contributed by a given feedback item f, i.e. independently from the service aspect, whose unknown rating shall be predicted, the attribute judgment contributed by f obtains the same weight (cf. Formula 6.21). More specifically, when estimating the unknown rating for a given service aspect, the rating contributed by feedback item f is weighted by the product of the service (offer) similarity and the feedback provider similarity of f (Formula 6.22). These similarity values refer to the similarity of the functionality

```
PredictedRatings ratingPredictionCoarse(Request $r$, Service $s$) {
    /* get feedback items */
    - get all feedback items $f \in F$ that refer to interactions with the
        provider $p$ of $s$

    /* create feedback matrix */
    - create a feedback matrix $FM_F$ from the feedback items $f \in F$

    /* determine the similarities using the feedback matrix */
    for(feedback item $f \in F$) {
        serviceSimilarities.add($f$,determineServiceSimilarity($s$,$f$,$FM_F$))
        providerSimilarities.add($f$,determineProviderSimilarity($r$,$f$,$FM_F$))
    }

    /* extract judgments from the feedback items */
    for(feedback item $f \in F$) {
        for(service aspect $a \in f$) {
            judgments.get($a$).add(getJudgment($a$,$f$))
        }
    }

    /* predict ratings */
    for(attribute $a$ of $r$) {
        predictedRatings.add($a$, aggregateRatings(judgments.get($a$),
            serviceSimilarities,providerSimilarities))
    }

    return predictedRatings
}
```

Listing 6.3: Leveraging coarse-grained similarity information to predict attribute-specific ratings for the service s with respect to the request (model) r

offered by the service, whose rating shall be predicted, and that offered by the judged service as a whole as well as to the posed service request (model) and the request (model) with respect to which the judgment was provided as a whole. However, this strategy does not account for the fact that, when predicting the unknown rating for a given service aspect, the judgments stemming from those feedback items, that refer to service interactions that were similar with respect to this service aspect, are more valuable for the rating prediction and thus should be weighted higher. As an example, consider a wine purchasing request (model) r. Let s be a matching service offer, whose unknown ratings shall be predicted, and f a feedback item, referring to a past service interaction with the provider of s. Imagine, that f also refers to a service interaction, where a bottle of wine was purchased. The wine that was sold during that interaction (moderately) differed from the wine that is offered by service s, but both wines have nearly the same price. Consider the prediction of the unknown rating for the service aspect *price*. Since the offered wines differ, the service similarity of f and thus its weight is not too high and consequently, the price judgment contributed by it will have a relatively low impact on the result of the rating prediction. Though giving low weights to the judgment contributions of f might be reasonable when predicting ratings for other service aspects that are specified in r, the weight of the price rating contributed by f should be high, since both, the offered and the purchased wine are

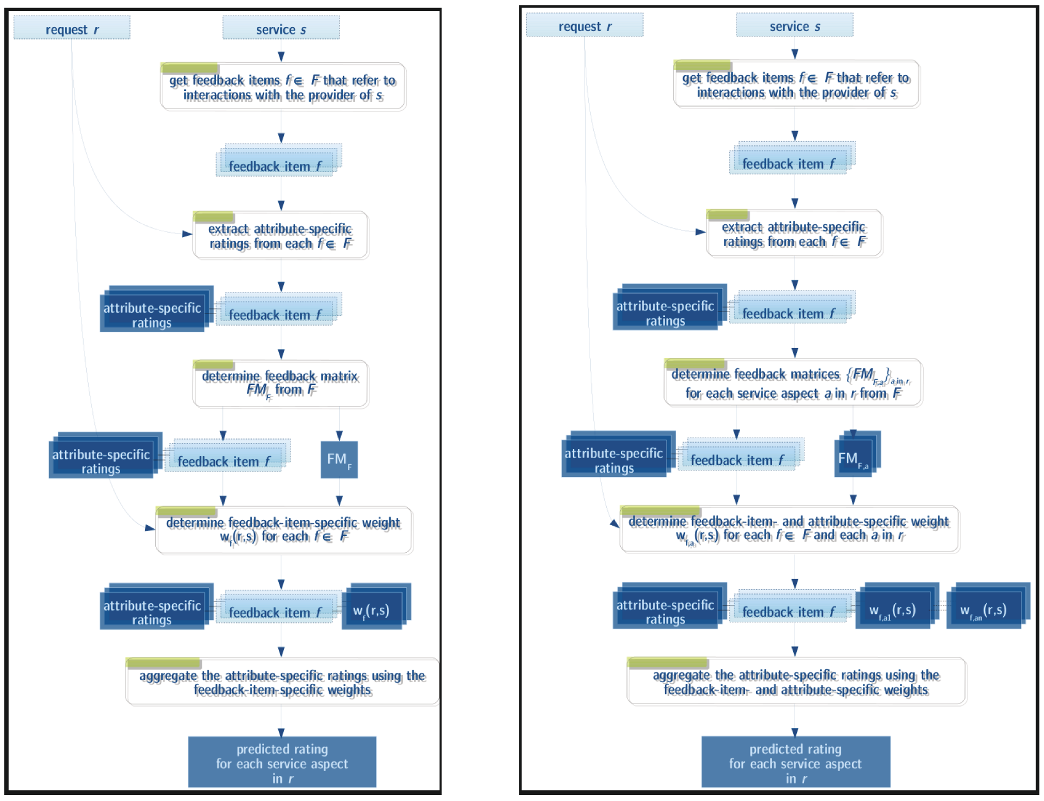

request r
service s
get feedback items f ∈ F that refer to interactions with the provider of s
feedback item f
extract attribute-specific ratings from each f ∈ F
attribute-specific ratings
feedback item f
determine feedback matrix FM_F from F
FM_F
determine feedback-item-specific weight w_f(r,s) for each f ∈ F
w_f(r,s)
aggregate the attribute-specific ratings using the feedback-item-specific weights
predicted rating for each service aspect in r
determine feedback matrices {FM_F,a} a in r for each service aspect a in r from F
FM_F,a
determine feedback-item- and attribute-specific weight w_f,a(r,s) for each f ∈ F and each a in r
w_f,a1(r,s)
w_f,an(r,s)
aggregate the attribute-specific ratings using the feedback-item- and attribute-specific weights
(a)
(b)

Figure 6.13.: Rating prediction based on coarse-grained (a) and fine-grained (b) similarity information

quite similar with respect to their price. Of course, this argumentation is only reasonable, if the (overall) service and feedback provider similarity of f is fairly high. Otherwise, we would highly weight a judgment that stems from an interaction that totally differs from the current one. In the remainder of this section, we will present a modified version of our rating prediction algorithm presented in the previous section (Listing 6.3) that accounts for those facts. It is based on the calculation of individual, i.e. attribute-specific, similarity values for each attribute judgment that is contributed by a feedback item.

Definition 6.10. **(Attribute-specific feedback provider and service (offer) similarity)** *Let r be a service request that was posed and s of service provider p a service whose offer matches to r. Consider the prediction of the unknown rating for s with respect to the service aspect a. The attribute-specific service (offer) similarity $attrSim_{service}(f, s, a)$ of a certain feedback item f (referring to an interaction with p) and the service s with respect to the service aspect a indicates the similarity of the functionality offered by the service that was judged in f and the functionality offered by s when compared with respect to the attribute a. The attribute-specific feedback provider similarity $attrSim_{provider}(f, r, a)$ of f*

and the request r with respect to the service aspect a indicates the similarity of r and the request on which f is based when compared with respect to the attribute a.

Calculating attribute-specific similarity values. Attribute-specific similarity values are determined by maintaining an individual feedback matrix for each request (model) attribute (Figure 6.14). Let r be a service request (model) that was posed and s of service provider p a service whose performance with respect to r shall be predicted. Assume, that the set of feedback items F, that contains judgments that refer to past service interactions with services of p, is provided to perform this task. Just as the feedback matrix defined in the previous section (Definition 6.9), aspect-specific feedback matrices contain a row for each available feedback item $f \in F$ and a column for each service offer $o \in O_{F,a}$, where $O_{F,a}$ is the set of offers o for which at least one of the feedback items in F provides matching information with respect to a. Let $FM_{F,a}$ be the feedback matrix for the service attribute a that has been specified in r. Suppose that the feedback item f was based on the request r_f. Then, instead of storing an (overall) matching value, the matrix entry $FM_{F,a}(f, o)$ stores the (partial) matching value $mv_a(r_f, o)$ of r_f and o with respect to a, i.e. the matching degree of the request and service subtrees rooted at a. The required (intermediate) matching degrees are part of the indirect context information (Definition 6.5) that are delivered with each feedback item. If a matching degree is not known, the corresponding matrix entry is marked as unknown.

Definition 6.11. (**Attribute-specific feedback matrix**) *The attribute-specific feedback matrix $FM_{F,a}$ for the service aspect a, that is derived from a set of feedback items F, is defined by*

$$FM_{F,a} = [FM_{F,a}(f, o) | f \in F \wedge o \in O_{F,a} \wedge FM_{F,a}(f, o) = mv_a(r_f, o)], \qquad \boxed{6.23}$$

where $O_{F,a} = \bigcup_{f \in F} O_{f,a}$ and $O_{f,a}$ is the set of service offers that match to r_f with respect to a, i.e. the set of service offers for which f provides matching information with respect to a. If a matching degree $mv_a(r_f, o)$ is not known, the corresponding matrix entry is marked as unknown.

Let $\vec{O}_{F,a}$ be the vector of offers from $O_{F,a}$ as sorted in the feedback matrix and let further $\overrightarrow{MR}_a(r, \vec{O}_{F,a})$ be the vector of (partial) matching degrees for a that we obtain for the request r and the offers in $\vec{O}_{F,a}$. To determine the attribute-specific feedback provider similarity $attrSim_{provider}(f, r, a)$ of f and r with respect to the service aspect a, i.e. the similarity of the a-ratings the provider of f and the target user give for the same services, we have to compare the row $FM_{F,a}(f, *)$ of the feedback matrix with $\overrightarrow{MR}_a(r, \vec{O}_{F,a})$, i.e.

$$attrSim_{provider}(f, r, a) = simMeasure(FM_{F,a}(f, *), \overrightarrow{MR}_a(r, \vec{O}_{F,a})), \qquad \boxed{6.24}$$

where unknown (due to missing matching information) entries are ignored.

Similarly, to determine the attribute-specific service (offer) similarity $attrSim_{service}(f, s, a)$ of f and s with respect to the service aspect a, i.e. the similarity of the functionality offered by the rated service s_f and that of service s, we are interested in, with respect to a, we compare the column for the service offer o_{s_f} of s_f and the column for the offer o_s of s, i.e.

$$\mathit{attrSim}_{service}(f, s, a) = \mathit{simMeasure}(\mathit{FM}_{F,a}(*, o_{s_f}), \mathit{FM}_{F,a}(*, o_s)), \qquad \boxed{6.25}$$

where again, unknown entries are ignored. In both formulas, *simMeasure* is either the correlation-based similarity measure (Formula 6.19) or the cosine-based similarity measure (Formula 6.20), that has been introduced in Section 6.6.1. If attribute-specific similarity information for a given feedback item $f \in F$ and a given attribute cannot be obtained, we assume the similarity to be 0.

Determining the weights for the rating prediction. Let r be the request that was posed and s the service, whose ratings with respect to r shall be predicted. Consider the prediction of the unknown rating for service aspect a. As detailed in Section 6.6.2 (Listing 6.3), the predicted attribute rating $\widetilde{rat}_a(r, s)$ is an aggregate of the judgments for service aspect a that are provided by the feedback items F_a that refer to interactions with the service provider of s and contribute a judgment for a. As already argued, the function 6.21 can be used as the aggregation function. However, instead of weighting each feedback item $f \in F_a$ with its (overall) similarity as proposed in the previous section, we introduce attribute- and feedback-item-specific relevance weights $w_{f,a}(r, s)$. The weight $w_{f,a}(r, s)$ for the judgment of a that was contributed by feedback item f is defined to be the product of f's overall similarity $\mathit{ovSim}(f, r, s)$ and the attribute-specific similarity $\mathit{attrSim}(f, r, s, a)$ of f with respect to service aspect a, i.e.

$$w_{f,a}(r, s) = \mathit{ovSim}(f, r, s) \cdot \mathit{attrSim}(f, r, s, a). \qquad \boxed{6.26}$$

The attribute-specific similarity $\mathit{attrSim}(f, r, s, a)$ of f with respect to service aspect a indicates the relevance of the feedback item f with respect to the service aspect a and is determined as the product of the attribute-specific feedback provider similarity and the attribute-specific service similarity of a, i.e.

$$\mathit{attrSim}(f, r, s, a) = \mathit{attrSim}_{provider}(f, r, a) \cdot \mathit{attrSim}_{service}(f, s, a). \qquad \boxed{6.27}$$

The proposed weighting scheme ensures that an attribute judgment is weighted high, if and only if both, the similarity of the feedback item as a whole and its similarity with respect to the rated aspect are high. Our evaluation shows that leveraging individual similarity values for each attribute rating results in significant improvements of the prediction quality when predicting attribute-specific ratings (Section 9.4).

Similar to the feedback-item-specific relevance weight $w_f(r, s)$, that was used in the previous section (Listing 6.3, Definition 6.21), $\mathit{ovSim}(f, r, s)$ denotes to the overall similarity of the feedback item f, i.e. its similarity with respect to the complete request r and service offer s, determined as the product of f's overall service similarity $\mathit{ovSim}_{service}(f, r, s)$ with respect to s and its overall feedback provider similarity $\mathit{ovSim}_{provider}(f, r)$ with respect to r, i.e.

$$\mathit{ovSim}(f, r, s) = \mathit{ovSim}_{provider}(f, r) \cdot \mathit{ovSim}_{service}(f, r, s). \qquad \boxed{6.28}$$

However, in contrast to $w_f(r,s)$, its calculation is not based on the comparison of (overall) matching values (i.e. by using a single feedback matrix), but on fine-grained similarity information in terms of the available attribute similarities. As indicated in [GA07] and shown in our evaluation (Section 9.4), this leads to more accurate similarity estimates and thus significantly improves the prediction quality. We tested and implemented two algorithm variants for calculating the overall service and feedback provider similarity: (1) determining the overall service/feedback provider similarity of a feedback item f as the mean of the attribute-specific service/feedback provider similarities over the set A_r of all service aspects that are considered in r (*attribute-specific similarity mean*), i.e.

$$\begin{aligned} \mathit{ovSim}_{service}(f,r,s) &= \frac{\sum_{a\in A_r} \mathit{attrSim}_{service}(f,s,a)}{|A_r|} \\ \mathit{ovSim}_{provider}(f,r) &= \frac{\sum_{a\in A_r} \mathit{attrSim}_{provider}(f,r,a)}{|A_r|}, \end{aligned} \tag{6.29}$$

and (2) determining the overall service/feedback provider similarity of a feedback item f as the weighted mean of those similarities (*weighted attribute-specific similarity mean*), i.e.

$$\begin{aligned} \mathit{ovSim}_{service}(f,r,s) &= \frac{\sum_{a\in A_r} w^*_{r,a} \cdot \mathit{attrSim}_{service}(f,s,a)}{\sum_{a\in A_r} w^*_{r,a}} \\ \mathit{ovSim}_{provider}(f,r) &= \frac{\sum_{a\in A_r} w^*_{r,a} \cdot \mathit{attrSim}_{provider}(f,r,a)}{\sum_{a\in A_r} w^*_{r,a}}. \end{aligned} \tag{6.30}$$

In the latter case, $w^*_{r,a}$ refers to the total weight of a service aspect a in request r, i.e. the weight that is given by the product of the weights that are assigned to the request attributes that lie on the path (Definition 6.4) of a in r as specified by the corresponding connecting strategies. As can be easily seen, the weight $w^*_{r,a}$ indicates to what degree the matching value of service aspect a influences the overall matching value of r[11]. This weighting scheme ensures that the similarity value of the service aspects, that are more important to the user, obtain a higher weight. Listing 6.4 and Figure 6.13(b) summarize the introduced procedures for feedback extraction and rating prediction. Figure 6.14 exemplary shows the data that are involved in this process.

6.7 Customizable Feedback-Aware Service Ranking and Presentation of Feedback Information

In the previous sections, we dealt with effective feedback elicitation (Section 6.5) and the utilization of the collected information to predict a service's performance in future

[11]We assume, that the connecting strategies are of the type weighted sum.

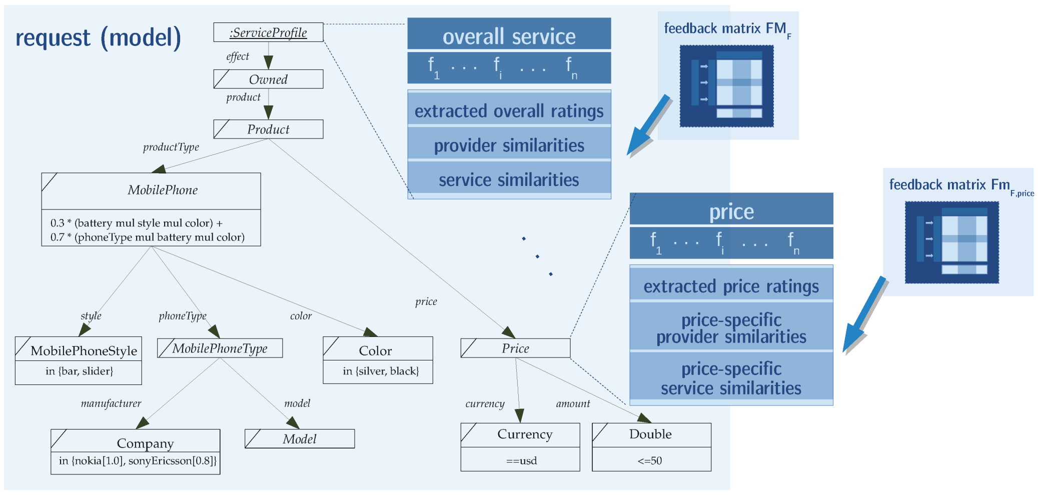


Figure 6.14.: Feedback aggregation based on fine-grained similarity information

interactions (Section 6.6). Though the knowledge that has been acquired in those steps is essential for enabling service consumers to make well-informed service selection decisions, it is of little use, if it is not communicated and appropriately presented to the user. Hence, in this section, we will focus on these issues. In particular, we will comment on how knowledge inferred from consumer feedback can be effectively and intuitively presented to make the user aware of the risk that is associated with the execution of a service (Requirement F.7, Section 6.7.1). We will also demonstrate how feedback information can be leveraged to rank and subsequently present alternative service offers in a consumer-specific and feedback-aware way. The latter will enable service consumers to customize the displayed information according to their personal risk attitude (Requirement F.8, Section 6.7.2).

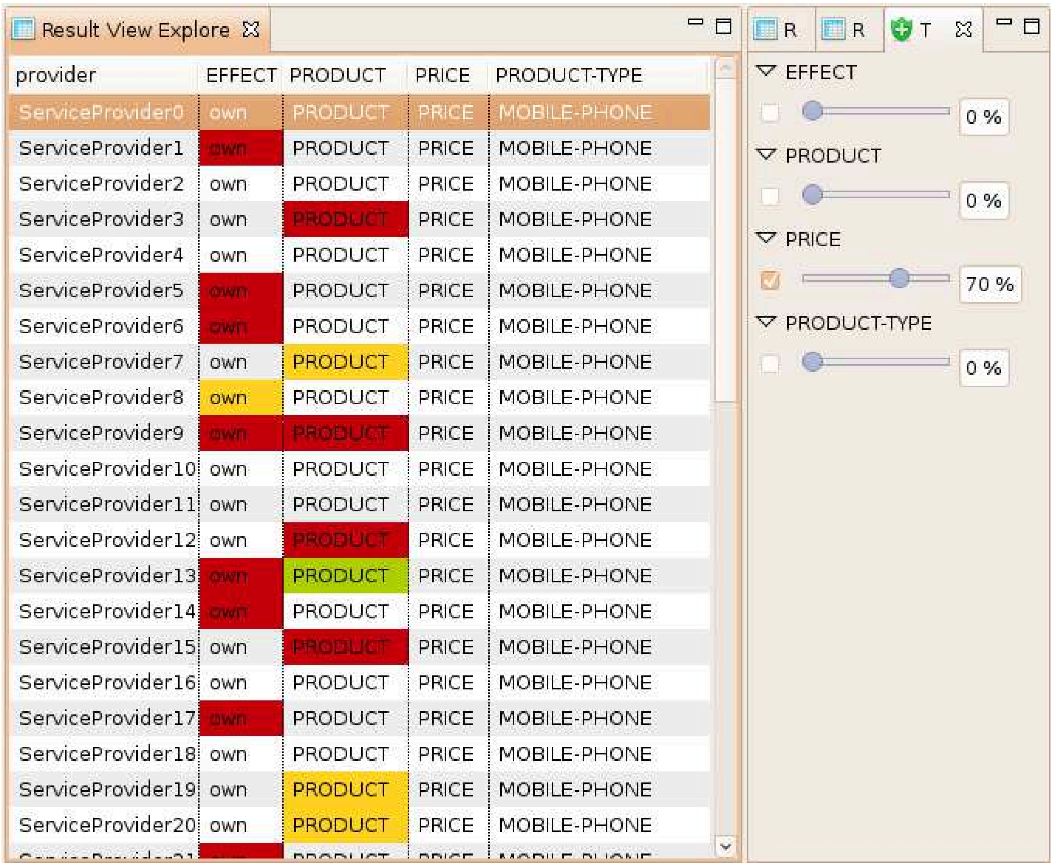
6.7.1. Presentation of Feedback Information

The judgment prediction mechanism introduced in the previous section, supplies us with a set of predicted ratings for each available and matching service alternative. Those ratings refer to the service aspects that have been specified in the posed service request (model), provided that sufficient feedback for a considered service aspect is available, and indicate whether and to what degree the offered service will provide the functionality that is required by the user. As mentioned earlier (Section 5.5), suitable service offers are presented to the user as a ranked list, that is sorted by the overall (expected) matching value of the offers and provides attribute-specific information for each service alternative. The predicted judgments related to each service aspect and offer are presented in addition to those data and are encoded by the background color of the corresponding cell in the results table (cf. Figure 6.15).

```
PredictedRatings ratingPredictionFine(Request $r$, Service $s$) {
    /* get feedback items */
    - get all feedback items $f \in F$ that refer to interactions with the
        provider $p$ of $s$

    /* create feedback matrices */
    for(service aspect $a$ of $r$) {
        - create a feedback matrix $FM_{F,a}$ from the feedback items $f \in F$
    }

    /* determine the similarities using the feedback matrices */
    for(feedback item $f \in F$) {
        for(attribute $a$ of $r$) {
            serviceSimilarities.add($f$,$a$,determineServiceSimilarity($s$,$f$,$FM_{F,a}$
                ))
            providerSimilarities.add($f$,$a$,determineProviderSimilarity($r$,$f$,
                $FM_{F,a}$))
        }
    }

    /* extract judgments from the feedback items */
    for(feedback item $f \in F$) {
        for(service aspect $a \in f$) {
            judgments.get($a$).add(getJudgment($a$,$f$))
        }
    }

    /* predict ratings */
    for(attribute $a$ of $r$) {
        predictedRatings.add($a$, aggregateRatings(judgments.get($a$),
            serviceSimilarities,providerSimilarities))
    }

    return predictedRatings
}
```

Listing 6.4: Leveraging fine-grained similarity information to predict attribute-specific ratings for the service s with respect to the request r

However, judgment prediction is based on experiences made by other service consumers in other service (offer) and request contexts and thus might produce inaccurate predictions, that differ from the judgments a consumer would provide when actually executing a considered service. This effect is even enhanced, since the suggested prediction algorithm presumes that service requests accurately reflect the service requirements and in particular the preferences of the service consumers who posed them (Assumption 6.1). This is an assumption that cannot be entirely fulfilled. Though we will demonstrate that the accuracy of the provided predictions is still sufficiently high (Chapter 9), it is unrealistic to assume that there will be a one-to-one correspondence between the predicted judgments and those, the user would provide when actually judging a service interaction. This is particularly true, at the beginning of the incremental feedback elicitation process presented in Chapter 5 of this thesis, where the conformance between the service consumer's request that is generated from the request model and his actual service requirements is likely to be low. To account for this problem, we decided to assign each service aspect of an offer to a class



provider	EFFECT	PRODUCT	PRICE	PRODUCT-TYPE
ServiceProvider0	own	PRODUCT	PRICE	MOBILE-PHONE
ServiceProvider1	own	PRODUCT	PRICE	MOBILE-PHONE
ServiceProvider2	own	PRODUCT	PRICE	MOBILE-PHONE
ServiceProvider3	own	PRODUCT	PRICE	MOBILE-PHONE
ServiceProvider4	own	PRODUCT	PRICE	MOBILE-PHONE
ServiceProvider5	own	PRODUCT	PRICE	MOBILE-PHONE
ServiceProvider6	own	PRODUCT	PRICE	MOBILE-PHONE
ServiceProvider7	own	PRODUCT	PRICE	MOBILE-PHONE
ServiceProvider8	own	PRODUCT	PRICE	MOBILE-PHONE
ServiceProvider9	own	PRODUCT	PRICE	MOBILE-PHONE
ServiceProvider10	own	PRODUCT	PRICE	MOBILE-PHONE
ServiceProvider11	own	PRODUCT	PRICE	MOBILE-PHONE
ServiceProvider12	own	PRODUCT	PRICE	MOBILE-PHONE
ServiceProvider13	own	PRODUCT	PRICE	MOBILE-PHONE
ServiceProvider14	own	PRODUCT	PRICE	MOBILE-PHONE
ServiceProvider15	own	PRODUCT	PRICE	MOBILE-PHONE
ServiceProvider16	own	PRODUCT	PRICE	MOBILE-PHONE
ServiceProvider17	own	PRODUCT	PRICE	MOBILE-PHONE
ServiceProvider18	own	PRODUCT	PRICE	MOBILE-PHONE
ServiceProvider19	own	PRODUCT	PRICE	MOBILE-PHONE
ServiceProvider20	own	PRODUCT	PRICE	MOBILE-PHONE

EFFECT	☐	0 %
PRODUCT	☐	0 %
PRICE	✓	70 %
PRODUCT-TYPE	☐	0 %

Figure 6.15.: Customizable feedback-aware service ranking and presentation of feedback information

based on its predicted judgment. Instead of coloring cells in the result table according to the predicted judgment of the corresponding service attribute, we color them as indicated by the attribute's class. The classification is based on the value of the predicted judgment $\widetilde{rat}_a(r, s)$ for the considered service attribute a with respect to the request (model) r and the offer s (Definition 6.21) and the confidence in this prediction (Section 9.4.6). We distinguish the following 4 classes:

- **class 0** - no prediction information available due to none or non-sufficient feedback information (low confidence)
- **class 1** - low suitability, i.e. $\widetilde{rat}_a(r, s) < \theta_{low}$, with high confidence
- **class 2** - neither nor, i.e. $\theta_{high} > \widetilde{rat}_a(r, s) \geq \theta_{low}$, with high confidence
- **class 3** - high suitability, i.e. $\widetilde{rat}_a(r, s) \geq \theta_{high}$, with high confidence

Since the suitability that is acceptable for a service consumer is user-specific, the thresholds θ_{low} and θ_{high}, with $\theta_{low} < \theta_{high}$, are consumer-specific and can be adjusted by the user. Also the confidence threshold that distinguishes high confidence from low confidence results can be adjusted (Requirement F.8). By referring to classes instead of single ratings, we achieve that prediction errors are hidden to some degree. Moreover, it can be avoided that the user is overwhelmed by being provided with too much information. The 4 classes are represented by different colors. Because of its commonly known interpretation, we use the traffic light color scheme, i.e. red to mark low suitability (class 1), green to denote high suitability (class 3) and yellow for ratings in between (class 2). The color white is used to indicate that no or insufficient feedback information are available (class 0) (cf. Figure 6.15 left).

By providing feedback information on a per attribute basis, the user can exactly locate the risk that is associated with service execution (Requirement F.7). If an offer is ranked high, due to its description having received a high matching degree with respect to the consumer's request (model), but at the same time the results of the feedback-based judgment prediction indicate a low matching degree, then the risk, that is associated with the execution of the service this offer refers to, is high. This is due to the fact, that it is unlikely, that the service will provide the functionality that has been offered and likely that it will perform worse than expected.

6.7.2. Customizable Feedback-Aware Service Ranking

Besides simply presenting feedback information, we also provide a mechanism that promises to facilitate decision making by allowing feedback information to influence the ranking of available service offers. However, as argued earlier, risk attitudes and thus the impact that feedback information shall have on this ranking differ among users. While one consumer is risk averse with respect to the price of a service, another one is willing to the take risk of having a higher price than offered, if the delivered quality is as promised. To account for this fact, we allow service consumers to specify their risk attitude very precisely, namely attribute-wise, and enable personalized, feedback-aware service ranking based on those preferences. The specified risk preferences control to what degree the predicted judgment for a certain attribute and the matching value will influence the final rank of a service offer. While risk averse people will base their decisions mainly on the predicted judgments (large impact of feedback information on an offer's rank), risk seeking persons will put more trust in the matching value provided by the semantic matchmaker (low impact of feedback information on an offer's rank). This is due to the fact that feedback information, if confidable, provide knowledge about a service's actual performance, whereas the matching degree might be based on the matchmaking of inaccurate service offers and thus might be inaccurate too. Since the ranking mechanism considers attribute-specific feedback information in the user-defined way, it is personalized and customizable to risk attitudes that might vary between different users and even between different request (model) contexts of a single user (Requirement F.8). In the remainder of this section, we will detailedly describe how service consumers can specify their attribute-specific risk attitude and how this is considered when ranking service alternatives.

As illustrated in Figure 6.15 (right), users can specify their risk attitude for each service aspect by indicating to what degree ($0-100\%$) the predicted judgment for this aspect shall influence an offer's rank with respect to this attribute. The value specified per attribute is called the *feedback weight* of this attribute. Necessary adjustments can be done via a slider that is provided for each service aspect, that is specified within the request (model). As an example, consider the service aspect *price* in Figure 6.15. Since the user whose personalized ranking scheme is depicted is risk averse with respect to the price of a service, he indicated that the predicted *price*-judgment shall have a large influence on a service offer's rank with respect to the attribute *price*, namely 70%, while the matching value only marginally influences it (30%). By adjusting the feedback weights, a user might also indicate that the ranking should be exclusively based on the predicted judgments (feedback weight of 100% for all leaf aspects of the request (model)) or not at all (feedback weight of 0% for all service aspects). The latter is the default, if no feedback weight has been specified.

Feedback-aware service ranking is implemented by slightly modifying the uncertain matchmaking procedure introduced in Section 5.6 and sorting alternative service offers according to their resulting modified and feedback-aware matching degrees. In particular, given a service request (model) r as well as a service aspect a and its feedback weight $w_{fb}(a)$, the (expected) matching value $MV_a(r, o_s)$ of service offer o_s with respect to the attribute a and the request (model) r is adjusted as follows

$$MV_a(r, o_s) = (1 - w_{fb}(a)) \cdot MV_a(r, o_s) + w_{fb}(a) \cdot \widetilde{rat}_a(r, s), \tag{6.31}$$

where o_s is the offer of service s. The adjusted attribute-specific matching degree is then provided to the matchmaker and further processed as usual. Obviously, this also works, if we perform standard matchmaking based on DSD requests. The presented ranking mechanism indicates, that the specification of feedback weights has to be restricted as follows. If a user specifies a feedback weight for a certain attribute, he is not allowed to specify a feedback weight for one of its parent attributes in the request (model) tree. This is due to the fact, that the matching degree of the parent attribute is already partly determined by the predicted judgment for the subattribute (which is also an indirect (partial) judgment for the parent attribute) and thus the actual feedback weight for the parent attribute would be higher than required by the user.

Changes to the feedback weights immediately affect the service ranking. The proposed mechanism facilitates a fine-grained specification of a user's risk attitude on the attribute level, but also allows to specify it coarsely, e.g. just for the complete request (model), i.e. the root attribute or not at all.

Part III.

Implementation and Evaluation

7 System Implementation

The developed and implemented service selection and decision support solution, whose architecture is outlined in Figure 7.1, comprises two main components corresponding to its major tasks as identified in Section 1.4. Those are a *Requirements Elicitation and Service Selection Component* (ReqElComp) and a *Feedback Component* (FbComp). Its implementation should be understood as proof of concept and as such does not natively support service invocation, i.e. it is supposed that an *Invocation Component* is made available by external providers. The proposed approach is build on top of the DIANE service description framework and borrows corresponding matchmaking and service discovery functionality from it. The system is implemented as an Eclipse Rich Client Application (RCP)[1], which enables fast development and ensures extensibility and portability of the provided software. Moreover, if required by the usage context, the application can be converted into a Rich Ajax-enabled Web-Application (RAP)[2] with moderate effort. This is due to the fact that both, the Eclipse Rich Client Platform, as well as the Rich Ajax Platform allow to develop applications following the Eclipse development model and share substantial parts of the application programming interface. In the remainder of this section, we will give a brief overview about the basic components of the implementation.

As indicted by its name, the *Requirements Elicitation and Service Selection Component* manages the process of requirements elicitation and service selection that has been introduced and detailedly described in Chapter 5. It assists the consumer in specifying service requirements via the graphical requirements representation (cf. Section 5.5.1) and/or based on the properties exhibited by the matching service offers that are presented to him. As detailed in Section 5.5.2, the former kind of support is provided by suggesting potential service requirements for consideration to the user. Required information are thereby derived from knowledge about the user's past service needs as stored in the *Request Model Database*. The latter kind of support is accomplished by educating the user about recurring properties of the available service alternatives and allowing him to specify not yet considered attribute conditions, refining, i.e. subtyping, an attribute condition's target set type and/or by critiquing one of the listed service offers on the basis of this knowledge

[1]http://www.eclipse.org/rcp/
[2]http://www.eclipse.org/rap/

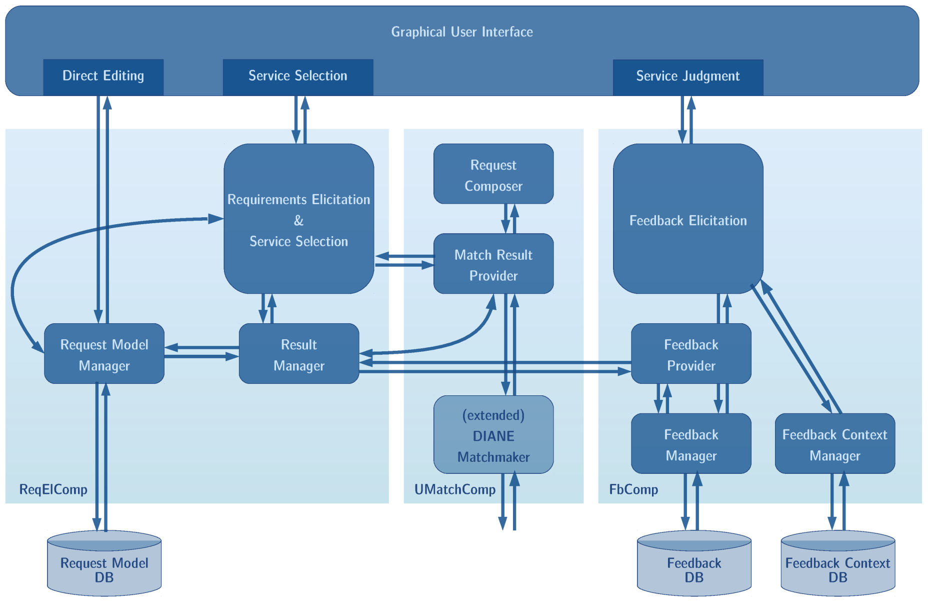


Figure 7.1.: System architecture providing components for the elicitation of consumer requirements and for service selection (ReqElComp), for semantic matchmaking (UMatchComp) and for the elicitation and usage of consumer feedback (FbComp) as well as a graphical user interface

(cf. Section 5.7). To be able to perform this task, the component maintains a request model (cf. Section 5.4), encoding the user's service requirements and preferences, and extracts required knowledge about service characteristics from the retrieved offers' semantic service descriptions. While the responsibility for the request model is borne by the *Request Model Manager*, which also informs registered system components about model changes, the provision of offer-related information is accounted for by the *Result Manager*, which continuously updates the required offer information as the underlying request model changes and keeps registered system components up-to-date. Information extraction and evaluation is enabled by appropriate ranking and clustering utilities. An implementation of the suggested uncertainty measure (Section 5.8.1) allows to identify and emphasize directions for needs exploration that are promising in terms of revealing a large amount of relevant requirements information, if selected, and thus allows for directing and focusing the requirements elicitation process. User interactions shall trigger appropriate changes to the request model (Section 5.9). For that purpose, a number of *interaction types* and *interaction models* have been implemented and registered with the Request Model Manager. While the former indicate available types of user interactions, the latter determine how a certain user action affects the request model and perform the actual update. Additional user-defined interaction types and models can be implemented and simply registered with the system.

The Requirements Elicitation and Service Selection Component works closely together with the *Uncertain Semantic Matchmaking Component* (UMatchComp), that relates characteristics of the available services to the user's uncertain service demands. As indicated, the system leverages external matchmaking and service discovery functionality provided by the DIANE semantic matchmaker for that purpose. As has been discussed in Section 5.6, the latter has been slightly extended to support uncertain matchmaking. The matchmaker's functionality is made accessible to the system by the *Match Result Provider*, which transforms the internal request model into a specific DSD service request description that can be consumed by the matchmaker (cf. Sections 4.2 and 5.6) and finally transfers the results delivered by the matchmaker into a format that is comprehensible for the system. Beside of the Match Result Provider, other, user-specified providers, contributing additional types of information about available service offers, which might be relevant in the context of the request model under construction, can be implemented and registered with the Result Manager. The implementation supports nesting of those providers.

In addition to the matching characteristics and the properties of the retrieved service offers, that are extracted using their semantic service descriptions, the system also provides the user with information about the trustworthiness of those offers derived from consumer-provided service judgments referring to past service interactions (cf. Section 6.7). Those judgments are elicited by the *Feedback Component* described in Chapter 6. Judgment information provided by other system participants is made available by the implementation of a *Feedback Provider*, that delivers consumer feedback related to the current request model and the matching service offers, and has been registered with the Result Provider. To perform the task of feedback elicitation, the Feedback Component utilizes information related to the request model as well as knowledge about the user's judgment preferences acquired during past judgment sessions. The latter is derived from information about the context of each feedback provision (cf. Sections 6.4 and 6.5.2), comprising for instance information about the type and number of judgments provided by the user as well as of information about the underlying request model, which are stored in a *Feedback Context Database* and are maintained by the *Feedback Context Manager*. The *Feedback Manager* is responsible for storing and maintaining the elicited consumer feedback and manages feedback propagation to and acquisition from other users.

The graphical user interface of the implementation is structured along the main tasks of the system - requirements elicitation and service selection as well as acquisition and usage of consumer feedback - and offers appropriate system views and interaction opportunities related to those tasks. In particular, it allows to create, browse, load, store and delete request models. It also supports editing of loaded request models via their graphical representation (cf. Section 5.5.1) in the *Direct Editing Perspective* and allows for exploring retrieved service alternatives and their characteristics (Section 5.7) via the *Service Selection Perspective*. Based on the presented service offers and their properties, the user can specify additional requirements related to those characteristics. He can also control whether information derived from consumer-provided service judgments shall be displayed and to what degree they shall influence the relevance ranking of the displayed service offers

(Section 6.7). Finally, the user can judge an invoked service via the *Service Judgment Perspective* as explicated in Section 6.5.

8 Evaluation of the Requirements Elicitation and Service Selection Mechanism

In this chapter, we will evaluate the requirements elicitation and service selection mechanism that has been suggested in Chapter 5 of this thesis. We start by recalling the evaluation objectives (Section 8.1), which arise from the list of requirements on the model of the user's service needs, on the process of requirements elicitation and service recommendation and on the presentation of information that has been compiled in Section 5.1. While the achievement of some of those objectives can be verified theoretically, for others a user study is required. In the remaining sections of this chapter, we will present the results of those evaluations. In particular, we will theoretically argue that the requirements that have been put on the requirements model, on the specific type of requirements elicitation and service selection procedure, on the way of specifying requirements and on service matchmaking are fulfilled (Section 8.2). After that, we will present the results of two user studies demonstrating the effectiveness and efficiency of our requirements elicitation and service selection procedure (Section8.3) and of the process of specifying service requirements using the graphical requirements representation that has been suggested (Section 8.4). We conclude with a summary of the evaluation results in Section 8.5.

8.1 Evaluation Objectives

The requirements elicitation and service selection procedure that has been introduced in Chapter 5 is supposed to support service consumers in efficiently making well-informed, balanced and consistent service selection decisions (Objective 3) and thus promises to be a key contribution to the achievement of the thesis goal. In Section 5.1, we compiled a list of requirements that need to be satisfied by such a mechanism in order to be able to perform this task and in order to do this effectively. In the following, we will recap those requirements that set the standards for the evaluation of the suggested approach to requirements elicitation and service selection.

Requirements to the Requirements Model

Requirement U.1. (Model contents) *The requirements model should comprise of knowledge about the consumer's service requirements and his preferences.*

Requirement U.2. (Descriptive power) *The mechanism used to model service requirements should provide means to semantically and richly describe desired service effects and should be flexible enough to represent requirements related to different target domains.*

Requirement U.3. (Model uncertainty) *Uncertainty about the consumer's actual service requirements and his preferences should be explicitly represented, measured and located within the requirements model.*

Requirements to Requirements Elicitation and Service Recommendation

Requirement U.4. (Service recommendation) *The system should be able to provide service recommendations and personalized assistance based on uncertain requirements.*

Requirement U.5. (Model construction) *The processes of requirements elicitation and service selection should be unified and incremental.*

Requirement U.6. (Incentives) *The system should provide incentives to think about and to express (correct) preferences and requirements and should encourage decision makers to think thoroughly about and to make tradeoffs.*

Requirement U.7. (Requirements specification) *The system should allow for requirements specification based on presented service alternatives.*

Requirement U.8. (Requirements awareness) *The system should contribute to the user's comprehension and awareness of his service requirements.*

Requirement U.9. (Model consistency) *The system should maintain consistency between the requirements model and the user's actual service requirements and preferences.*

Requirement U.10. (User involvement) *The user should be involved in the model construction process and should be enabled to interactively contribute to the quality of the system's requirements model.*

Requirement U.11. (Uncertainty reduction) *The system should effectively reduce uncertainty about the consumer's service requirements. It should thereby focus on parts of the model that are relevant in light of the available service options and in light of the user's requirements and preferences.*

Requirement U.12. (User-initiated actions) *Expressions of preferences and requirements should be user-initiated and should be effectively guided by the system.*

Requirement U.13. (Process coherence) *Guidance should result in a process of requirements elicitation and service selection that is coherent.*

Requirement U.14. (User education) *The system should educate users about relevant service alternatives and their characteristics and motivate them to consider this knowledge when making a selection.*

Requirement U.15. (Selection efficiency) *The process of service selection should be efficient, i.e. made within an appropriate period of time and with adequate mental effort.*

Requirements to the Presentation of Information

Requirement U.16. (Information presentation) *The system should emphasize relevant and important information and present them in a format that makes them easy to perceive and easy to comprehend.*

Requirement U.17. (Selection consistency) *The system should support the user in making a selection that is consistent with his requirements.*

8.2 Partial Validation of Requirements Achievement

In this section, we will theoretically verify that the requirements model that has been suggested in Section 5.4 fulfills the requirements that have been put on it (Requirements U.1-U.3). Moreover, we will show that the requirements elicitation and service selection procedure that has been proposed in this thesis is designed in the required way, i.e. fulfills the Requirements U.4, U.5 and U.7.

8.2.1. Appropriateness of the Requirements Model

As an extension of standard DSD request descriptions, the requirements model that has been introduced in Section 5.4 of this thesis inherits the properties of the latter and as such is capable of semantically and richly describing a consumer's service requirements in terms of desirable service effects. It also offers language constructs for specifying preferences (over attribute values and preferences referring to the importance of service aspects) (Requirement U.1) and is flexible enough to model service requirements related to different target domains (given that appropriate domain ontologies are available) (Requirement U.2). Uncertainty about the consumer's service requirements and his preferences is explicitly represented and located in the requirements model and measured (Requirement U.3). In particular, uncertainty about the acceptability of certain attribute values and the relative importance of attributes is modeled by means of probability distributions (cf. Section 5.4). By attaching these to the respective attribute or requirement they refer to, uncertainty can be located within the model. Finally, a measure has been presented in Section 5.8.1 that quantifies the uncertainty related to a given request model. Hence, summarizing, we can state, that the requirements model that has been suggested in this thesis fulfills the requirements that have been put on it (Requirements U.1-U.3).

8.2.2. Appropriate Kind of the Requirements Elicitation and Service Selection Procedure

The requirements elicitation and service selection procedure that has been introduced in Chapter 5 is unified and incremental (Requirement U.5) in the sense of interweaving requirements elicitation and service selection into a joint process, which aims at incrementally acquiring knowledge about the service consumer's service requirements and preferences by alternating phases of intermediate service recommendation based on partially known requirements and phases of requirements specification and refinement based on the presented service alternatives (Requirement U.7). The latter feature is detailedly discussed in Section 5.7. The described way of specifying service requirements and making service selections is enabled by the system's ability to provide service recommendations and personalized assistance based on uncertain requirements (Requirement U.4). While the former is implemented by the matchmaking procedure that has been discussed in Section 5.6, the latter is grounded on the mechanisms that have been particularized in the Sections 5.7 and 5.8. They allow to determine and suggest interaction opportunities, i.e. service aspects and subtypes to choose for consideration as well as selectable tradeoff opportunities that are promising in light of the user's known requirements and promising in light of the available service options. As a result, we can state, that the requirements elicitation and service selection procedure, that has been proposed in this thesis, fulfills the Requirements U.4, U.5 and U.7.

8.3 Evaluation of the Requirements Elicitation Process

In the evaluation of our approach, we wanted to find out, whether end-users were able to formulate their service needs by using our system, whether they were able to find the service functionality they desire and whether they feel supported in that task. In particular, we were interested in whether the proposed system helped users to effectively construct their service requirements and whether this was possible within an appropriate period of time. In the remainder of this section, we will first introduce the methodology that has been applied to evaluate our requirements elicitation and service selection approach (Section 8.3.1). After that, we will describe the user study and the underlying test data that have been used for the evaluation (Section 8.3.2). Finally, we will detailedly discuss the evaluation results (Section 8.3.3).

8.3.1. Evaluation Methodology

The methodology that has been used to evaluate our approach to requirements elicitation and service selection is an extension of the evaluation design applied in [CP07b]. Figure 8.1 provides an overview about the approach that proceeded as follows. The test users were first asked to think about a specific service they would like to use. They were not completely free in their choice, but had to choose from given service categories. The participants were then provided with a questionnaire (*Pre-System-Usage-Questionnaire*) containing questions related to their background (age, gender, knowledge, ...) and about their initial service requirements with respect to the chosen service (MM_{init} in Figure 8.1). After a 5-minutes introduction to our system, the users were asked to use it to learn more about their requirements with respect to the service they would like to use and to identify the service offer that best suits to these service requirements (Sel_{use} in Figure 8.1). Thereby, the test users specified their service requirements exclusively based on the presented service alternatives, i.e. did not use the graphical requirements representation to directly specify service requirements, and selected a service out of a given collection of service offers. To make the choice more difficult, all offers presented to the user were taken from the selected service category. The users started with an empty request model, i.e. no specified attributes. Once a user selected an offer, he was asked to complete a second questionnaire (*Post-System-Usage-Questionnaire*) comprising of questions about his (updated) service requirements (MM_{use} in Figure 8.1), his confidence in the specified requirements and the appropriateness of the selected service and questions related to the usefulness of the provided system.

In order to verify the quality of the elicited service requirements and the service selection decision that has been made by the user, we provided him with a scrollable table containing a list of all available service offers and all of their properties (all that are known to the system). Thereby, each column corresponded to a particular service property and was sortable. The participant was then asked to look through this list and check whether there is an offer other than the selected that fits better to his requirements (Sel_{exh} in Figure 8.1). Note, that this offer has not necessarily to be the offer that best suits to the user's service requirements. We are just interested in whether the user selected another offer or not. In case of a switch, the user was asked about the reasons for this (*Post-Browsing-Questionnaire*). Note, that this could be either due to a yet unconsidered service alternative or due to a yet unconsidered service requirement that the user became aware of by browsing through the provided list. In the latter case, the user is also asked about his updated service requirements (MM_{exh} in Figure 8.1). Note, that the task of verifying whether a given offer is actually the best choice is much less mentally demanding than selecting the best offer out of a set of offers (the original service selection task). Hence, presuming that the number of available offers is kept reasonably small, it can be assumed that the test users performed this task correctly and not just sticked to their original choice.

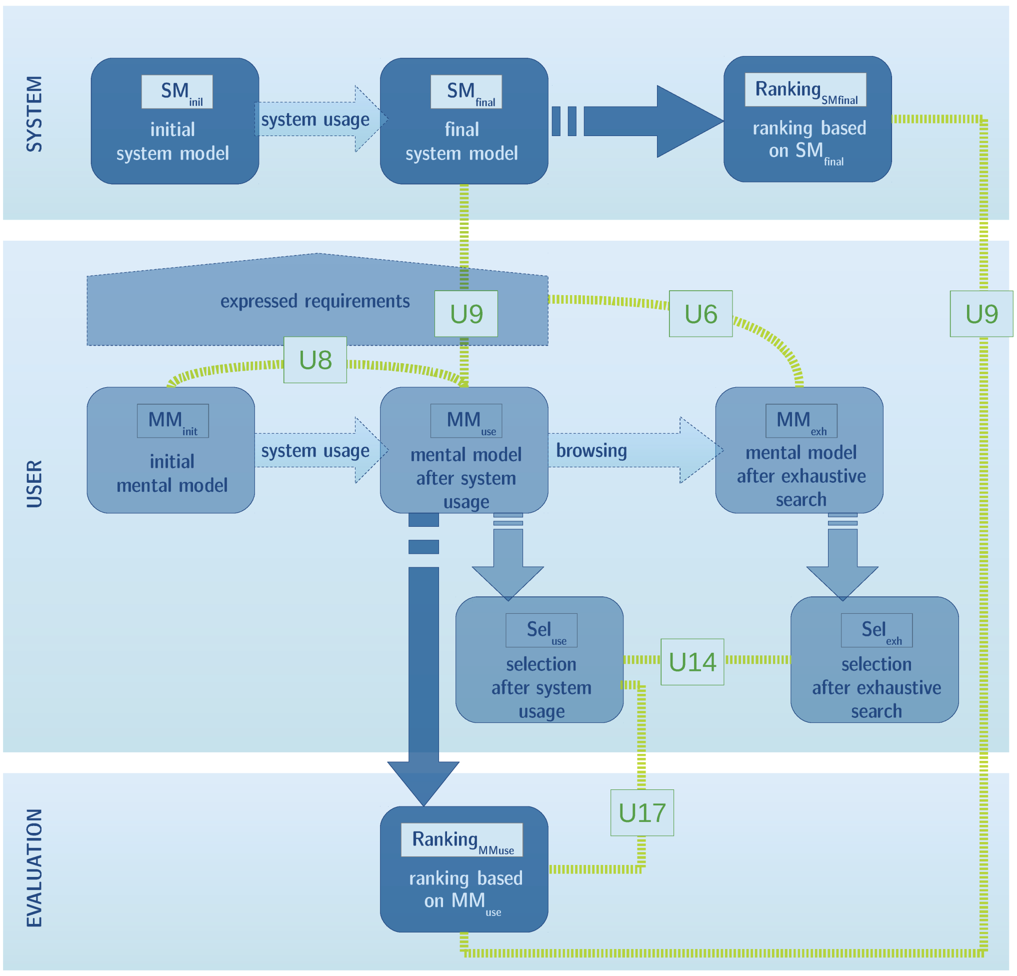


Figure 8.1.: Evaluation methodology - the dark blue boxes refer to information that are system-generated and/or provided, light blue boxes refer to information that are user-generated and/or provided, the green lines indicate which information have to be compared to verify the indicated requirement

During the test, the user's interactions with the system as well as the state of the internal request model (SM_{init} to SM_{final} in Figure 8.1) was logged to have this information for the evaluation. As we will see, by comparing pairs of the recorded requirements models and/or the rankings of the available service alternatives that result from them, we can objectively verify different properties of the proposed system and thus decide, if the requirements that have been stated in Section 5.1 are fulfilled. The subjective impressions of the test users acquired using the questionnaires supplement these results.

8.3.2. Test Data and Test Setting

Following a preliminary evaluation of our approach with 4 participants that has been published in [KKR11], we performed an evaluation with 10 test users (6 males and 4 females). 9 of them previously used services via the Web, but only 2 of them were familiar with Web Services. Table 8.1 summarizes the demographic information about the test participants as well as information about their background knowledge.

Aspect of concern	**Frequency/Range**
demographic information	
age	25 – 58 years
gender	6 male, 4 female
background knowledge/skills	
scientific background	90%
computer science background	70%
previously use services via the Web	90%
familiar with	
computer items and their properties	80%
Web Services	20%
object-oriented programming	70%
generalization/specialization	100%
taxonomy	60%
ontology	70%

Table 8.1.: Demographic information and background knowledge of the test participants

Data set Evaluating our approach to service selection required a set of service offers from which to select. This offer set should ideally comprise of

- real world services with corresponding semantic offer descriptions and
- should refer to kinds of services that are typically used by end-users.

To emphasize the advantages of our approach compared to the traditional research and online purchasing approach and to demonstrate its effectiveness (cf. Section 1.3), service offers should

- refer to a service domain that is complex in terms of the number of service properties and their potential interdependencies and
- complex in terms of the number of available service alternatives.
- Moreover, the selection decision should be important in terms of resulting in some kind of loss for the decision maker, if not solved properly.

Finally, to derive information about the quality of the requirements model created by our system, which are required for the evaluation of the performance prediction procedure that has been proposed in Chapter 6 (Section 9.4.5), we had to base both evaluations on the same set of services (cf. Sections 9.4.1 and 9.4.2).

Though a few Semantic Web Service test data collections for evaluation purposes exist (e.g. OWLS-TC [1], they did not meet our requirements. In particular, they often comprise of a relatively small set of rather simple services (e.g. converter or location services) that are also sometimes somewhat unrealistic and artificial. Hence, to evaluate our approach, we had to create a new data set, that served as a base for our user study. To have at least a realistic set of services, we extracted structured information about computer items from a major online seller to generate semantic descriptions of services selling computer items. We chose this kind of services, since it is typically used by end-users. Moreover, it is sufficiently complex in terms of the number of available service alternatives and in terms of the number of interrelated service properties that have to be considered and important in terms of resulting in monetary loss if not selected properly. One could argue that using a very specific class of services, i.e. those selling products, for our evaluation affects the significance of our results. However, our approach does not make any assumptions on the type of services it is applied to. Hence, as long as the underlying service domain can be appropriately described using DSD service descriptions, our approach is applicable as well. Nonetheless, it would be worthwhile to evaluate our approach with test data from a more service-focused domain such as flight booking or hotel reservation. Unfortunately, at the time of our evaluation, structured data from these domains were hard to obtain. This recently changed, which offers new opportunities for future research.

For the purpose of our evaluation, we generated service offers from the 8 categories *desktop PC, server, digital watch, e-book reader, PDA, organizer, notebook* and *electronic dictionary* (cf. Section 9.4.2 for details). The test users were allowed to choose one of these categories and were asked to think about desirable properties of a service that sells a computer item of the selected category. During the test, the study participants had to choose from a collection of 200 services from the selected category. This number was chosen to make the decision sufficiently difficult to demonstrate the effectiveness of our approach, but at the same time keeping the mental effort required for the exhaustive search that had to be performed after system usage at an acceptable level.

[1]http://projects.semwebcentral.org/projects/owls-tc

Questionnaire formulation and evaluation Questions in the questionnaires were formulated as statements, where the users had to indicate their level of agreement using the scale -2 (strongly disagree), -1 (disagree), 0 (neither agree nor disagree), 1 (agree), 2 (strongly agree). The presented evaluation results refer to the percentage of users that agreed/disagreed with a considered statement, which is indicated in brackets, where necessary. This percentage is accompanied with the significance level p, i.e. the probability of the result being not statistically significant, resulting from a chi-squared test[2]. If the same statement had been posed several times during the evaluation (this holds for example for statement U8-4 referring to the user's confidence about his service requirements), we also used a chi-squared test to assess whether there is a significant difference in the answer distributions. In those cases, the significance level is also given in brackets. The questionnaires have been carefully designed to avoid typical biases in the responses (see e.g. [FA99, Fra96, TP98, CP05] for a discussion). In particular, some of the statements were stated multiple times, but in a different way to avoid framing effects, i.e. a bias of the answers caused by the formulation of the statements. Hence, note, that depending on the way a statement is given, agreement or disagreement of the test users has to be interpreted differently with regard to the fulfillment of a certain system property. Finally, the rating scale (a Likert-type scale [lik32]) was chosen and presented in a way that reduces biases in the judgments. The complete questionnaires can be found in Appendix B.

Acquisition of the test users' service requirements As already mentioned, we acquired knowledge about the study participants' service requirements at several stages of the evaluation procedure, namely once at the beginning of the study (MM_{init} in Figure 8.1), once after a participant used our system to select a service (MM_{use} in Figure 8.1) and once after having looked through the entire list of available services (MM_{exh} in Figure 8.1). For this purpose, the users were asked to indicate the service aspects that are important to them. They were also asked whether they have specific requirements on these aspects, such as *"the price should be lower than 1000 €"* or *"the more the better"*. We did not restrict the users in the way and type of these requirements. Finally, we asked the participants to weight the indicated service aspects against each other. Since simply assigning a numerical weight to each of these aspects is likely to result in an inconsistent and rather arbitrary weighting, we used an approved procedure that is typically used in the Analytic Hierarchy Process [Saa08] to acquire the desired weights. This approach is based on pairwise comparisons of the importance of the considered service aspects and results in a more consistent weight assignment. We used the analysis tool Java Analytic Hierarchy Process (JAHP)[3] by Maxime Morge to acquire the desired weights.

Ranking the available service alternatives based on the requirements models acquired this way was done by evaluating the services with respect to the service aspects that were considered as relevant by the test users. More specifically, we assessed for each relevant

[2]The null hypothesis was a uniform distribution of the judgments. The resulting significance level was inverted, i.e. $p = 1-$ significance level of the null hypothesis.

[3]http://www.lifl.fr/ morge/software

service aspect to what degree the user's requirements on this aspect were fulfilled by the considered offer and then weighted those degrees using the acquired weights to compute the overall rank of the service offer. In particular, the degree of fulfillment was 1 if a certain constraint of the type *"attribute value </<=/>/>=/!=/= value"* was fulfilled and 0 otherwise. For preferences of the type *"the more/less/bigger/... the better"*, we used a linearly increasing/decreasing preference function over the entire range of values of the considered attribute as occurring in the data set referring to the selected service category. Thereby, a fulfillment degree of 0 was assigned to the range minimum and a degree of 1 was assigned to the range maximum. The fulfillment degrees resulting from several constraints/preferences referring to a single service aspect were multiplied. Rankings based on the recorded system models were calculated using the uncertain matchmaking approach suggested in this thesis (cf. Section 5.6).

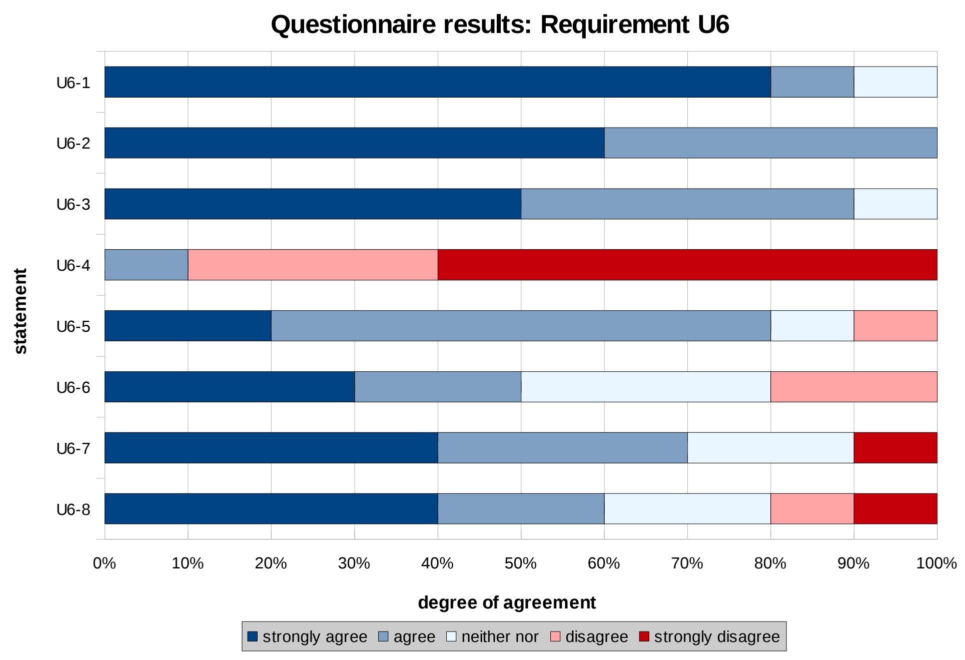
8.3.3. Results

In the remainder of this section, we will discuss the results of our evaluation that has been performed using the methodology and test data described above. In particular, we will conclude, whether the required system properties that have been discussed in Section 5.1 are fulfilled or not.

Incentives for requirements construction and expression As indicated by the test users (Figure 8.2), the system succeeded in facilitating the specification of service requirements. The questionnaire respondents agreed, that it was easy to indicate service aspects that are of interest to them (90% agreed with statement U6-1, $p < 0.01$), that it was easy to refine the type of a service aspect (100% agreed with statement U6-2, $p < 0.01$) and that it was easy to specify constraints on a service aspect's values by critiquing a selected service alternative (90% agreed with statement U6-3, $p < 0.01$). The test users clearly disagreed with the statement of having not been able to specify the requirements and preferences they wanted to express (90% disagreed with statement U6-4, $p = 0.01$). An inspection of the logged session interactions reveals that the proposed system succeeded in stimulating the test users to construct and specify service requirements. In the course of requirements specification, the test users expressed requirements on averagely 4.4 ± 0.84 service aspects, refined the type of averagely 1.8 ± 0.26 service aspects and made use of the opportunity to critique available service alternatives 3.7 ± 0.51 times in average (cf. Table 8.2). Among the means for requirements specification that are offered by the proposed system, the respondents considered the provided tradeoff opportunities to be most helpful to identify yet unconsidered, but important service requirements (30% of the participants), followed by the displayed service attributes (20% of the participants)[4].

However, we wanted to find out, if the test users not just expressed service requirements, but indicated correct requirements. To verify that, we compared the service requirements

[4]50% of the participants did not comment on this issue.



ID	Statement/Question
U6-1	It was easy to add a new service aspect to my requirements.
U6-2	It was easy to refine the type of a service aspect.
U6-3	It was easy to specify constraints on an attribute's values by critiquing a selected service alternative.
U6-4	I was not able to specify the requirements and preferences I wanted to express.
U6-5	The service aspects recommended for consideration pointed me to service aspects that are important to me and that I did not yet consider.
U6-6	The recommended type refinements for already specified service aspects pointed me to yet unconsidered requirements.
U6-7	The recommended tradeoff alternatives induced me to balance between my competing requirements.
U6-8	The presented service alternatives induced me to indicate additional requirements and preferences.
U6-9	Which recommender was most helpful to you?

Figure 8.2.: Questionnaire statements/questions and results related to Requirement U.6

that had been indicated by the study participants after having inspected all available service alternatives based on all of their known properties (MM_{exh} in Figure 8.1), and the requirements expressed during the system interaction. Thereby, we restricted ourselves to those participants that did not change their requirements after having looked through the

Aspect of concern	**Frequency**
use of interaction opportunities	
specified service aspects	4.4 ± 0.84
type refinements	1.8 ± 0.26
choosing a tradeoff opportunity	3.7 ± 0.51
correctness of the expressed requirements	
% of the specified service aspects that are in fact relevant	$84\% \pm 11\%$
% of the specified constraints that are in line with the user's requirements	$91\% \pm 13\%$
% of the chosen tradeoff opportunities that are in line with the user's requirements	$100\% \pm 0\%$

Table 8.2.: Evaluation results related to Requirement U.6

entire list of available services (90% of the test users). The latter ensures, that the model to which the user's expressed requirements are compared to is correct and complete. As it turned out, the questionnaire respondents did not just express service requirements, but thought thoroughly about them and subsequently specified correct service requirements, i.e. those they actually have. In particular, $84\% \pm 11\%$ of the service aspects that had been specified using the system were in fact relevant to the user, $91\% \pm 13\%$ of the specified constraints and all tradeoff opportunities that had been chosen during the test sessions were in conformance with the user's actual service requirements (cf. Table 8.2).

The evaluation results clearly indicate that the proposed system effectively stimulates the construction of correct service requirements. The questionnaire responses confirm that, but reveal that not all presented information and interaction opportunities supported the users equally well in that task. The recommendation of service aspects for consideration and the critiquing tool were considered most helpful. In particular, 80% of the test users agreed with the statement that the service aspects recommended for consideration pointed them to yet unconsidered but important service aspects (80% agreed with statement U6-5, $p = 0.02$) and 70% indicated that the recommended tradeoff alternatives induced them to balance between competing service requirements (70% agreed with statement U6-7, $p = 0.03$). However, only 50% agreed that the recommended type refinements for already specified service aspects pointed them to yet unconsidered requirements (50% agreed with statement U6-6, $p = 0.26$) and only 60% agreed with the statement of being encouraged to indicate additional requirements and preferences by viewing the presented service alternatives (60% agreed with statement U6-8, $p = 0.16$). Agreement to these two statements was also not significant.

We argue that the suggested approach to requirements elicitation effectively stimulated the test users to think thoroughly about and to construct their service requirements, that it encouraged and enabled the respondents to express correct requirements and preferences and did not hamper this task by making it difficult to perform. It thus fulfills Requirement U.6.

Contribution to the comprehension and awareness of service requirements

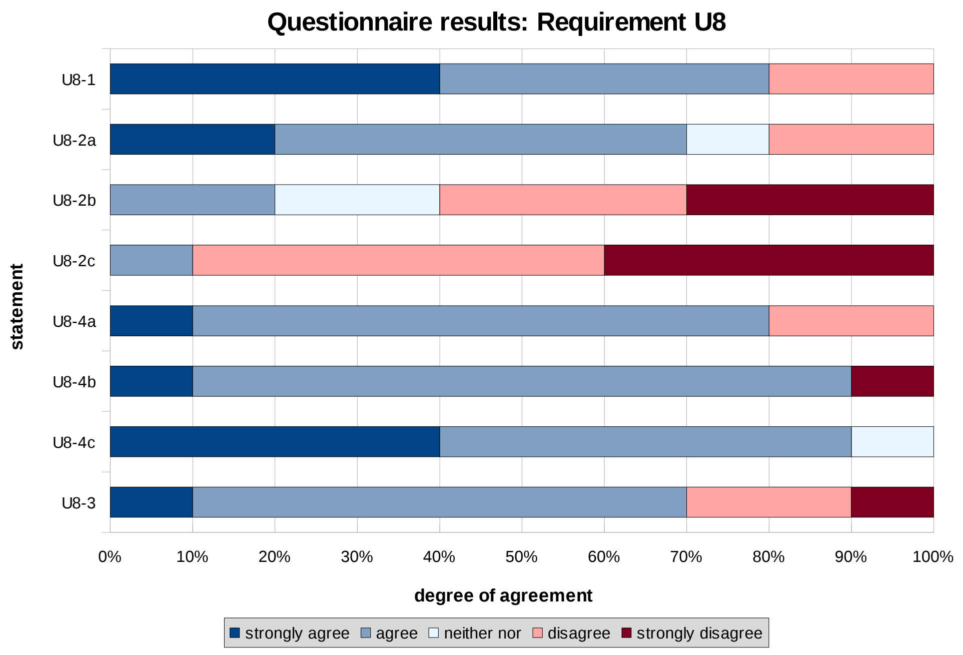
A comparison of the initial requirements specified by the participants (MM_{init} in Figure 8.1) and those they provided after having chosen a service by using the tool (MM_{use} in Figure 8.1) reveals that the proposed system successfully contributed to the test users awareness of their service requirements. After having made their final choice, the respondents were able to indicate requirements on averagely 1.55 ± 0.87 service aspects, they have not been aware of before. They abandoned requirements on averagely 0.34 ± 0.46 service aspects, that turned out to be of marginal relevance in light of the available service alternatives, were able to indicate additional constraints on averagely 1.11 ± 0.51 service aspects and, in average, revised constraints on 0.78 ± 0.85 service aspects. 70% of the participants changed the relative importance of their requirements related to the values of the considered service aspects (cf. Table 8.3). Just one of the questionnaire respondents

Aspect of concern	Frequency
ability to indicate requirements	
indication of previously non-aware-of service aspects	1.55 ± 0.87
detection of irrelevant service aspects	0.34 ± 0.46
indication of additional constraints	1.11 ± 0.51
revision of constraints	0.78 ± 0.85
change of requirements importance	70%

Table 8.3.: Evaluation results related to Requirement U.8

detected additional requirements (on a single attribute) after having inspected all available service alternatives based on all of their known properties. This strongly indicates that the proposed system successfully educated the respondents about the service requirements that are important to them and contributed to their comprehension and awareness of these.

The subjective impression of the degree of support provided by the system reflected this fact (cf. Figure 8.3). The test users clearly agreed with the statement of having learned more about their service requirements and preferences by using the system (80% agreed with statement U8-1, $p = 0.06$). Finally, we indirectly assessed whether the system contributed to the test users' awareness of their requirements by asking them how confident they are about their requirements and if they believe that the requirements they are aware of are complete, i.e. if there are no missing requirements. As it turned out, 70% of the respondents agreed that using the system contributed to the users' confidence in their requirements (70% agreed with statement U8-3, $p = 0.21$). However, this result was not significant. This is reconfirmed by the fact that the respondents' confidence in their requirements before using the system (80% agreed with statement U8-4a, $p = 0.06$) had increased after using the system (90% agreed with statement U8-4b, $p = 0.01$). However, due to the fact, that the test user's confidence in their requirements was high, even before using the system (even though this was not justified, as we demonstrated), this result was not significant ($p = 0.29$). The respondents' confidence about their service requirements was significantly ($p < 0.01$) higher than initially after having viewed all available service alternatives (90% agreed with statement U8-4c, $p < 0.01$), i.e. having viewed all available service offers and their properties further increased the test users' confidence. Looking



ID	Statement/Question
U8-1	By using the system, I learned more about my requirements and preferences.
U8-2	There might be additional requirements that are important to me and that I am not aware of yet.
U8-3	I feel more confident about my requirements than before using the system.
U8-4	I feel confident about my requirements.

Figure 8.3.: Questionnaire statements/questions and results related to Requirement U.8

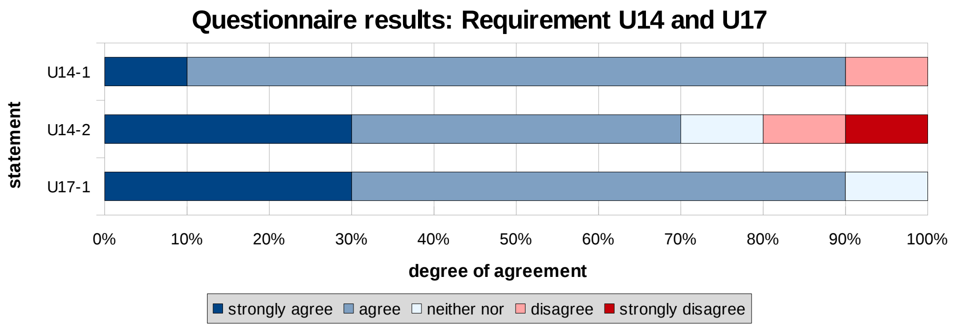
at the test users' subjective impression of the completeness of their constructed requirements, we observed a much more distinct effect. When asking the survey participants whether they believe that there are additional requirements that are important to them, but that they are not aware of yet, 70% clearly agreed before using the system (70% agreed with statement U8-2a, $p = 0.1$). After having used the system to select a service only 20% agreed with that statement (80% disagreed with statement U8-2b, $p = 0.16$). Hence, the system was successful in convincing people of the fact, that they are aware of all their requirements ($p < 0.01$). Viewing all available service alternatives and their properties, again further contributed to the users' confidence in the completeness of their requirements (90% disagreed with statement U8-2c, $p = 0.01$), but not significantly ($p = 0.16$).

We have shown that the system effectively contributes to the test users' comprehension and awareness of their service requirements and that these requirements are in fact correct and complete. Hence, Requirement U.8 is fulfilled.

Effective consumer education and selection consistency By displaying the service attributes and subtypes that have been specified in the available service offers (along with the percentage of offers that specifies them), the decision support system, that has been proposed in this thesis, educates the user about service aspects and subtypes that are worthwhile to be considered. It also points the user to those of his service requirements that might be conflicting with respect to the available offers by informing him about common properties of the offers that fulfill a certain critique. Moreover, the system assigns a high rank to offers that are relevant to the user, i.e. that best fit to his known requirements and therefore should be considered. The sortable results table, that presents available service alternatives and their properties, is structured according to the service aspects that are of interest to the user and thus facilitates the comparison of service alternatives with respect to the user's standards. It therefore supports him in making selection decisions that are consistent with the user's service requirements.

In fact, 90% of the test users confirmed, that the system made it easy to compare available service alternatives and their characteristics with respect to their standards (90% agreed with statement U17-1, $p < 0.01$). After having viewed all available service alternatives, just one of the test persons switched to another service due to having found a service alternative that better fits to his requirements than the originally selected (comparison of Sel_{use} and Sel_{exh} in Figure 8.1). This strongly indicates, that the test participants have been effectively educated about available service alternatives by the system. The subjective impression of the test users confirms this. As can be seen in Figure 8.4, 90% of the users agree with being educated about relevant service alternatives and their characteristics after using the system (90% agreed with statement U14-1, $p = 0.01$), 70% agree that the offered tradeoff opportunities educated them about conflicting requirements (70% agreed with statement U14-2, $p = 0.1$). At this point, we would like to note, that the latter does not mean that 30% of the test users did not consider the offered tradeoff opportunities as helpful. The reason for this is, that the system aimed at educating the user about common characteristics of the service offers that fulfill a given critique. Those characteristics might, but not necessarily do conflict with the user's service requirements. As it turned out, those users that did not agree with statement U14-2 considered the critiquing tool as helpful, but simply did not encounter conflicting requirements by using it.

However, effective decision making does not only require the user to consider relevant service alternatives and their properties and to carefully resolve conflicting service requirements, it also necessitates the decision maker to evaluate available offers with respect to his service requirements and finally make a selection the that fits well to these requirements (Definition 1.1). To find out, whether the test participants' selection decisions were in fact consistent with their service requirements, we determined the rank of the offer that had been selected by a test user using our system (Sel_{use} in Figure 8.1) with respect to the



ID	Statement/Question
U14-1	I feel well educated about relevant service alternatives and their characteristics.
U14-2	The presented tradeoff alternatives educated me about conflicting requirements.
U17-1	It was easy to compare available service alternatives and their characteristics with respect to my standards.

Figure 8.4.: Questionnaire statements/questions and results related to Requirement U.14 and Requirement U.17

user's actual service requirements, i.e. those indicated by the user after having used the system (MM_{use} in Figure 8.1). We restricted ourselves to those users, that did not had hidden requirements they discovered after having viewed all available service offers and their characteristics (90% of the study participants), since decisions made on partially known requirements are inconsistent by definition (Definition 1.1). As it turned out, the test user's selection decisions were in fact consistent with their service requirements. More specifically, the mean rank of the selected service alternative with respect to the user's actual requirements was 8.33 ± 5.33 (out of 200).

This shows, that the system successfully educates service consumers about relevant service alternatives and their characteristics, motivates them to consider this knowledge when selecting a service and effectively supports them in making a selection that is consistent with their service requirements. Thus, the Requirements U.14 and U.17 are fulfilled.

Model consistency As a result of our evaluation, we found that the conformance between the user-specified requirements indicated after system usage (MM_{use} in Figure 8.1) and the system-maintained requirements model at the selection time (SM_{final} in Figure 8.1) was high (cf. Table 8.4). In all cases, the internal request model covered all service aspects that were important to the user. Also the conformance between the relative importance of those aspects as indicated by the test person and that, which was documented in the model, was high. We used two different measures to evaluate the consistency of the importance weights: (1) the mean difference of the service aspects' relative weights in the user-provided model and the system model and (2) the weighted mean difference of the specified service aspects' rank in the user specified model and in the system-maintained model (we assigned the ranks 1 (highest), 2, . . .). Thereby, service aspects where weighted by the inverse of their rank, i.e. $1/rank$. This means, the penalty for a wrong rank was higher for service aspects that were more important to the users. As it turned out, the mean weight discrepancy was 0.12 ± 0.03, the mean difference between the user specified and the modeled ranks was 1.06 ± 0.48.

Aspect of concern	**Value**
model consistency	
attribute coverage	100%
mean weight discrepancy	0.12 ± 0.03
mean difference of attribute rank	1.06 ± 0.48
mean rank difference of top ten offers	30.98 ± 8.50
mean rank difference of top ten offers (CR < 0.1)	23.04 ± 7.88
mean rank difference of top ten offers (CR > 0.1)	38.92 ± 11.89
weighting consistency	
mean consistency ratio before system usage	0.21 ± 0.10
mean consistency ratio after system usage	0.14 ± 0.07

Table 8.4.: Evaluation results related to Requirement U.9

Much more interesting than the discrepancies between the models themselves is the effect these inconsistencies have on the ranking of available service alternatives, i.e. its effect on the recommendation quality of the system. To assess this, we determined the top ten offers with respect to the user-indicated requirements model (MM_{use} in Figure 8.1), i.e. the service alternatives that actually fit best to the user's service needs, and calculated the mean difference between these offers' rank with respect to the user-indicated model and the system-maintained model (SM_{final} in Figure 8.1). As we found, the mean rank difference was 30.98 ± 8.50, i.e. about 15% of the maximal possible rank difference (200). Though this result shows, that there is in fact a high degree of consistency between the models and the resulting rankings, it implies that relevant service alternatives can be assigned such a low rank by the system that causes them to be out of the user's viewing range in the results table (in the test setting, the first 28 service alternatives were visible without scrolling).

Investigating the origin of the inconsistency in both, models and rankings, revealed, that it lies in the inconsistency of the relative weights that the test users assigned to the single service aspects of interest and that have been retrieved using the Analytic Hierarchy Process (AHP) [Saa08] (cf. Section 8.3.2). Though weights that are elicited using the AHP, i.e. by pairwise comparing the importance of service aspects, are typically more consistent than weights that have been acquired by simply asking the user to indicate a weight for each service aspect, they are still inconsistent to some degree. For instance, a user might indicate that aspect A is much more important than aspect B and aspect C is slightly less important than aspect B. Finally, indicating that A is slightly more important than C is not consistent with the two former statements. Saaty, who proposed the AHP, provided a measure for the consistency of weightings derived using the AHP, the *consistency ratio CR*. The lower the CR-value, the more consistent the determined weighting. According to Saaty, the consistency ratio should not exceed 0.1 to be usable for decision making [Saa08]. Though by using our system the mean consistency ratio over all user-indicated weightings (those of MM_{use}, not the system-determined!) decreased from 0.21 ± 0.10 (MM_{init}) to 0.14 ± 0.07 (MM_{use}), 50% of the weightings exceed the CR-threshold of 0.1, i.e. were too inconsistent to be usable. Hence, though the above results referring to the ranking consistency indicate a tendency, they are of limited significance. To find out, whether the observed inconsistency between the service rankings based on MM_{use} and SM_{final} arise from the inconsistency of the user-provided weightings (i.e. the users' inability to express their true preferences related to the relative importance of service aspects) or is in fact due to the inconsistency of the system-maintained requirements model, we determined the mean rank difference of the top ten service alternatives separately for the 5 fairly consistent ($CR < 0.1$) and the 5 inconsistent weightings obtained by asking the test users after system usage. As it turned out, the mean rank difference was lower, namely 23.04 ± 7.88, for the more consistent weightings and higher than that determined over all obtained weightings, namely 38.92 ± 11.89, for the inconsistent weightings. This seems to indicate, that the observed ranking difference does not exclusively result from model inconsistency, but is partially attributed to the test users' inability to express the true relative importance of the service aspects in terms of (consistent) weights. Thus, the actual consistency between the rankings is supposed to be higher. This hypothesis is supported by our finding (see above), that people actually found their desired service by using our system. However, due to the lack of appropriate means to precisely measure the test users' true service requirements and preferences (with reasonable effort[5]), it cannot be verified.

Summarizing our findings, we can state, that there is in fact a high degree of consistency between the user's actual service requirements and preferences and the system-created requirements model, both, in terms of aspect coverage and in terms of the resulting service rankings. However, due to the fact, that a user's requirements, particularly the relative importance of service aspects, cannot be accurately acquired with reasonable effort, the actual degree of conformance cannot be precisely determined. Hence, the fulfillment of Requirement U.9 can only be partially verified.

Effective uncertainty reduction In Section 5.8, we provided a measure that quantifies the uncertainty about the actual matching degree of an offer with respect to a given requirements model that results from missing knowledge about a user's service requirements. We already argued, that this measure is designed in a way that accounts for the relevance of information in light of a user's known requirements and in light of the available service offers' properties. This means, if given a user's known service requirements,

[5]see e.g. [Cle91] for other weight assessing methods

a certain piece of requirements information is of minor relevance for the calculation of an offer's matching degree with respect to the user's requirements, the uncertainty resulting from the lack of this information is low. Similarly, if given the system's knowledge about a certain service offer, a certain piece of requirements information is of minor relevance for the calculation of this offer's matching degree with respect to the user's requirements, the uncertainty resulting from the lack of this information is low. Thereby, a piece of information is relevant for the calculation of a matching degree, if it has the potential to contribute a large amount to it. By emphasizing those interaction opportunities that, if applied, decrease the measured uncertainty of a given requirements model with respect to the k best, i.e. highest ranked and thus most promising, service offers by a large degree, we encourage the user to choose those interaction opportunities and thus to effectively reduce model uncertainty.

However, we wanted to find out whether uncertainty reduction also resulted in savings in terms of steps to take to identify the offer that is most suitable to a user. In particular, we wanted to compare (1) the number of interaction steps with the system (number of add attribute/compromise/refine type interactions) that were required to identify a certain offer when always following the system's most promising interaction suggestion (that complies with the user's requirements) with (2) the number of steps required, if always the least promising (in terms of its ability to reduce model uncertainty) choice was taken and (3) with the number of steps that were actually required by the test users. Since it would have imposed an unacceptable effort to the test users, if we had asked them to perform the same service selection task three times, we let a program perform the service selection tasks (1) and (2) automatically. In particular, taking a given test participant's final selection as the target choice[6], the program interacted with the system in the indicated way. If both, a type refinement or adding an attribute could be performed, we randomly selected one of them with equal probability. A critiquing interaction was always performed, if applicable and always referred to the top-ranked offer. Finally, we measured the number of interaction steps, that were required to have the target offer ranked among the top ten.

As it turned out, choosing always the most promising option recommended by the system significantly reduced the mean number of required interaction steps from 5.8 to 4.5 (cf. Table 8.5) for the tests that always took the least promising interaction option (t-test with significance level 0.07). The number of required steps was not just decreased in average. Also the maximal number of steps that was required for any test run could be reduced (from 10 to 6) as well as the variation among the number of required steps for the single target offers fell, i.e. the standard deviation significantly decreased from 2.44 to 1.08 (F-test with significance level 0.05). This is desirable, since it indicates that by following the system's suggestions (if compliant with the user's requirements), the number of interaction steps to take is reliably reduced to a low level.

However, we were also interested in whether the test users actually followed the system's suggestions (if compliant with their service requirements) and thus benefited from them.

[6] We could not rely on the requirements model indicated by the test participants (MM_{use}), since as argued in the previous paragraph, it did not perfectly comply with the user's actual requirements.

Aspect of concern	Value
mean number of interaction steps	
when always choosing the least promising interaction	5.8 ± 2.44
when always choosing the most promising interaction	4.5 ± 1.08
taken by the test users	9.9
maximal number of interaction steps	
when always choosing the least promising interaction	10
when always choosing the most promising interaction	6
chosen interactions	
list of recommended service aspects	
upper part	67%
lower part	33%
list of recommended tradeoff opportunities	
upper part	63%
lower part	37%

Table 8.5.: Evaluation results related to Requirement U.11

If this were true, the users would more frequently choose recommendations in the upper half of the list of recommended interaction opportunities (i.e. those the system considers most promising). Otherwise, we would expect to have the selected interaction opportunities uniformly distributed over the two halves of the list. As it turned out, for the list of recommended service aspects, there had been 67% of the items selected from the upper part and 33% from the lower part, which are significantly more items from the upper part (chi-squared test, significance level 0.02). As for the recommended tradeoff opportunities, there had been 63% of the items selected from the upper part of the list and 37% items from the lower part. However, this result was not significant (chi-squared test, significance level 0.14). The lists with potential type refinements were too short to draw meaningful conclusions. As a result, we can conclude, that the test users in fact considered the system's recommendations.

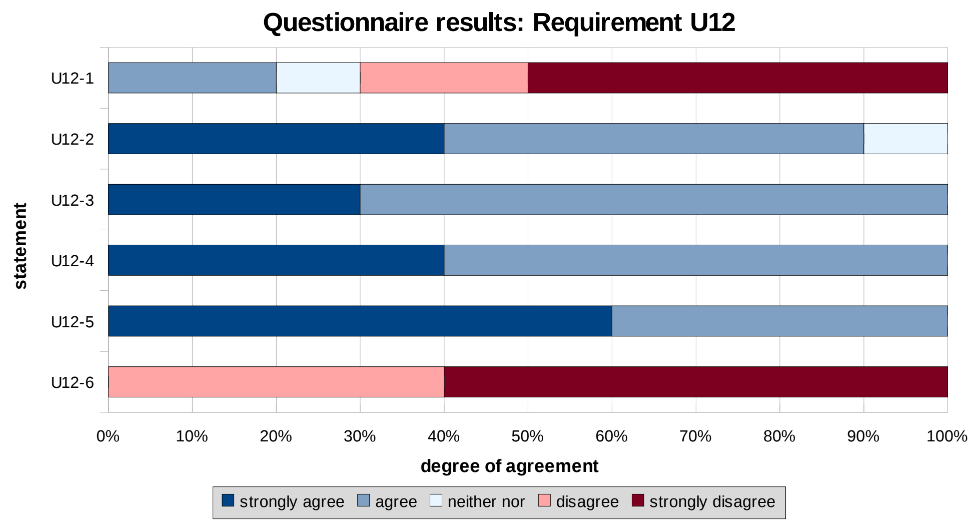
Finally, note, that the number of steps that were actually taken by the test users, 9.9 in average, was higher than the number of required steps as measured in the two automatically performed test runs. This is due to the fact, that though a suitable service might have been ranked high after a certain number of user interactions (as in the automatic test runs), the test participants had to verify whether the considered offer was in fact the best choice (e.g. by checking how well the offer suits to their requirements with respect to a yet unconsidered, but relevant service aspect). Moreover, in contrast to the automatically performed test runs, were the target offer was known right from the beginning, the test users had to successively discover their service requirements.

Summarizing, we can state, that the suggested system is in fact able to identify those interaction opportunities that are most promising in terms of their ability to reduce uncertainty about the service consumer's requirements and thereby focuses on parts of the model that are relevant in light of the available service options and in light of the user's requirements and preferences. Moreover, we have shown, that by taking these interaction opportunities, the number of interaction steps required to identify the offer that is most suitable can be significantly reduced. Finally, we demonstrated, that the test participants actually considered the system's recommendations and thus benefited from these effects. Thus, we can conclude that Requirement U.11 is fulfilled.

User autonomy and effective and coherent guidance Expressions of preferences and requirements as given by a potential service consumer that uses our system are user-initiated and not forced by the system. In particular, critiques applied to the properties of available service offers are self-initiated and might refer to any of the available offers. Moreover, the user can freely choose from the selection of service aspects and subtypes that is recommended to him for consideration. The presented list of opportunities is also complete, i.e. comprises of all applicable alternatives. This is in conformance with the test users' subjective perception (cf. Figure 8.5). The participants felt not forced by the system to state requirements and/or preferences they do not have (70% disagreed with statement U12-1, $p = 0.1$), and agreed, that they were free to express the preferences and requirements they have (90% agreed with statement U12-2, $p < 0.01$).

As indicated by the survey participants, the system not just allowed for user-initiated expressions of preferences and requirements, but also effectively guided the user in that process (cf. Figure 8.5). In particular, the users felt guided in the process of thinking of and expressing requirements and preferences (100% agreed with statement U12-3, $p < 0.01$), they felt supported in exploring and comparing available service alternatives and their characteristics in a systematic manner (100% agreed with statement U12-5, $p < 0.01$; 100% disagreed with statement U12-6, $p < 0.01$) and finally felt guided in the process of selecting a service that fits to their preferences and requirements (100% agreed with statement U12-4, $p < 0.01$).

The guidance provided by the system was also coherent, i.e. adjusted well to the user's intention and the context of the task at hand (cf. Definition 5.1). In particular, service attributes and subtypes recommended for consideration are restricted to those that directly refer to the service aspects that have been already considered by the user (those that can be directly added to the hierarchical requirements model) and thus are relevant in the context of the user's stated requirements. Offered tradeoff opportunities are displayed on demand and are relevant to the task at hand. This is due to the fact, that they directly refer to a given critique that has been specified by the user. These findings are confirmed by the test users. As can be seen in Figure 8.6, 90% of the survey participants agreed that the service aspects, type refinements and tradeoff alternatives provided by the system were relevant in the context of the task they intended to perform (90% agreed with statement U13-1, $p < 0.01$; 100% disagreed with statement U13-3, $p < 0.01$) and did not indicate a particular type of recommendations provided by the system that was not relevant to them (question



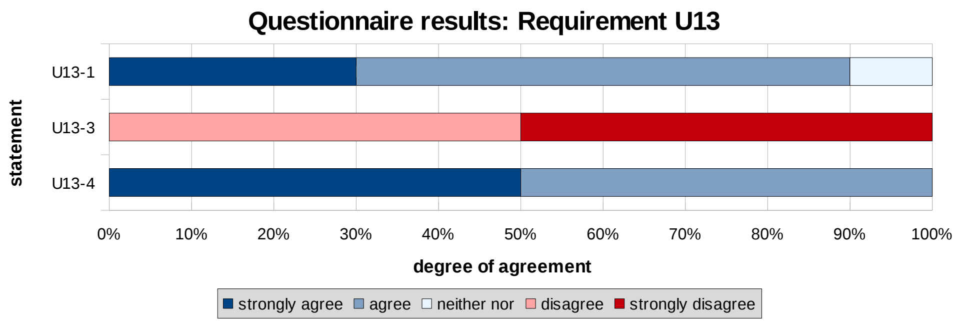
ID	Statement/Question
U12-1	The system forced me to state requirements and/or preferences I do not have.
U12-2	I felt free to express the preferences and requirements I have.
U12-3	The system guided me in the process of thinking of and expressing requirements and preferences.
U12-4	The system guided me in the process of selecting a service that fits to my preferences and requirements.
U12-5	The system helped me to explore and compare available service alternatives and their characteristics in a systematic manner.
U12-6	I got lost in the process of service selection.

Figure 8.5.: Questionnaire statements/questions and results related to Requirement U.12

U13-2). Finally, 100% of the test users agreed that the service alternatives presented by the system were relevant to them (100% agreed with statement U13-4, $p < 0.01$).

Summarizing the discussed evaluation results, we state that expressions of preferences and requirements as provided by a user of our system are user-initiated and are effectively and coherently guided by the system. Thus, the Requirements U.12 and U.13 are fulfilled.

Service selection efficiency According to Definition 1.2, the process of service selection is efficient, if a selection is made within an appropriate period of time and with adequate mental effort. Thereby, appropriateness refers to the importance of the service selection task at hand. However, both, the importance of a service selection task and the appropriateness of the effort and time spent to perform it, are subject to the personal perception of a user. Hence, the efficiency of the decision making process when supported by



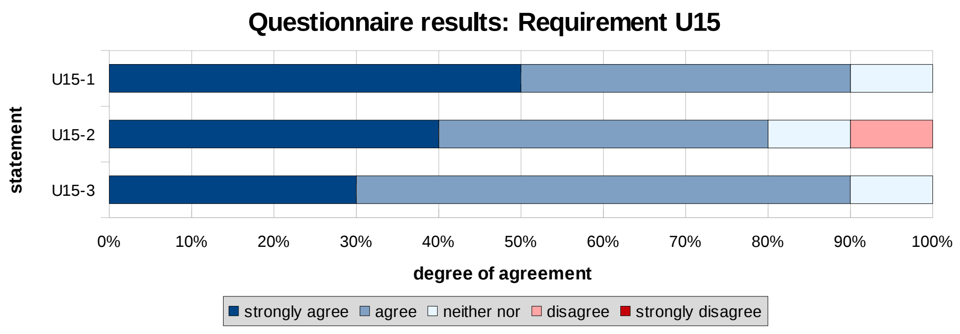
ID	Statement/Question
U13-1	The service aspects, type refinements and tradeoff alternatives provided by the system were relevant in the context of what I intended to do.
U13-2	If you do not agree, which recommendations were not relevant?
U13-3	The interaction opportunities (i.e. the selectable service aspects, type refinements and tradeoff alternatives) offered by the system rendered it impossible for me to do what I intended to do.
U13-4	The service alternatives presented by the system were relevant to me.

Figure 8.6.: Questionnaire statements/questions and results related to Requirement U.13

the proposed system could not be evaluated in terms of objective measures, the test users had to be rather asked for their subjective impression.

When asked about those issues, the respondents agreed that the amount of time that was required to identify the service that best fits to their preferences and requirements was acceptable (90$\%$ agreed with statement U15-1, $p < 0.01$). They even agreed, that by using the system, it took them considerably less time than it usually takes (80$\%$ agreed with statement U15-2, $p = 0.02$). Finally, the test users indicated, that the task of selecting an appropriate service alternative was not mentally demanding (90$\%$ agreed with statement U15-3, $p < 0.01$). Hence, the proposed system effectively supports service consumers in making service selection decisions efficiently and thus fulfills Requirement U.15.

Effective presentation of relevant information Information provided by the system are presented in a plain format to make them easy to perceive and easy to comprehend. Relevant and important information are emphasized, dispensable information are hidden. In particular, matching service alternatives are displayed in a simple table where the columns refer to the service aspects that are of interest to the user. Relevant service alternatives are placed on top. The top 10 service offers are additionally highlighted. To avoid information overflow and to ease the processing of relevant information, attribute-related information in the results table can be hidden, if not required. Service aspects and

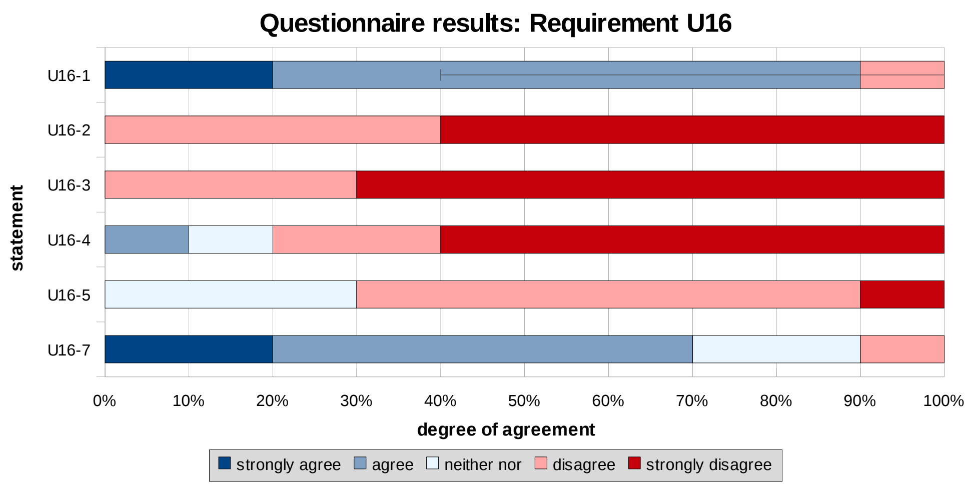


ID	Statement/Question
U15-1	The amount of time that was required to identify the service that best fits to my preferences and requirements was acceptable for me.
U15-2	By using the system, it took me considerably less time to identify a suitable service than it usually takes.
U15-3	It was easy to identify the service alternative that fits best to my preferences and requirements.

Figure 8.7.: Questionnaire statements/questions and results related to Requirement U.15

attribute subtypes recommended to the user for consideration are sorted by their usefulness and are restricted to those that have been specified in the available service offers and that are applicable in the context of the user's request model. Thereby, the recommendations that are considered to be most useful by the system are placed on top. Applicable critiques and resulting tradeoff opportunities are displayed on demand. The different item types of a tradeoff opportunity are color-coded to facilitate comprehension.

As indicated by the test users, system-provided information were in fact easy to comprehend and relevant information were easy to identify (cf. Figure 8.8). In particular, 90% of the survey participants said, that the presented information about available service alternatives and their characteristics were easy to understand (90% agreed with statement U16-1, $p < 0.01$). Both, the meaning of provided information as well as the purpose and way of using of offered interaction opportunities were clear and comprehensible for the test users. More specifically, the purpose of critiquing was apparent to all survey participants (100% disagreed with statement U16-2, $p < 0.01$). Though the meaning of the presented tradeoff opportunities was intelligible for the majority of the users, it remained incomprehensible to one of the participants (80% disagreed with statement U16-4, $p = 0.02$). All were able to make use of the recommended attribute subtypes (100% disagreed with statement U16-3, $p < 0.01$). To the vast majority of the test users, the amount of information presented by the system was acceptable (70% disagreed with statement U16-5, $p < 0.01$), but could be further reduced. As the users indicated, information such as the name of a service as well



ID	Statement/Question
U16-1	The presented information about available service alternatives and their characteristics were easy to understand.
U16-2	I did not understand what critiquing is for.
U16-3	I did not understand what the presented attribute types are for.
U16-4	I did not understand what the presented tradeoff alternatives mean.
U16-5	The amount of information presented by the system overwhelmed me.
U16-6	If you agree, which information should not be presented?
U16-7	It was easy to identify information that are important to me.

Figure 8.8.: Questionnaire statements/questions and results related to Requirement U.16

as units of measurement, if clear from the context, should not be presented at all (question U16-6). Finally, most of the test users agreed that it was easy to identify information that are important to them (70% agreed with statement U16-7, $p = 0.03$).

Summarizing, we can state that the system introduced in this thesis, was successful in presenting information in a format that makes them easy to perceive and easy to comprehend. Though the visibility of relevant and important information could be further improved, the majority of the test users were able to easily identify the information that are important to them. Hence, Requirement U.16 is fulfilled. Presentation issues related to the graphical requirements representation are discussed in Section 8.4.

8.4 Evaluation of the Graphical Model Representation

By using the graphical requirements representation introduced in Section 5.5, the user shall be involved in the model construction process by directly specifying service requirements and shall be enabled to interactively contribute to the quality of the system's requirements model by correcting the system-maintained model, if necessary (Requirement U.10). In this section, we evaluated whether the test users were in fact able to perform these tasks, i.e. were able to create a request model that expresses their service needs, whether the test users considered the proposed requirements representation as useful and how they judge its usability. In the remainder of this section, we will first introduce the methodology that has been applied for the evaluation (Section 8.4.1). After that, we will describe the evaluation setting (Section 8.4.2). Finally, we will detailedly discuss the evaluation results (Section 8.4.3).

8.4.1. Evaluation Methodology

To evaluate the usability of our graphical requirements representation, we performed a user study adopting the Thinking-aloud Protocol [Lew82], an approved technique for usability testing. During a series of tests, users were provided with our graphical requirements representation. Given two requirements specifications in natural language, they were then asked to create two request models based on these specifications by using the graphical requirements representation. The test participants were requested to comment on their thoughts while trying to perform this task. By observing this process, we gained valuable insights on how users perceived the interface of the graphical requirements representation and whether they were able to accomplish the given tasks effectively. In particular, we could examine, whether users approached the given tasks in the intended way and if not, where and why they had difficulties.

8.4.2. Test Setting

The main difficulty in the context of our evaluation, was the selection of a set of requirements specifications whose creation is manageable in a single test session and in an appropriate period of time and that is representative in the sense that it covers all modeling constructs that are required in a real world scenario. To meet those requirements, we constructed a set of two requirements specifications by extracting relevant pieces from the set of test requirements used by the Semantic Web Service Challenge[7], an approved initiative aiming at the evaluation of Semantic Web Service technologies. The basic idea of the challenge is to provide a set of scenario-based discovery, mediation and choreography problems that have to be solved by the participants. Each scenario includes a set of request descriptions in natural language and list of Web Services that match to it. Correct solutions

[7]http://sws-challenge.org

must retrieve the correct services for each request and rank them in the specified order. The solutions are verified and certified by the challenge staff. For the purpose of our evaluation, we considered the request specifications contained in the Shipment Discovery and Hardware Purchasing scenario of the challenge, since they refer to service domains that are comprehensible for end-users and cover a wide range of discovery problems. Though the set of requirements specifications in the scenario covered various aspects from a matchmaking perspective, it was redundant with respect to the modeling elements that were used. Hence, we extracted and (partially extended) relevant pieces from the scenario request descriptions and combined them into the following requirements specifications for our test, formulated in natural language:

Requirements specification 1
You'd like to ship a package ...

from: ... address ...
to: ... address ...
package dimensions: (l/w/h) 10/2/3 (inch)
package weight: 20 pounds
price: less than 120$
pickup: within two working days

Requirements specification 2
You want to own a computer, namely a ...

Apple Mac Book 13" with
at least Intel Duo Core Processor 2.0 GHz
at least 1100 MB RAM
at least 120 GB Hard Disc Size
color: white or black, but black is more desirable
for at most 1500 $

price and color are most important to you, price is two times more important than color

We omitted parts of the challenge's test requirements collection that referred to multi attribute conditions and to the specification of multiple effects at a time, since those are aspects that can be expressed in DSD, but are currently not supported by our request model implementation and the graphical requirements representation.

The user study was performed with 4 persons, which is the recommended number of participants for this kind of study [Nie93]. The procedure of a test session with a single user was as follows. At the beginning of the session, we gave a 5 minutes introduction to the main features of our graphical requirements representation using the specification of a request model for a printing service as an example. After that, the user was provided with the two verbal requirements descriptions to create. He was asked to use the graphical requirements representation to create two request models that appropriately model the service requirements specified in the verbal descriptions and to comment aloud on the actions taken to perform this task. During the test, we recorded how the user approached the task, where problems arose and how much time it took to perform the task. After the test session, we discussed the problems that were encountered during the test and also asked the test user to suggest improvements to the graphical requirements representation's interface. Finally, we asked the user to complete a questionnaire related to the usability of the graphical requirements representation (cf. Figure 8.9). Again, questions in the questionnaire were formulated as statements, where the users had to indicate their level of agreement using a scale from -2 (strongly disagree) to 2 (strongly agree). The presented evaluation results refer to the percentage of users that agreed/disagreed with a considered statement and are accompanied with the significance level p resulting from a chi-squared test. The complete questionnaire can be found in Appendix B.

The recommendations provided by the subtree and subtype recommender, that were provided during the test, were based on two request model specifications that had been created as a sample solution to the model creation tasks that had to be performed by the test users. To not overly simplifying the test task, the depth of the request model subtrees that could be created by using the subtree recommender was restricted to 1. All the test persons were neither familiar with the semantic service description language DSD, nor with the underlying service ontology or ontologies in general. Two of the study participants were computer scientists, two were not.

8.4.3. Results

All of the 4 test participants were able to create the two request models by using the graphical requirements representation. Though the users had some difficulties when starting to model the first requirements specification, they understood the modeling approach quickly and applied the basic features of the graphical requirements representation correctly, usually after having modeled the first one or two service aspects. Table 8.6 shows the results. The depicted table indicates how much time (in minutes) the test users required in average to model the two test requirements specifications. The two additional rows indicate how much time an expert user required for the same task with (*expert with rec.*) and without (*expert*) using the subtree recommender. As can be seen, the expert required 11.75 minutes to model requirements specification 1 and 4.5 minutes to model requirements specification 2. In our opinion, the amount of time required by the expert user is acceptable, in particular, when considering that a user's service requirements typically are not as complex as those specified in requirements specification 1. Moreover, employing the conversational approach to service selection proposed in this thesis, relieves the user from directly providing all his service requirements and encourages the user by presenting intermediate service results. However, the time required by the test users (21.25 ± 4.58 minutes for requirements

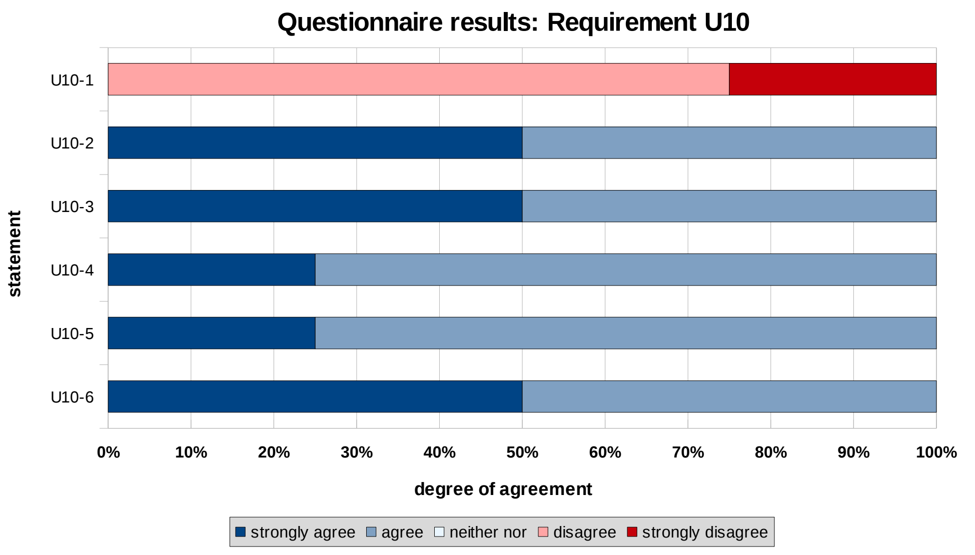
	req. spec. 1	**req. spec. 2**
expert	11.75	4.5
expert with rec.	8	3
test users (avg)	21.25 ± 4.58	11.38 ± 1.62

Table 8.6.: Time required to model requirements specification 1 and 2 (in minutes)

description 1 and 11.38 ± 1.62 minutes for requirements description 2), though acceptable in case of the second description, was too much. The comments provided by the users as well as the discussions following the tests revealed the reasons. As it turned out, the main problem was caused by the modeling of the underlying ontology and not by the interface of the graphical requirements representation. While the test users did not have problems with modeling their service requirements in terms of service effects, they were confused by the fact that a single service aspect could be modeled differently. For instance, the type of a processor can be modeled as a subtype of *Processor* or as an attribute. As a consequence the users often had to try which of both, subtyping or adding a subattribute, was the appropriate option. As a solution to that problem they proposed to gray out options that are not applicable. In doing so, it becomes obvious which option to choose. In addition to this ontology-related problem, the users remarked that they found it annoying to create separate subattributes for amounts and units, e.g. as they are required when specifying the price of a service or the weight of the package in requirements description 1. Instead, they would like to have the opportunity to input those information at once. A final remark was related to the representation of the attribute importance via the thickness of the connecting line. While the users appreciated to specify attribute importance this way, they found it difficult to compare different line widths to assess the relative importance of attributes. As a solution, they proposed to label the connecting line with a number indicating the importance.

The subtype recommender and in particular the subtree recommender were considered to be very helpful by the test users. While users did not make use of this option when they started to model requirements description 1, they made increasing use of this feature once they used it for the first time. However, we observed that the users often tried to model the required service aspects on their own and used the subtree recommender just in case they had no idea of how to model a certain aspect, which luckily happened not that often. With one exception, the users did not explicitly use the subtree recommender to accelerate the request model creation process, though as indicated by the results in Table 8.6, the amount of time that could have been saved by this is considerable. Finally, the possibility to collapse parts of the request model tree to save space was considered as positive.

These findings comply with the opinions expressed by the test users when completing the questionnaire (cf. Figure 8.9). In particular, the participants confirmed, that by using the graphical requirements representation, they were able to specify the service requirements they wanted to express (100% disagreed with statement U10-1, $p = 0.08$), that they were



ID	Statement/Question
U10-1	I was not able to specify the requirements and preferences I wanted to express.
U10-2	It was easy to specify requirements and preferences using the tool.
U10-3	The graphical requirements representation is intuitive and easy to understand.
U10-4	The graphical requirements representation allowed me to adjust my requirements and preferences as desired.
U10-5	The subtree recommender is helpful.
U10-6	The subtype recommender is helpful.
U10-7	What type of requirements/preferences, if any, were not easy to specify?

Figure 8.9.: Questionnaire statements/questions and results related to the graphical requirements representation

able to adjust their requirements and preferences as desired (100% agreed with statement U10-4, $p = 0.08$) and that this task was easy to perform (100% agreed with statement U10-2, $p = 0.08$)[8]. The study participants also indicated that the graphical requirements representation was intuitive and easy to understand (100% agreed with statement U10-3, $p = 0.08$). Both, the subtype recommender and the subtree recommender were considered to be helpful (100% agreed with statement U10-5, $p = 0.08$; 100% agreed with statement U10-6, $p = 0.08$). Note, that these results only supplement the observations made during the user study and should be understood as a general trend, rather than significant results, since they reflect the answers of 4 test persons.

We have demonstrated, that by using the graphical requirements representation the test users were able to directly specify and/or modify service requirements. Thus, they are

[8]Question U10-7 was not answered by any of the study participants.

involved in the model construction process and can interactively contribute to the quality of the system's requirements model by correcting the latter, if necessary. Hence, Requirement U.10 is fulfilled.

8.5 Summary and Conclusions

In the previous sections, we have shown that the requirements related to the requirements elicitation and service selection procedure that has been proposed in this thesis (Requirements U.1 to U.17) that have been compiled in Section 5.1 are fulfilled.

- In particular, we developed an incremental and interactive method for requirements elicitation and service selection (**fulfillment of Requirement U.5** as shown in Section 8.2).
- We also designed a requirements model, that is capable of semantically and richly describing a user's service requirements and preferences by means of the desired service effects (**fulfillment of Requirement U.1** as shown in Section 8.2), that is flexible enough to represent requirements related to different target domains (**fulfillment of Requirement U.2** as shown in Section 8.2) and that explicitly represents and locates uncertainty about the consumer's actual service requirements and his preferences by means of probability distributions (**fulfillment of Requirement U.3** as shown in Section 8.2).
- We also developed a mechanism that is capable of retrieving matching service offers and of providing personalized assistance based on the uncertain requirements encoded in this requirements model (**fulfillment of Requirement U.4** as shown in Section 8.2).
- By presenting these offers and their properties as well as by suggesting occurring service aspects and possible type refinements for consideration, the devised system makes the user aware of his hidden service requirements (**fulfillment of Requirement U.8** as shown in Section 8.3), stimulates thorough consideration (**fulfillment of Requirement U.6** as shown in Section 8.3) and subsequent expression of these in an intuitive and informal manner based on the matching service alternatives that are presented by the system (**fulfillment of Requirement U.7** as shown in Section 8.2).
- In particular, the system encourages decision makers to explicitly resolve conflicting requirements by thinking thoroughly about and to make tradeoffs (**fulfillment of Requirement U.6** as shown in Section 8.3). This is achieved by allowing the user to critique attributes of the displayed service alternatives and asking him to accept required tradeoffs on other attributes in turn or to abandon his critique.
- The suggested solution does not prompt or enforce a consumer to provide preferential statements he is not able or not willing to provide, i.e. expressions of preference and of requirements are user-initiated and not pre-determined by the system (**fulfillment of Requirement U.12** as shown in Section 8.3).

- As has been shown, this results in the expression of correct preferences and service requirements (**fulfillment of Requirement U.6** as shown in Section 8.3).
- It has been also demonstrated, that the process of requirements elicitation and service selection is effectively guided by the system (**fulfillment of Requirement U.12** as shown in Section 8.3).
- The provided guidance is coherent, i.e. adjusts well to the user's intention and the context of the task at hand rather than forcing the user to follow a certain sequence of requirements specification. It thus enables the user to understand and adjust to the service selection process and motivates and enables him to provide useful information (**fulfillment of Requirement U.13** as shown in Section 8.3).
- By presenting matching service offers, the proposed system also educates the user about relevant service alternatives and their characteristics (Requirement U.14).
- Finally, the suggested solution to service selection successfully encourages decision makers to consider the acquired knowledge about their service requirements as well as about relevant service alternatives and their characteristics when making a selection (**fulfillment of the Requirements U.14 and U.17** as shown in Section 8.3). It therefore effectively supports service consumers in making well-informed, balanced and consistent service selection decisions.
- Effective decision support is particularly enabled by two properties of the system. Firstly, its ability to emphasize relevant and important information and to present them in a format that makes them easy to perceive and easy to comprehend (**fulfillment of Requirement U.16** as shown in Section 8.3) and secondly, its ability to maintain consistency between the internal requirements model and the user's actual service requirements and preferences (**fulfillment of Requirement U.9** as could be only partially verified in Section 8.3). The latter is essential for the provision of meaningful and personalized support.
- The achievement of the second property is partly attributed to the fact, that the user is involved in the model construction process and thus is enabled to interactively contribute to the quality of the system's requirements model by correcting it if necessary using the graphical requirements representation that has been suggested in this thesis (**fulfillment of Requirement U.10** as shown in Section 8.4).
- As has been shown, the suggested solution not just supports service consumers in effectively making service selection decisions, but also enables them to make those decisions efficiently, i.e. within an appropriate period of time and with adequate mental effort (**fulfillment of Requirement U.15** as shown in Section 8.3).
- This is achieved by the provision of means to measure (**fulfillment of Requirement U.3** as shown in Section 8.2) and effectively reduce uncertainty about the consumer's service requirements and thereby focusing on parts of the model that are relevant in light of the available service options and in light of the user's requirements and preferences (**fulfillment of Requirement U.11** as shown in Section 8.3).

Hence, we conclude that **the operational goal of developing an incremental and interactive method for requirements elicitation and service selection that effectively supports service consumers in making well-informed, balanced and consistent service selection decisions and that enables them to do this efficiently (Objective 3) is fulfilled**.

9 Evaluation of the Feedback Mechanism

This chapter is dedicated to the evaluation of the feedback mechanism that has been proposed in Chapter 6 of this thesis. It will start with a recap of the evaluation objectives (Section 9.1), which manifest themselves in the list of requirements on the process of feedback elicitation, on feedback propagation, on the performance prediction procedure as well as on feedback presentation that has been compiled in Section 6.1. While the achievement of some of those objectives can be verified theoretically, for others a user study or a simulative evaluation is required. In the remaining sections of this chapter, we will present the results of those evaluations. In particular, we will theoretically verify that feedback elicited according to the proposed feedback mechanism is detailed, appropriate, comprehensive and meaningful. We will also show that only those feedback information that are required to allow for a good prediction quality are shared and that the quality of those information is adjustable to account for differing privacy needs. Finally, we will argue, that feedback-derived information are presented in a way that makes the user aware of the risk that is associated with the execution of a service and thereby accounts for different risk attitudes (Section 9.2). After that, we will present the results of a user study demonstrating the effectiveness of our feedback structure recommendation algorithm (Section 9.3). The evaluation of the performance prediction procedure has been done simulatively. The results are presented in Section 9.4. Finally, we conclude with a summary of the evaluation results in Section 9.5.

9.1 Evaluation Objectives

As argued in Section 1.4, a feedback mechanism is able to provide the knowledge that is required to make potential service consumers aware of the risk that is associated with the execution of a service and thus promises to contribute to the thesis goal of enabling well-informed and balanced service selection decisions in an environment where knowledge about service capabilities might be inaccurate and incomplete. In Section 6.1, we compiled a list of requirements that need to be satisfied by such a mechanism in order to actually meet those expectations by being both, effective in terms of acquiring the demanded information and effective in terms of its ability to support service selection. In the following, we will recall those requirements which set the standards for the evaluation of the feedback mechanism introduced in Chapter 6.

Requirements to Feedback Elicitation

Requirement F.1. (Feedback quality) *A feedback mechanism for Semantic Web Services should ensure that elicited feedback is detailed, comprehensive and appropriate in the context of a certain service interaction and that elicited feedback is meaningful, even if the services and requests (request models) that are involved in the service interactions are diverse and potentially refer to different application domains.*

Requirement F.2. (Adaptive elicitation) *The process of feedback elicitation should flexibly and automatically adjust to a consumer's willingness to provide feedback.*

Requirements to Feedback Propagation

Requirement F.3. (Quality of shared information) *Shared feedback should only comprise of necessary information being of a quality that is required to allow for a desired prediction quality. The quality of the shared information should be adjustable to account for differing privacy needs.*

Requirements to the Performance Prediction

Requirement F.4. (Effective exploitation) *Consumer feedback should be effectively exploited to predict a service provider's future performance, even if available feedback refers to service interactions that are diverse with respect to the services and requests (request models) that were involved.*

Requirement F.5. (Context dependency) *A feedback mechanism for Semantic Web Services should account for the context-dependent nature of service performance and service judgments by taking the request and service context, in which a judgment was made, into account when using feedback to predict a service's future performance.*

Requirement F.6. (Prediction confidence) *A feedback mechanism for Semantic Web Services should provide a confidence measure for its performance predictions.*

Requirements to the Presentation of Feedback Information

Requirement F.7. (Effective presentation) *Feedback information should be presented in a way that makes the user aware of the risk that is associated with the execution of a service.*

Requirement F.8. (Adjustable presentation) *Feedback information should be presented in a way that is adjustable to different risk attitudes.*

9.2 Partial Validation of Requirements Achievement

In this section, we will theoretically verify that the feedback mechanism which has been suggested as part of our thesis (Chapter 6), fulfills the Requirements F.1, F.3, F.7 and F.8. In particular, we will argue, that consumer feedback that is elicited according to the proposed mechanism is detailed, appropriate, comprehensive and meaningful (Requirement F.1). We will also show that only those feedback information that are required to allow for a good prediction quality are shared and that the quality of those information is adjustable to account for differing privacy needs (Requirement F.3). Finally, we will argue, that feedback-derived information are presented in a way that makes the user aware of the risk that is associated with the execution of a service and presented in a way that is adjustable to different risk attitudes (Requirements F.7 and F.8).

9.2.1. Elicitation of Appropriate, Comprehensive and Meaningful Multi-Attribute Feedback

As detailedly discussed in Section 6.4, consumer ratings might refer to any service aspect that is specified in the service request (model) that led to the judged service interaction and thus provide detailed information about the actual performance of a service. Since potential judgment targets are recruited from the request description (request model), it is ensured that they are appropriate in the context of a considered service interaction. This is true independent from the type of service request (model) which was posed and thus independent from the type of service interaction that has to be judged. Elicited consumer feedback is also comprehensive, i.e. judges all service aspects that, from the perspective of the judgment provider, are relevant in the context of a certain service interaction. This property is guaranteed, since our solution ensures that feedback provision is based on valid feedback structures (Definitions 6.2 and 6.3). As already argued in Section 6.4, the latter assures that the service consumer, either directly or indirectly (by providing an aggregated rating), judges all service aspects that are considered in the request (model). However, this argumentation is only valid, if the request (model) that led to a service judgment actually covers all service aspects that are important to the consumer (Assumption 6.1). Though this assumption cannot be guaranteed, we will show that at the time a service is invoked (and finally judged) our solution ensures that it is fulfilled at least to a degree which is sufficiently high to achieve a good prediction quality (Section 9.4).

To verify that consumer feedback, which is elicited according to the feedback mechanism that has been proposed in this thesis, is meaningful (Definition 6.1), we will argue

- that the provided judgments refer to the consumer's service requirements,
- that the concepts used to describe service aspects and thus the concepts used to describe judged aspects, valid constraints on them as well as valid relationships among them are well-defined and shared among the system participants,
- that the request and the service context underlying an expression of feedback are modeled and recorded and
- that it is evident whether and how feedback made in one context can be used to infer about a service provider's performance in another context.

As already detailed, consumer judgments refer to the consumer's service requirements. This is due to the fact, that potential judgment targets reference the service aspects that are specified in the service request (model), which encodes the service consumer's requirements with respect to the service interaction that is judged (Section 6.4). Moreover, by coupling judgment targets with the consumer's semantically described service requirements that have been specified in the service request (model), we provide judgments with a well-defined and commonly agreed upon meaning. This is due to the fact that service aspects covered in the request (model) refer to concepts in the service ontologies, which define not only valid service aspects and valid constraints on them, but also valid relationships among them and are shared among the system participants. A feedback item comprises all information that are required to effectively utilize the provided judgments. In particular, it contains the feedback structure underlying an expression of feedback, which encodes the paths of the judged service aspects and thus ensures the comparability of judgments referring to the same service aspect, but not necessarily to the same context. As described in Section 6.4, the request and the service (offer) context underlying a feedback item are recorded indirectly by means of the (partial) matching results for the posed request and the available offers (indirect context information, Definition 6.5). As we will demonstrate in Section 9.4, the collaborative-filtering-based techniques described in Section 6.6 can be successfully applied to this type of information in order to determine feedback relevance and by this means allows for the effective exploitation of feedback provided in one context to infer about a service's performance in another context. Our experimental results will also evidence that a good prediction quality can be achieved, even if the judgments that are utilized for the prediction refer to service interactions that are diverse and pertain to different, but related application domains (Section 9.4).

Summarizing this argumentation, we state, that consumer feedback which is elicited according to the feedback mechanism proposed in this thesis is detailed, appropriate, comprehensive and meaningful. Thus Requirement F.1 is fulfilled.

9.2.2. Appropriate Quality of Shared Information

Using consumer feedback to predict a service's future performance requires that consumers share the experiences they have made in past service interactions. This always implicates that the involved parties reveal sensitive information such as with whom they interacted and how they liked the outcome of these interactions. This is particularly true for context-aware approaches. As we have seen (Section 6.2), those typically produce more accurate predictions, but this comes at the cost of having to share and thus divulge additional information about the context of a judged (service) interaction. However, the amount and quality of the revealed information and the achieved prediction quality largely differ among existing approaches. As seen, context-aware trust and reputation systems (Section 6.2.3) as well as context-aware collaborative filtering systems (Section 6.2.1), typically share explicit information about the context of a service interaction. Context-aware approaches to experience-based service provider selection even require to share complete service request descriptions (Section 6.2.4). Our solution overcomes this problem by solely relying on the exchange of indirect context information given by the (partial) matching results for the posed request and the available offers. In doing so, distributing explicit context information and thus providing easily accessible personal information can be avoided. Compared to traditional, i.e. non-context-aware, systems more personal information are shared, but, as we will show, at the gain of a significantly improved prediction quality (Section 9.4). Summarizing, we can say that compared to existing context-aware approaches that are similarly powerful, less personal information are revealed when using the performance prediction procedure suggested in this thesis. It remains an open question, whether the amount and quality of the shared information can be further reduced while maintaining a similar prediction quality.

As already argued, there is a conflict of interest between the detailedness, i.e. the number and type, of the judged service aspects and the user's privacy concerns. The more detailed the provided judgments are, the higher is the quality that we can expect of the performance prediction (Section 9.4). However, providing more detailed judgments reveals more extensive (personal) information about the consumer's request (model) and his satisfaction with the delivered service. A strength of the proposed solution is that, by letting judgment providers freely choose the feedback structure to judge, it enables them to make this compromise according to their personal privacy restrictions. However, the effectiveness and flexibility of the approach strongly depends on the quality of the underlying ontology. An ontology that provides appropriate concepts for a fine-grained description of service requirements, will also allow for fine-grained tradeoffs.

As a result of our discussion, we state that the suggested approach requires the user to share only those feedback information that are needed to allow for a good prediction quality. Thereby, the quality of those information is adjustable to account for differing privacy needs. Thus Requirement F.3 is fulfilled.

9.2.3. Effective and Personalized Feedback Presentation

As detailedly discussed in Section 6.7.1, a service's predicted matching degrees with respect to its single attributes are color-coded and displayed in the results table in addition to the service properties. Presenting feedback-derived information this way makes the user aware of the risk that is associated with the execution of a service. This is due to the fact, that if an offer is ranked high, since its description having received a high matching degree with respect to the consumer's request (model), but at the same time the results of the feedback-based judgment prediction indicate a low matching degree, then the risk, that is associated with the execution of the service this offer refers to, is high. Hence, it is unlikely, that the service will provide the functionality that has been offered and likely that it will perform worse than expected. The user can easily identify those risky choices among the most promising service offers, since they are marked red. Thus requirement F.7 is fulfilled.

As argued in Section 6.7.2, the devised feedback presentation also accounts for different risk attitudes. In particular, it allows service consumers to specify their risk attitude precisely at the attribute level and enables personalized, feedback-aware service ranking based on those preferences. The specified risk preferences control to what degree the predicted judgment for a certain attribute and the matching value will influence the final rank of a service offer in the results table. Moreover, thresholds for the acceptability of predicted matching degrees and the required confidence level of predictions can be customized. Hence, we conclude that Requirement F.8 is fulfilled.

9.3 Evaluation of the Judgment Target Recommendation Mechanism

We already argued, that leveraging multi-aspect consumer feedback to predict a service's future performance has several benefits compared to the usage of single judgment feedback (cf. Section 6.1). On the other hand, providing such feedback can impose considerable time and effort on the user. To ensure feedback quality, the process of eliciting multi-aspect consumer feedback should therefore flexibly and automatically adjust to a judgment provider's willingness to accept this (Requirement F.2). In this section, we will demonstrate that the feedback elicitation mechanism with its judgment recommendation component that has been proposed in this thesis (Section 6.5) fulfills this requirement.

In the remainder of this section, we will first introduce the methodology that has been applied to evaluate our approach (Section 9.3.1) as well as the test data that have been used (Section 9.3.2). After that, we will describe the tests that have been performed and detailedly discuss their results (Section 9.3.3). The section concludes with a discussion of the evaluation results (Section 9.3.4) and a short summary (Section 9.3.5).

9.3.1. Evaluation Methodology

The goal of this evaluation is to analyze the quality of the recommendations produced by the algorithm, i.e. to investigate how well the suggested feedback structures fit to the users' judgment preferences. Since we cannot draw on a productive system with service consumers actually requesting for services and service providers offering service functionality, we could not run online tests, but had to perform our evaluations on a real-world or at least a realistic set of service judgment data. An advantage of this methodology is, that it allowed us to run the recommendation algorithm with various parametrizations whose resulting quality can be compared. However, the main problem of finding an adequate set of real world service judgments, that can be used as a basis for the evaluation remained. This data set should ideally comprise

- real world or at least realistic service judgment information[1] that are detailed, i.e. attribute-specific,
- information about the services that have been judged as well as
- information about the user's service requirements which gave rise to the service interaction that was judged.

The latter were required to analyze whether and how the judgment behavior of service consumers depends on the interaction context and how well the algorithm adjusts to this behavior.

- Moreover, the judgment data should follow the judgment scheme that is an integral part of our algorithm. In particular, this means that judgments should refer to service requirements and that service providers should be allowed to provide aggregated judgments for a set of judgments.
- Finally, the data set should be sufficiently large to allow for statistical evaluations.

Though real-world data sets of judgment data exist, they typically comprise of single overall service or product ratings. In contrast, data sets based on multi-criteria judgments are rare [AMK11]. Nonetheless, several online platforms such as Epinions.com, Ebay.com or Movies.yahoo.com elicit and use detailed service judgments and thus might serve as a source of such data. However, they lack necessary information about the judgment providers' service requirements, which are essential for our recommendation algorithm and thus are important for our evaluation. They also do not provide ratings that follow the judgment scheme that is required by our algorithm. They rather offer judgments that refer to a selection of non-related service aspects from which the user can choose. As a consequence, we could not draw on an existing data set for the purpose of our evaluation, but had to retrieve the required data from test users.

[1]For the purpose of this evaluation, information about what service aspects have been judged are sufficient. The actual ratings are not required, though they might also provide valuable insights into a service consumer's judgment behavior.

We identified two alternative ways of performing this task. One could either ask test users to create their own service requests (request models) and finally indicate which aspects to judge or could alternatively provide test users with predefined service requests (request models) and ask them to mark the aspects they would like to judge. An advantage of the first strategy is that the service requests (request models), on which the judgment decisions are based, are realistic, since they are user provided. However, on the other hand, eliciting judgment information this way would require users to have the ability to create service requests (request models). Even if the test persons were capable of performing this task[2], creating a number of service requests (request models) that is sufficiently large for a statistical evaluation would still be very time-consuming and annoying and thus might affect the quality of both, the provided requests (request models) and the judgment decisions that are made. Finally, the judgment information provided by the test users would refer to different service requests (request models) and thus would be incomparable, even if we would restrict potential requests (request models) to a certain service domain.

The second strategy, where users are provided with a set of predefined service requests (request models) and are asked to indicate which aspects to judge, does not exhibit this disadvantages. In particular, the effort that is burdened on the test persons is acceptable. Moreover, providing users with predefined requests (request models) allows us to influence and vary the characteristics of the test data and thus to compare the recommendation algorithm's performance under different conditions. However, this advantage is at a cost. Requests (request models) are no longer user-provided and thus might be artificial to some degree. Users might also have difficulties to identify themselves with the provided requests (request models) and thus might produce factitious responses. Nonetheless, we are convinced that the advantages of the second opportunity outweigh its disadvantages and hence pursued this strategy to elicit test data for our evaluation. Finally, we would like to remark that both of the suggested strategies require the test users to indicate the aspects they are willing to judge without being provided with an actual service response. The reason for this is that we do not have those information at our disposal. Nonetheless, they might affect a user's response. We will discuss this issue in Section 9.3.4.

9.3.2. Test Data

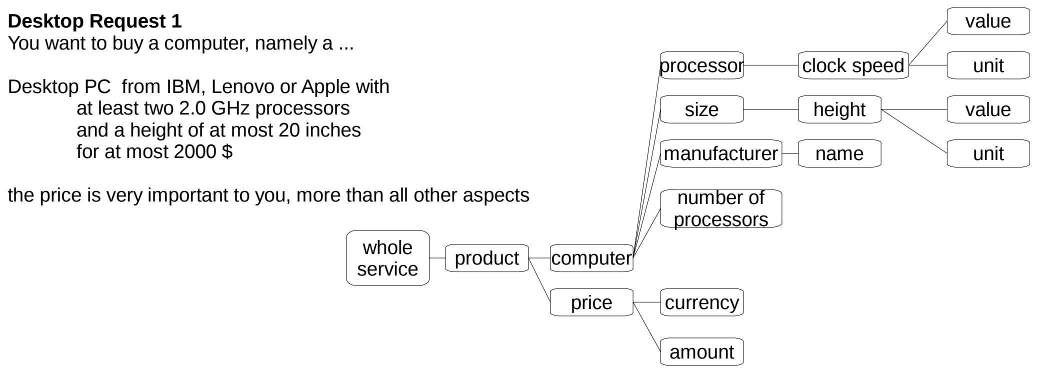
The test data were obtained using a number of 11 test users aged between 25 and 30 years (7 male and 4 female), of which 6 had a computer science background and 5 not. The elicitation procedure was as follows. Each test person was provided with a questionnaire comprising 12 service request models[3] covering typical requirements of consumers looking for computer items. 6 of the request models referred to services that offer desktop PCs and another 6 asked for services that offer digital watches. We chose these types of request

[2] As shown in Chapter 8, users are in fact able to create a request model that encodes their service requirements.

[3] The involved test data as well as the testing procedure are based on the usage of request models, but can be easily applied to DSD service requests. In the latter case, similar evaluation results can be expected.

models, since the test users were familiar with buying and rating products online as well as with the product domain itself. Moreover, the two request model types share common attributes, e.g. for both kinds of request models an attribute price could be specified, and thus allow to demonstrate the recommendation procedures ability to infer about a judgment providers rating preferences across service domains.

Figure 9.1 shows a sample request model from the questionnaire. The complete form can be found in Appendix C. As can be seen, request models of a single type varied with re-


Desktop Request 1
You want to buy a computer, namely a ...

Desktop PC from IBM, Lenovo or Apple with
at least two 2.0 GHz processors
and a height of at most 20 inches
for at most 2000 $

the price is very important to you, more than all other aspects

Figure 9.1.: Questionnaire - example request model

spect to the attributes that were specified, with respect to the attribute values that were acceptable and with respect to the requester's preferences related to these values and attributes. Each request model was given in terms of a textual description (Figure 9.1 left). The actual request model that was used for the tests, was derived from this description by creating a request model with the indicated properties via the graphical editor that has been introduced in Section 5.5. Thus, test users were not required to have knowledge about a certain service description language or the request model editor. In addition to each request model, the corresponding request model tree, comprising only the request model attributes that might be potentially judged, was provided (Figure 9.1 right).

Before being asked to complete the questionnaire, the test persons were given a short introduction on how to read and how to proceed with the form. In particular, the test users were instructed to imagine that they had posed the service request models provided at the left side of the form and shall highlight the service aspects they are willing to judge after service interaction in the request model tree on the right side of the form (blue-colored attributes in Figure 9.2)[4]. It was also pointed out that judging a parent attribute means providing an aggregated judgment for its subattributes and that just one attribute per tree path shall be judged to avoid redundant and potentially inconsistent judgments. Finally, it was indicated that the set of judged service aspects should either directly or indirectly (by providing an aggregated rating) cover all leaves of the request model tree.

[4]In doing so, users were not misled into choosing a feedback structure they do not want to judge, just because of being provided with a recommended structure.

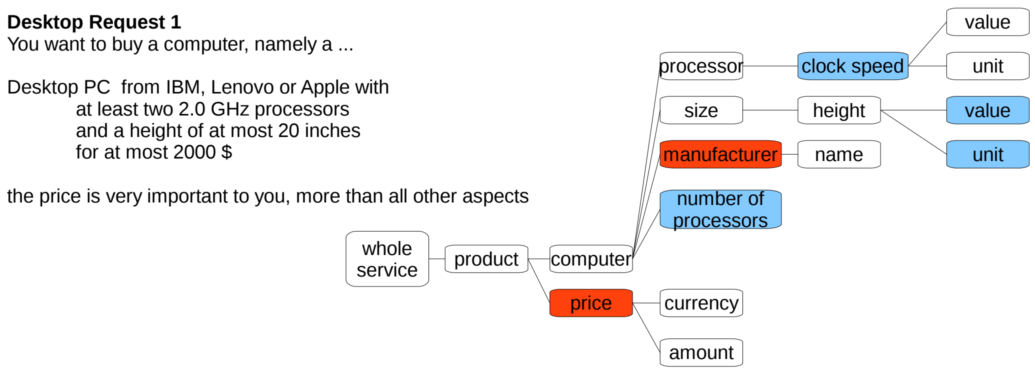

Desktop Request 1
You want to buy a computer, namely a ...
Desktop PC from IBM, Lenovo or Apple with
at least two 2.0 GHz processors
and a height of at most 20 inches
for at most 2000 $
the price is very important to you, more than all other aspects
whole service
product
computer
processor
clock speed
value
unit
size
height
value
unit
manufacturer
name
number of processors
price
currency
amount

Figure 9.2.: Questionnaire - example request model where the test person highlighted the service aspects that he is willing to judge (blue-colored aspects)

For each user and request model, it was recorded which and how many service aspects were judged. As a result of the data elicitation process, all questionnaires were answered correctly. Where the judged service aspects did not cover all leaf aspects of the tree (because in rare cases the test persons omitted ratings), we marked additional attributes as judged. More specifically, we additionally highlighted the minimal set of attributes that was required to fulfill the desired property, while at the same time keeping all attributes marked by the test person. As an example, consider the request model tree depicted in Figure 9.2. Assume that the test person indicated that he is willing to judge the attributes that are highlighted blue. To cover the attributes *manufacturer.name*, *price.currency* and *price.amount* that have not been covered by the user, we have to mark the attributes *manufacturer* and *price* (highlighted red in Figure 9.2) in addition to those attributes that are already highlighted by the user.

9.3.3. Tests and Results

In this section, we will report about the tests that have been performed using the elicited test data described in the previous section and will detailedly discuss their results. The first part of the present section provides a statistical analysis of the test data to empirically verify the validity of the assumptions that underlie the judgment recommendation algorithm to be evaluated (Assumptions 6.2 and 6.3). It is also concerned with the selection of appropriate values for the parameters of the algorithm. The second part of this section is dedicated to the evaluation of the recommendation quality, indicating how well the suggested feedback structures fit to the users' actual judgment preferences.

Validity of the assumptions As stated in Section 6.5, the proposed recommendation algorithm is based on the assumption that a user who judged a certain number of service aspects in a past judgment session is likely to be willing to judge a similar number of service aspects in a future judgment session, if the session refers to a similar request model (Assumption 6.2). Similarly, it is presumed that a user that judged a certain kind of service aspects in a past judgment session is likely to be willing to judge similar kinds of service aspects in a future judgment session, if the session refers to a similar request model (Assumption 6.3). The similarity of the number and kind of service aspects that have been judged in a past judgment session to those the user is willing to judge after a future service interaction is the higher, the more similar the involved request models are. In this section, we will empirically verify those assumptions using the elicited test data. In particular, to verify Assumption 6.2, we investigated whether there is a linear relationship (correlation) between the request model similarity $sim_{req}(r, r')$ of any two test data request models r and r' and the similarity of the number of service aspects that have been judged with respect to those models ($sim_{num}(fs,fs')$ of the judged feedback structures fs and fs'). To validate Assumption 6.3, we evaluated whether there is a linear relationship between the request model similarity $sim_{req}(r, r')$ of any two test data request models and the similarity of the kinds of service aspects that have been judged with respect to those models ($sim_{attr}(fs,fs')$ of the judged feedback structures fs and fs'). Existence and strength of these relationships were analyzed separately for each test user and with respect to different subsets of the test data. Correlation was measured in terms of Pearson's sample product-moment correlation coefficient

$$corr(\vec{x},\vec{y}) = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2\sum_{i=1}^{n}(y_i-\bar{y})^2}}, \tag{9.1}$$

where the pairs (x_i, y_i), $1 \leq i \leq n$, are the observed samples of the random variables X and Y whose relationship shall be investigated. This means, in the context of our evaluation $x_i = sim_{req}(r_i, r'_i)$ for a given pair of test data request models r_i and r'_i and $y_i = sim_{num}(fs_i,fs'_i)$ or $y_i = sim_{attr}(fs_i,fs'_i)$, respectively. The correlation coefficient $corr(\vec{x},\vec{y})$ takes values from the interval $[-1, 1]$. A value of $corr(\vec{x},\vec{y}) = 1$ signifies a perfect positive linear relationship between the involved variables, i.e. values from Y increase as the values from X increase, while a value of -1 denote a perfect negative linear relationship, i.e. values from Y decrease as the values from X increase. A value of $corr(\vec{x},\vec{y}) = 0$ indicates that there is no linear! relationship between the variables. However, this does not necessarily mean that there is no non-linear relationship between the variables. Values between -1 and 1 denote different degrees of correlation between the variables. The closer the value is to 0 the weaker is the correlation, the closer it is to -1 or 1, the stronger is the correlation. Finally note, that correlation between variables does not necessarily imply causality, since there might be other underlying causes for the correlation. However, it can point to a potential causal relationship.

The Figures 9.3 and 9.4 show the results of our correlation analysis. They depict the Pearson correlation between the request model similarity $sim_{req}(r, r')$ of any two test data request models r and r' and the similarity of the kinds of service aspects that have been judged with respect to those models ($sim_{attr}(fs,fs')$ of the judged feedback structures fs and fs', Figure 9.3) and the similarity of the number of service aspects that have been judged with respect to those models ($sim_{num}(fs,fs')$ of the judged feedback structures fs and fs', Figure 9.4). The correlation is given separately for each test user and for 3 different subsets of the test data, namely, the set of request models that refer to services offering desktop PCs (*desktop PC request models*), the set of request models that refer to services offering digital watches (*digital watch request models*) and the complete set of request models comprising both, the desktop PC and digital watch request models (*mixed*). In case of the latter, we only considered heterogeneous pairs of request models, i.e. those that referred to different kinds of services. In addition to the correlation coefficients, their 95% confidence interval is depicted. The presented results refer to the parameter configuration $\alpha = 0.05$, $\beta = 0.75$ and $\gamma = 0.2$ for the request model similarity measure sim_{req} and a value of $a = 1.1$ for the parameter of the similarity measure sim_{num}. As we will show in the remainder of the section, this parameter configuration is close to be optimal with respect to the strength of the correlation relationships. Nonetheless, the presented results are representative, since though for alternative parameter configurations the resulting correlation coefficients differ, they exhibit the same trend (c.f. Figure 9.6).

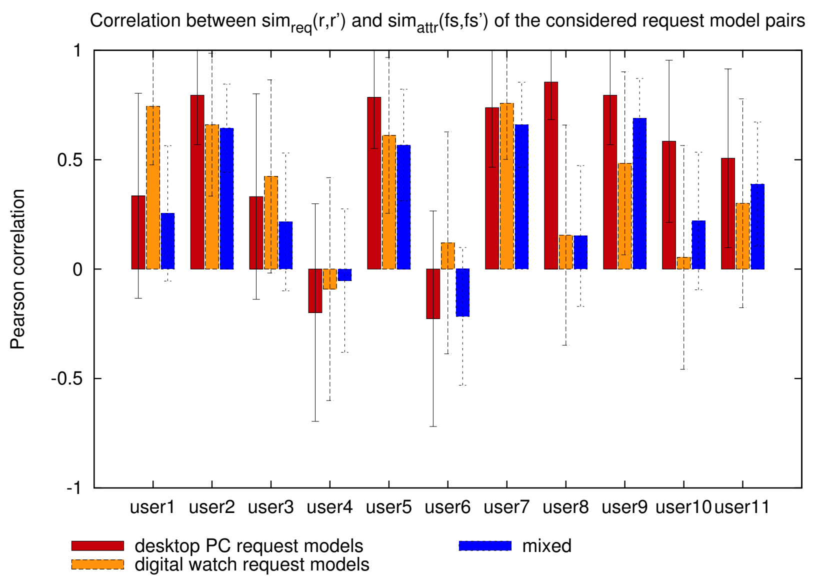


Figure 9.3.: Correlation between $sim_{req}(r, r')$ and $sim_{attr}(fs,fs')$ of the considered request model pairs

Figure 9.3 indicates, that the semantic similarity of request models and the similarity of the kinds of service aspects that have been judged with respect to them correlate. This means, that the service aspects that have been judged by a user were the more similar, the more similar the request models they were based on have been. We also observe that the degree of correlation is typically higher for highly similar request models, i.e. request models of the same type (red and orange bars in Figure 9.3), and lower for request models of different types (blue bars in Figure 9.3). These results empirically evidence the validity

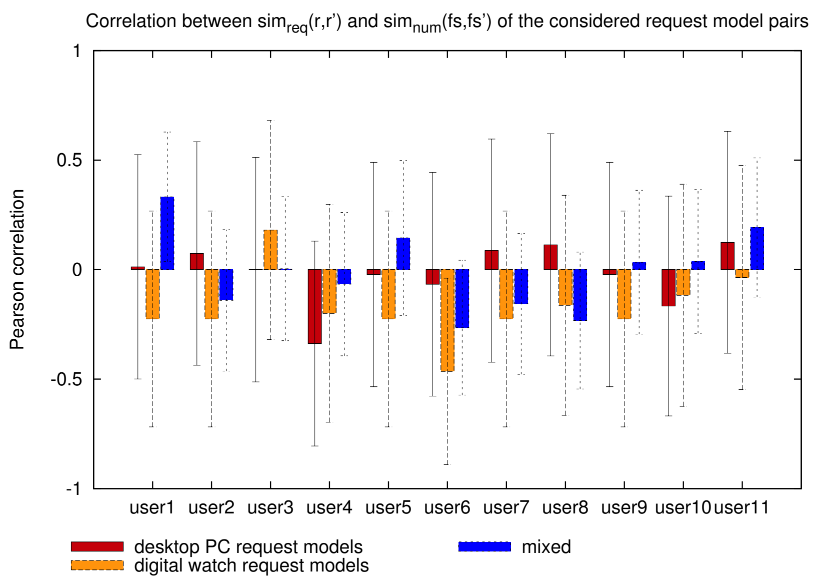


Figure 9.4.: Correlation between $sim_{req}(r, r')$ and $sim_{num}(fs, fs')$ of the considered request model pairs

of Assumption 6.3[5]. However, as the results for *user 4* and *user 6* indicate, the assumption is not valid for every user. Moreover, the strength of correlation is user-dependent.

In contrast, there is typically no correlation between the number of service aspects that have been judged with respect to two request models and their semantic similarity (Figure 9.4). The reason for this is revealed by a look at Figure 9.5. It shows the mean number of service aspects that have been judged with respect to request models of type *desktop PC* (red bars) and the mean number of service aspects that have been judged with respect to request models of type *digital watch* (orange bars) accompanied by their standard deviation. As can be seen, for a given user, the number of judged service aspects does not vary much among the request models of a given type. It typically differs by at most 1. However, there are usually larger differences in the number of judged aspects if we consider request models of different types. This means, that there is indeed a relationship between the semantic similarity of two request models and the number of service aspects that have been judged with respect to them, but at a more coarse-grained level than in case of the similarity referring to the kinds of aspects that have been judged. Hence, the results also support Assumption 6.2. However, as well as for Assumption 6.3, the results indicate, that the assumption is not valid for every user, i.e. every user is different! For example, the number of aspects that have been judged with respect to request models of type *desktop*

[5]In fact, correlation shows more than we require. Since in addition to the fact that as the similarity between two request models increases, the similarity between the kinds of service aspects that have been judged with respect to them increases, it also implies that if the similarity between two request models decreases, the similarity between the kinds of service aspects that have been judged with respect to them decreases. The latter is not required.

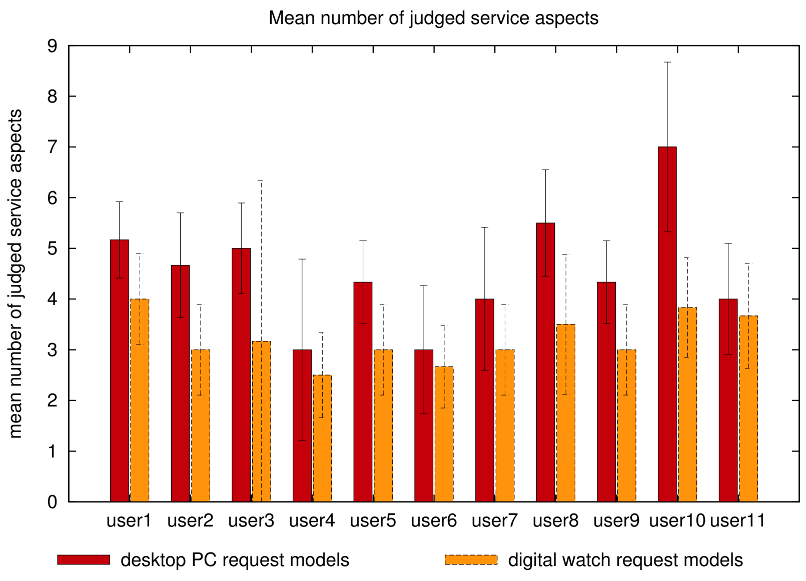


Figure 9.5.: Mean number of judged service aspects

PC and the number of aspects that have been judged with respect to request models of type *digital watch* do not differ largely for test users 4 and 6. Also interestingly, other than for the remaining test users, the number of aspects judged by *user 3* differed largely among the request models of type digital watch (Figure 9.5). However, there is a (weak) correlation between the similarity of the request models and the similarity of the number of aspects that have been judged with respect to them (cf. Figures 9.4 and 9.5).

Determining appropriate parameter values The similarity measures introduced in Section 6.5 depend on several parameters, which might influence the accuracy of the feedback structure recommendation algorithm proposed in Section 6.5. These are the parameters α, β and $\gamma = 1 - (\alpha + \beta)$ of the request model similarity measure sim_{req}, that determine the influence of the type similarity, the subtree similarity and the direct condition similarity on the request model similarity (cf. Section 6.5.3), as well as the parameter a of the measure sim_{num} indicating the similarity of the number service aspects that have been judged (cf. Section 6.5.2). Since the algorithm's accuracy can be expected to increase as the strength of the correlation between the semantic similarity of the request models and the number and kind of service aspects that have been judged based on them increases, we chose optimal parameter values for α, β, γ and a in a way that maximizes the correlation between the semantic similarity $sim_{req}(r, r')$ of any two test data request models r and r' and the similarity of the kinds of service aspects that have been judged with respect to them ($sim_{attr}(fs,fs')$ of the judged feedback structures) as well as the similarity of the number of service aspects that have been judged with respect to them ($sim_{num}(fs,fs')$ of the judged feedback structures). For that purpose, we determined the

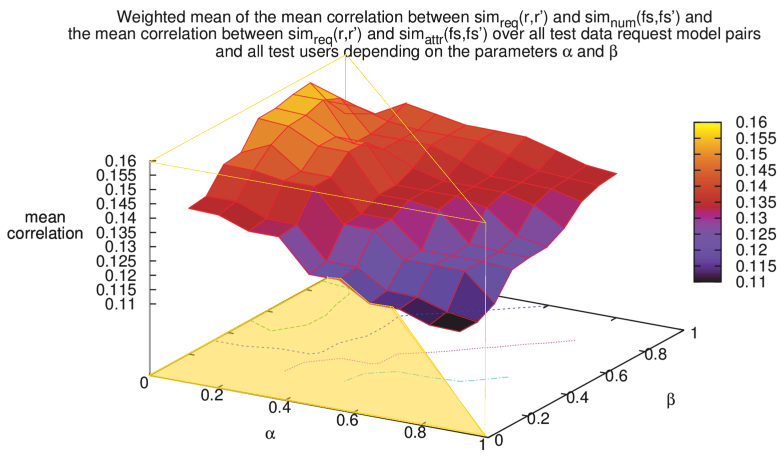


Figure 9.6.: Weighted mean of the mean correlation between $sim_{req}(r,r')$ and $sim_{num}(fs,fs')$ and the mean correlation between $sim_{req}(r,r')$ and $sim_{attr}(fs,fs')$ over all test data request model pairs and all test users depending on the parameters α and β

mean correlation between $sim_{req}(r,r')$ and $sim_{num}(fs,fs')$ as well as the mean correlation between $sim_{req}(r,r')$ and $sim_{attr}(fs,fs')$ over all pairs of test data request models and all test users depending on the values of the parameters α, β and γ. Since there was no parameter configuration that was optimal with respect to both correlation values, we had to select a parameter configuration that offers a good compromise between both issues. As a measure for the suitability of a given parameter configuration, we used the mean of the two correlation coefficients. Hence, the parameter configuration that achieved the highest mean correlation was considered most suitable, were both correlation values were equally important. The result is depicted in Figure 9.6, which presents the suitability of various configurations of the parameters α and β (only values within the yellow solid are valid). As can be seen, the highest suitability is achieved as β approaches 0.8 and α approaches 0. Hence, for the purpose of our evaluation, we chose parameter configuration $\alpha = 0.05$, $\beta = 0.75$ and $\gamma = 0.2$, which is close to be optimal[6]. As can be seen, the weight allocated to the type similarity is low compared to the other similarity values' weights. A reason for this might be that, though the test data request models refer to different types, those types are closely related and thus are not relevant for distinguishing different request models.

[6]We could have applied a multi-criteria optimization procedure to determine the optimal parameter configuration, however as Figure 9.6 indicates, there are only minor differences in the correlation coefficients of parameter configurations whose values differ by a small amount. Hence, it can be expected that the impact of choosing just a close to optimal parameter configuration on the recommendation accuracy is low in particular compared to the effort introduced by the selection and execution of a multi-criteria optimization procedure.

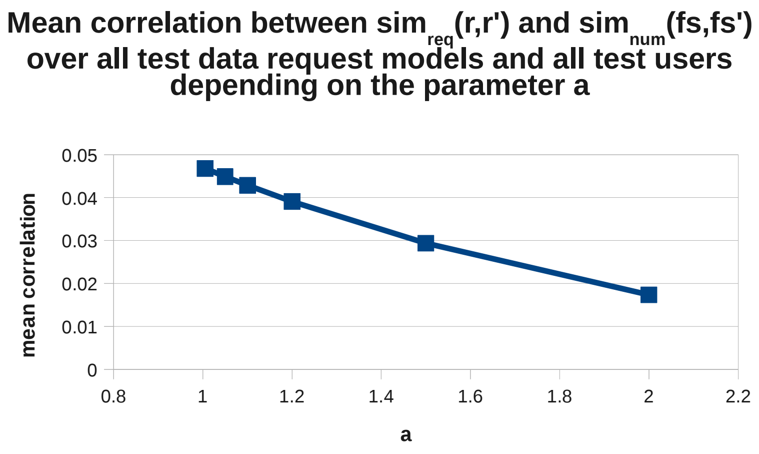


Figure 9.7.: Mean correlation between $sim_{req}(r, r')$ and $sim_{num}(fs,fs')$ over all test data request model pairs and all test users depending on the parameter a

Figure 9.7 depicts the mean correlation coefficient between $sim_{req}(r, r')$ and $sim_{num}(fs,fs')$ over all test data request model pairs and all test users depending on the parameter a of $sim_{num}(fs,fs')$. As can be seen, the mean correlation increases as a approaches 1. However, the increase in the strength of correlation is only marginally. For the purpose of evaluation, we therefore chose $a = 1.1$.

Recommendation accuracy To evaluate the accuracy of our rating structure recommendation algorithm, we ran a number of tests with different subsets of the test data elicited from each test user. The basic procedure for each test was as follows. Starting with no information about a user's previous judgment behavior, several judgment sessions were performed. During each session, one of the test data request models was selected. After that, the system proposed a feedback structure using the algorithm suggested in Section 6.5 provided with knowledge about the user's judgment behavior in the previous judgment sessions of the test. After being provided with the recommended feedback structure, the suggested structure was adjusted according to the structure that has been judged by the test user (as indicated in the questionnaire). This was done by expanding/collapsing feedback structure nodes. By clicking on a leaf node, all its direct children were expanded. By clicking on any other node, the complete subtree rooted at this node was collapsed.

The quality of the proposed feedback structure was measured as the edit distance between the recommended feedback structure and the feedback structure that was judged by the test user. More formally, we counted the number of expand/collapse operations the user would have to perform to create the structure whose leaves he finally judged. The rationale behind this measure is, that the edit distance is a direct measure of the user's effort to produce the desired structure and thus, in our opinion, is a good measure for the quality of the recommended structure.

For each test user, we performed 3 separate test runs using different subsets of the test data elicited from the user. Those were

- two tests *HomDesk* and *HomDigi*, each based on a set of request models that were homogeneous in the sense that the involved request models referred to services offering the same type of computer item, and
- a test *Het*, that was based on a set of heterogeneous request models which referred to services offering different types of computer items.

While test *HomDesk* referred to the set of desktop PC request models that have been judged by the test users, test *HomDigi* referred to the set of digital watch request models for which judgment data have been elicited. During the tests, each request model was considered twice and processed in the sequence given by the questionnaire (Appendix C). More specifically, the processing sequence of test *HomDesk* was *desktop PC request model 1, . . ., desktop PC request model 6, desktop PC request model 1, . . ., desktop PC request model 6* and the processing sequence of test *HomDigi* was *digital watch request model 1, . . ., digital watch request model 6, digital watch request model 1, . . ., digital watch request model 6*. Test *Het* considered both, the desktop PC and the digital watch request models judged by the test users, where request models referring to desktop PCs and request models referring to digital watches were processed alternately and in the sequence given by the questionnaire. The resulting processing sequence was *desktop PC request model 1, digital watch request model 1, . . ., desktop PC request model 6, digital watch request model 6*.

The Figures 9.8, 9.9 and 9.10 show the results of our tests. While the blue curve(s) of each figure indicate(s) the mean edit distance over all test users for each judgment session of a test, the red curve(s) refer(s) to the mean edit distance that would result, if the system would always suggest to provide a single overall judgment for the invoked service (default, if no experiences about past judgment sessions are available). The error bars indicate the 95% confidence interval for the mean edit distance.

The Figures 9.8 and 9.9 show the evaluation results for the tests *HomDesk* and *HomDigi*. The first half of the curves depicted in those figures refers to sessions were the request models are judged for the first time, while the second half shows the edit distance resulting from sessions that refer to request models which have been already judged, i.e. where the user's judgment behavior with respect to that model is known to the system. Consider the first part of the curves presented in Figure 9.8. As can be seen, the edit distances resulting from the recommended feedback structures are lower than that resulting from always suggesting the default structure and decrease as the knowledge acquired in previous judgment sessions increases. This indicates that the test users' judgment behavior in future judgment sessions was successfully inferred from their past judgment behavior that referred to similar request models. The mean number of editing operations required to be performed by the users to create the desired feedback structure was 0.4 ± 0.6 (after the 6th judgment session). This

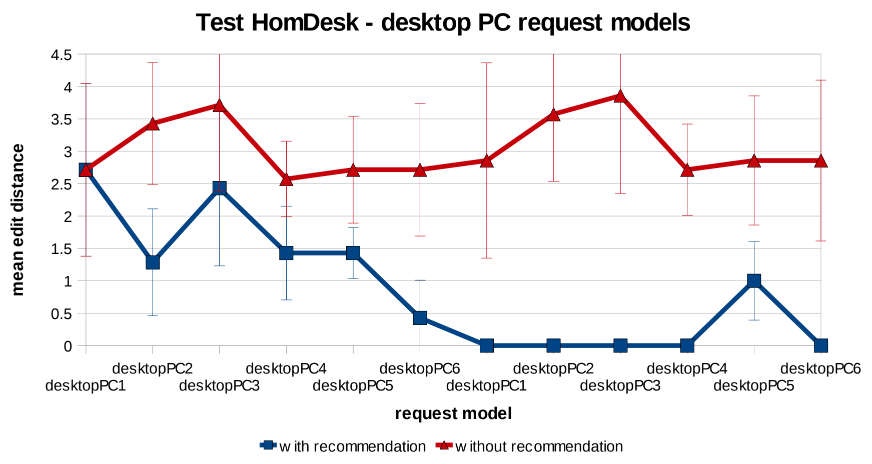


Figure 9.8.: Results of test *HomDesk*

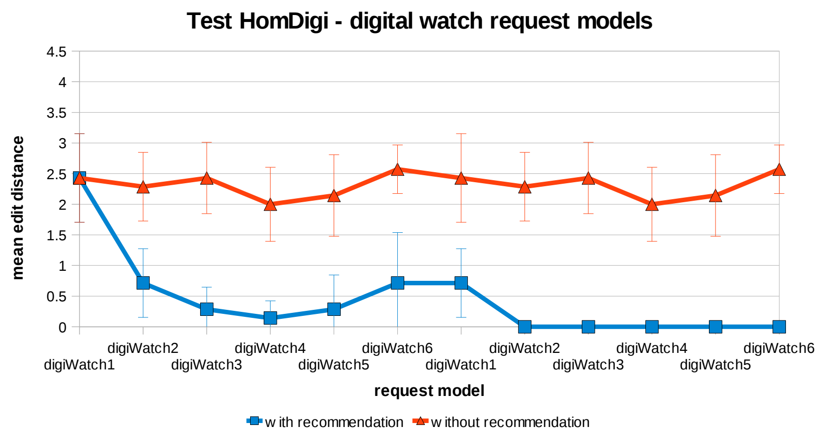


Figure 9.9.: Results of test *HomDigi*

is a reduction by 2.3 ± 1.3 operations. As expected, the edit distance further decreases to 0 during the second part of the test. This is due to the fact that knowledge about the users' judgment preferences with respect to the involved request models is already available. The results show that the proposed recommendation mechanism typically correctly references this knowledge, i.e. by comparing the similarities of the involved request models chooses the right judgment experience to generate a recommendation. This does not hold for *desktop PC request model 5*, which is not correctly referenced. Instead, a judgment experience that was based on a similar request model was leveraged for the recommendation. As a result, we observe an edit distance larger than 0 for this model.

The results for test *HomDigi* performed with the digital watch request models are depicted in Figure 9.9. The behavior that can be observed is similar to that during test *HomDesk*.

However, the determined edit distances are lower. The latter is due to the fact, that digital watch request models are less complex than desktop PC models, i.e. due to less attributes, the depth of the involved request model trees is lower and thus potential edit distances are smaller. The mean number of editing operations required to be performed by the users to create the desired feedback structure was 0.7 ± 0.8 (after the 6th judgment session). This is a reduction by 1.9 ± 0.9 operations. Again, the edit distance decreases to 0 during the second part of the test. However, as can be seen, *digital watch request model 1* is not correctly referenced.

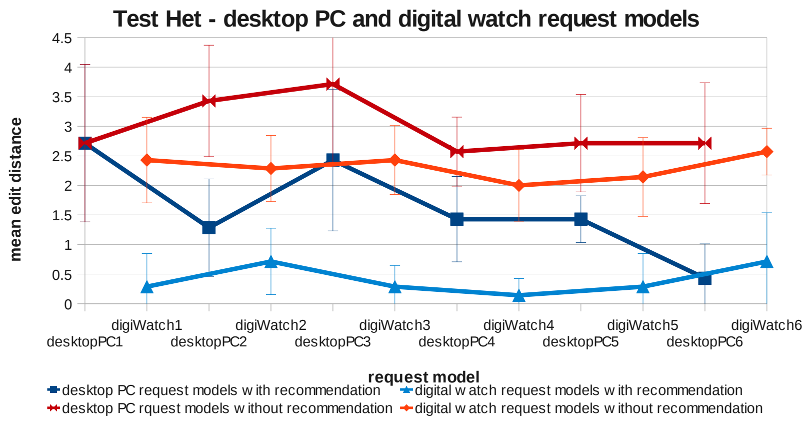


Figure 9.10.: Results of test *Het*

Figure 9.10 shows the evaluation results for test *Het* involving both types of request models. As can be seen, the curves depicted in this figure coincide with the first half of the corresponding curves shown in the Figures 9.8 and 9.9. Hence, the fact that judgment experiences referring to different kinds of request models are available to the recommendation mechanism, has no negative impact on its accuracy, i.e. judgment experiences referring to different types of request models are referenced correctly. Instead, we notice that the mean edit distance observed for *digital watch request model 1* is even lower than that observed during the homogeneous test. This implies that, in contrast to the first session of test *HomDigi*, where we cannot draw on previous judgment experiences, the knowledge acquired from the first desktop PC session was successfully exploited to improve the recommendation accuracy for *digital watch request model 1*.

9.3.4. Discussion

The results of our evaluation indicate, that a user's future judgment behavior can be inferred from knowledge about the kind and number of service aspects the user judged in past judgment sessions. However, beyond those purely behavioral findings, it would be worth investigating the psychological reasons that underlie this behavior. In particular, there might be other aspects that have an impact on the user's choice to judge certain service aspects, which are not covered by the two considered factors. For instance, judgment decisions might depend on the quality of the service that was finally provided, e.g. if the delivery time of a service was particularly good or bad the user might want to indicate that. At this time, we cannot investigate this issue, since due to the fact that knowledge about the (mis-)behavior of service providers is not available, test persons have to indicate their judgment decisions without having information about the actual outcome of the invoked service. In addition to that, a user's judgment behavior is likely to be inconsistent and arbitrary to some degree and thus might be unpredictable to a certain extent. It would be also interesting to analyze that issue.

We also found that judgment behavior is highly individual, i.e. user-specific. However, the evaluation results also suggest that there are classes of judgment providers who share behavioral patterns (stereotypes). For instance, while the future judgment behavior of the test users 4 and 6 did not depend on the kind of service aspects they judged in the past, the remaining test users' behavior was highly contingent on this factor. Identifying stereotypes and classifying users according to these would facilitate judgment recommendation and is likely to further improve its accuracy.

Finally, it would be interesting to investigate, whether the fact that a certain feedback structure was recommended to the user has an impact on his judgment decision.

It would be very promising to explore these and other issues more closely. Since this topic is strongly related to human behavior, follow-up work in this line of research should involve methods and results from psychology and human-computer interaction and should be conducted in collaboration with researchers from these fields.

Beside those additional research issues, extensions to the presented evaluation are required to ensure the validity and increase the reliability of its results. This should include the elicitation of test data related to a larger number and wider range of request models. Ideally, the utilized test data should be gathered from a productive system that applies the suggested recommendation mechanism. This would require minimal effort by the (test) users and would deliver realistic data. An evaluation performed on those data would therefore provide highly significant results.

9.3.5. Summary

The key findings of our evaluation are:

- As a result of a correlation analysis of the elicited test data, we found that a user who judged a certain kind of service aspects in a past judgment session is likely to be willing to judge similar kinds of service aspects in a future judgment session, if the sessions refer to similar request models. The similarity of the kind of service aspects that have been judged in the past judgment session to those the user is willing to judge after the future service interaction is the higher, the more similar the involved request models are. This even held for very similar request models, i.e. for models of the same type. As a consequence, Assumption 6.3 of the proposed feedback structure recommendation algorithm is fulfilled.

- We also found that a user who judged a certain number of service aspects in a past judgment session is likely to be willing to judge a similar number of service aspects in a future judgment session, if the sessions refer to similar request models. The similarity of the number of service aspects that have been judged in the past judgment session to the number of aspects the user is willing to judge after the future service interaction is the higher, the more similar the involved request models are. Therefore, Assumption 6.2 of the proposed feedback structure recommendation algorithm is also fulfilled. As we found, this assumption is only true for judgment sessions that are based on request models of different types. The number of judged service aspects with respect to request models of a single type did not vary much and thus is easy to predict.
- Though the test users' judgment behavior could be mainly explained by these two factors, for a few test users these factors were irrelevant or less relevant, i.e. the way users choose the service aspects to judge is highly individual.
- Our evaluation results indicate that the test users' judgment behavior in a future judgment session could be successfully inferred from their past judgment behavior that referred to similar request models, which resulted in a high recommendation accuracy. This even held, if the past experiences referred to request models of different types.
- For the desktop PC request models, the mean number of editing operations required to be performed by the test users to create the desired feedback structure was $0.4{\pm}0.6$ (after the 6th judgment session). This is a reduction by $2.3{\pm}1.3$ operations compared to a recommender with no knowledge about a user's previous judgment behavior.
- For the digital watch request models, the mean number of editing operations required to be performed by the test users to create the desired feedback structure was $0.7{\pm}0.8$ (after the 6th judgment session). This is a reduction by $1.9{\pm}0.9$ operations compared to a recommender with no knowledge about a user's previous judgment behavior.

Summarizing we state, that by recommending feedback structures the user is willing to judge, the suggested recommendation mechanism has proven to flexibly and automatically adjust to a consumer's willingness to provide feedback and thus enables effective feedback elicitation. Therefore, Requirement F.2 is fulfilled.

9.4 Evaluation of the Performance Prediction Mechanism

Effectively exploiting consumer feedback to predict the future performance of available services is crucial for enabling service consumers to make well-informed service selection decisions in a setting where service capabilities might be inaccurately and incompletely described. In this section, we will verify that the performance prediction procedure suggested in this thesis (Section 6.6) fulfills this requirement (Requirement F.4). We will show, that this is true, even if available feedback refers to service interactions which are diverse with respect to the services and requests (request models) that were involved. As we will demonstrate, this is enabled by both, leveraging detailed consumer feedback and taking the request and service (offer) context in which a judgment was made, into account when using feedback to predict a service's future performance (Requirements F.1 and F.5). However, the quality of the performance prediction strongly depends on the kind and accuracy of available consumer feedback. Based on our evaluation results, we will thus derive and present a confidence measure indicating the reliability of the predictions produced by the proposed algorithm (Requirement F.6).

The remainder of this section is structured as follows. After having commented on the methodology (Section 9.4.1) and the test data (Section 9.4.2) on which the evaluation is based and after having provided details on the test configurations that have been run (Section 9.4.3), we will detailedly discuss the evaluation results. In a first step (Section 9.4.4), we will have a look at the evaluation results that have been derived under the assumption, that service requests (in case of a request model, the generated request) accurately reflect a requester's actual service requirements (Assumption 6.1). After that, we will investigate how those results change, if requirements models are inaccurate (Section 9.4.5). Finally, we will derive and present a confidence measure taking our evaluation results into account (Section 9.4.6). The section concludes with a discussion (Section 9.4.7) and a short summary of the main evaluation results (Section 9.4.8).

9.4.1. Evaluation Methodology

Evaluating the accuracy of the proposed prediction procedure requires a data set on which it can be performed and tested with differing parametrizations. This data set should ideally comprise

- real world or at least realistic services with corresponding semantic offer descriptions,
- semantic service requests, each matching to a fraction of those services, and
- detailed, i.e. attribute-specific, consumer judgments that refer to interactions with the test data services and are based on the test data requests.

The latter serve as both, an input for the prediction procedure and an indicator for the accuracy of the predictions that it produces.

- The data set should be sufficiently large to allow for statistical evaluations.

- To demonstrate the advantages of our approach in a cross-domain scenario with complex service requirements, services and requests should be sufficiently complex, richly described and
- should refer to different, but related service domains.

In contrast to other research areas, where real world data sets for testing purposes are available (e.g. the Netflix Prize data set [7] in the area of collaborative filtering systems), we could not draw on such data. Though a few Semantic Web Service collections for evaluation purposes exist (e.g. OWLS-TC [8], the Semantic Web Service Challenge scenarios [9] or the Jena Geography Dataset [10]), they do not meet our requirements. In particular, they often comprise a relatively small set of rather simple services (e.g. converter or location services) which are also sometimes somewhat unrealistic and artificial. They typically do not contain service requests at all or service requests that match only to a small fraction of the services that are contained in the collection [Küs10]. Finally, we are not aware of a test collection that offers attribute-specific relevance judgments.

Hence, to evaluate our approach, we had to create a (semi-)artificial data set, that served as a base for our tests. To have at least a realistic set of services, we extracted structured information about computer items from a major online seller to generate semantic descriptions of services selling computer items. We chose this service category, since it comprises services that are sufficiently complex and refer to distinct, but related item domains. Moreover, real world data for other types of services in a number that is sufficient for statistical evaluations are hard to obtain. Since real user data, in particular real world service requests as well as information about the cheating behavior of service providers, are not available, we had to appropriately model those aspects to generate the required test data. Though a generated data set has the undesirable property that it is artificial to some degree, a generation-based approach also has the advantage that it allows to influence and vary the characteristics of the resulting test data and thus to evaluate their influence on the prediction quality.

9.4.2. Test Data

For the purpose of evaluation, we extracted information about $6,888$ computer items and their attributes from a major online seller. Each of these items belonged to one of the 8 categories *desktop PC, server, digital watch, e-book reader, PDA, organizer, notebook* and *electronic dictionary*. Table 9.1 provides an overview about the number of extracted items per category and the attributes that were considered.

[7] http://developer.netflix.com/docs
[8] http://projects.semwebcentral.org/projects/owls-tc
[9] http://sws-challenge.org/wiki/index.php/Scenarios
[10] http://fusion.cs.uni-jena.de/professur/jgd

Category	# items	Attributes
desktop PC	2,576	size (length, width, height), manufacturer, model, processor (clock speed, type), platform (producer, type), memory (size, type), #processors, floppy disc, graphics card, hard disc, #memory slots, system bus, secondary cache, display (size, resolution), modem, ethernet
digital watch	1,143	size (length, width, height), manufacturer, model
electronic dictionary	663	size (length, width, height), manufacturer, model
e-book reader	24	size (length, width, height), manufacturer, model, battery, platform (producer, type), memory (size, type)
PDA	809	size (length, width, height), manufacturer, model, processor (clock speed, type), platform (producer, type), memory (size, type), battery, display (size, resolution), modem
notebook	824	size (length, width, height), manufacturer, model, processor (clock speed, type), platform (producer, type), memory (size, type), battery, display (size, resolution), modem, floppy disc, graphics card, hard disc, #memory slots, #processors, secondary cache, system bus
organizer	157	size (length, width, height), manufacturer, model, battery, platform (producer, type), memory (size, type)
server	692	size (length, width, height), manufacturer, model, processor (clock speed, type), platform (producer, type), memory (size, type), #processors, floppy disc, graphics card, hard disc, #memory slots, system bus, secondary cache

Table 9.1.: Extracted computer items - their category and properties

From each item, we created a DSD service offer description (cf. Section 4.2) referring to a service (instance) offering this item. For that purpose, we created an ontology appropriately modeling the computer items domain(s) and complementing DSD's upper service ontology as well as the category ontologies. Figure 9.11 depicts the most important concepts and their attributes that are defined in this ontology. The entire ontology can be found in Appendix A. Note, that the 8 computer item categories give rise to 8 distinct, but related service domains, i.e. to 8 different, but related types of services (Figure 9.11). Service offers were randomly assigned to service providers, where the offers were uniformly chosen from the available offers in each category. We generated several kinds of service providers,

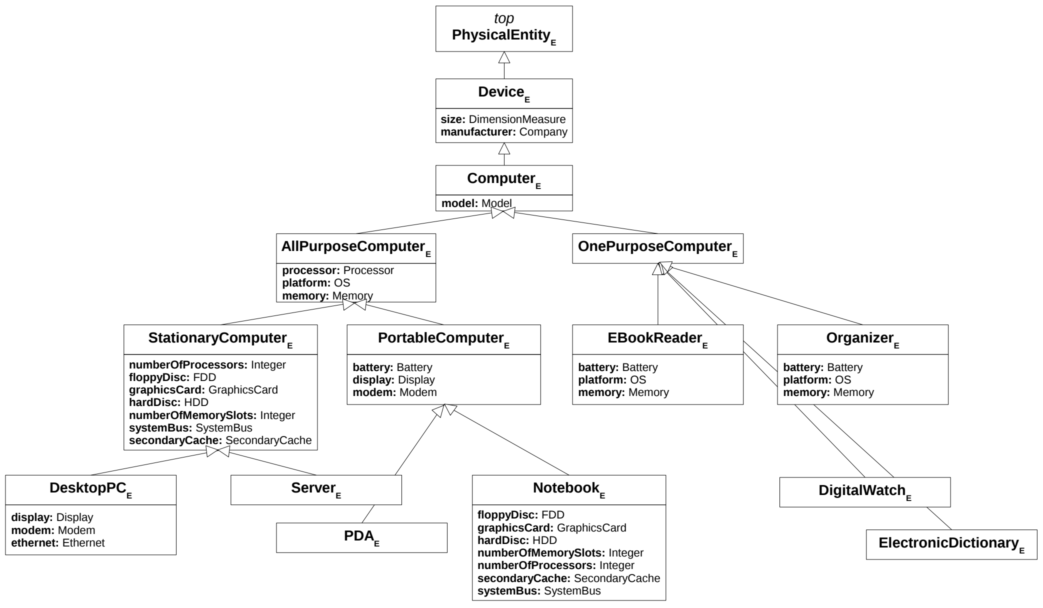


Figure 9.11.: Ontology for the computer items domain (excerpt)

where a provider was characterized by the type of services (service instances) he offered and the type of (mis-)behavior he exhibited. Regarding the type of the offered services, we distinguished between *homogeneous and heterogeneous service providers*. While homogeneous providers offered services of a single type (however with differing attribute values), such as only services selling desktop PCs or only services selling notebooks, heterogeneous providers offered services from different categories. For our test runs, we chose the following configurations:

- Homogeneous providers offered 40 services each selling a desktop PC.
- Heterogeneous providers offered 5 services per service type (= 40 services in total).

To simulate misbehavior, we generated modified versions of the service offers of a provider. This was done as follows. Each of a service offer's numerical attribute values was changed with a certain probability and by a certain amount of the original attribute value (that was indicated in the original offer description). The altering direction differed as it was appropriate for the type of service attribute. For instance, the value of the attribute *price* was increased, since higher prices are typically less desirable than lower prices, and the *hard disc size* was decreased, since a lower hard disc size is usually less preferable than a larger disc size. A nominal attribute's value changed with a certain probability, where the new attribute value was uniformly chosen from the values that occurred in the generated original offers. While the service offer descriptions that were created based on the original items represented a provider's promised services and were used to determine an offer's matching degree with a given request, the actual output of a service was determined by the modified offer description of the invoked service. More specifically, this means, we interpreted the matching value of the modified service offer as consumer rating. Figure 9.12 illustrates this issue.

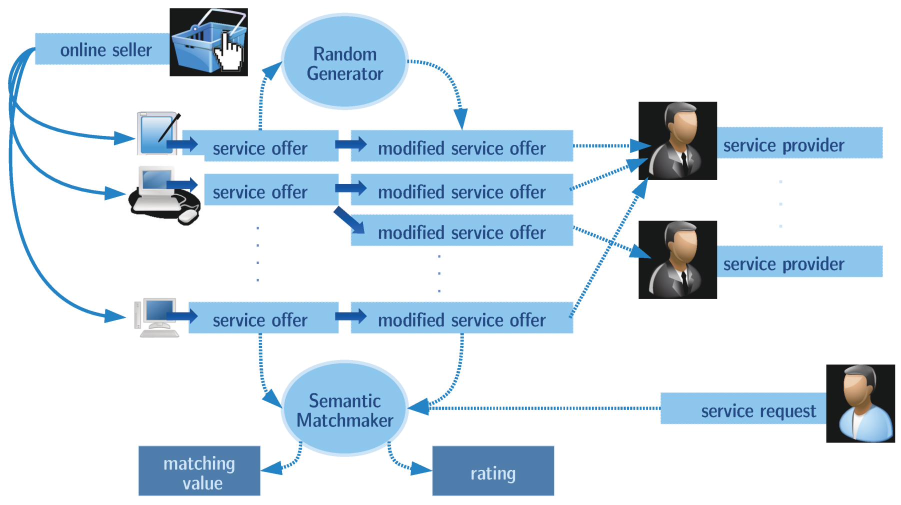


Figure 9.12.: Test data model

In the context of our evaluation, we distinguished two types of providers with different behavior configurations. Providers of type *heavy deceiver* provided services that largely differed from those promised in the (original) offer descriptions, while *weak deceivers* provided services that somewhat differed from those they promised. In particular,

- *heavy deceivers* modified a service attribute's value with probability 0.8 and by an amount between 0% and 70% of the original attribute value (the latter only holds for numerical attributes),
- *weak deceivers* altered a service attribute's value with probability 0.6 and by an amount between 0% and 30% of the original attribute value (the latter only holds for numerical attributes).

We also generated DSD service requests (cf. Section 4.2) covering typical requirements of consumers looking for computer items. All requests referred to services offering computer items of a specific type, e.g. a notebook, and belonged to one of six categories determined by the price and item-specific requirements of the requester. We distinguished the item-dependent price categories *low/medium/high prices acceptable* (price categories 0, 1 and 2) and created requests either specifying only non-functional attributes (*non-expert users*), like product color, or both functional, e.g. disk size, and non-functional attributes (*expert users*). A request attribute was specified with a certain probability (*probability attribute filled*) and its acceptable values (a range for numerical attributes) were chosen in a way which ensured that at least a certain minimal fraction (estimated) of offers matched to the request, i.e. had a matching degree > 0, with respect to this attribute (*min attribute* *matches*). By default, attributes that were specified in the request, but not in the offer were assumed to not match (cf. default missing strategy *assume_failed* Section 4.2). For each request attribute, we changed that behavior with a certain probability (*probability ignore missing attribute*), by adding the alternative missing strategy *assume_value[missing attribute matching value]* (cf. missing strategy *assume_value[n]* Section 4.2). This means, with a certain probability we indicated that a missing attribute shall match with the predefined matching degree *missing attribute matching value*. Our test runs were performed with the parameter configuration depicted in Table 9.2.

Parameter	**Value**
probability attribute filled	0.7
min attribute matches	40%
probability ignore missing attribute	0.2
missing attribute matching value	0.3

Table 9.2.: Request generation - basic parameter configuration

Consumer preferences, i.e. direct conditions, over single numerical attributes were modeled using an item-type-dependent decreasing/increasing (depending on the attribute type) linear preference function over the attribute's range of acceptable values. As an example, consider the preference functions for the attributes *price* and *hard disc size*, depicted in Figure 9.13.

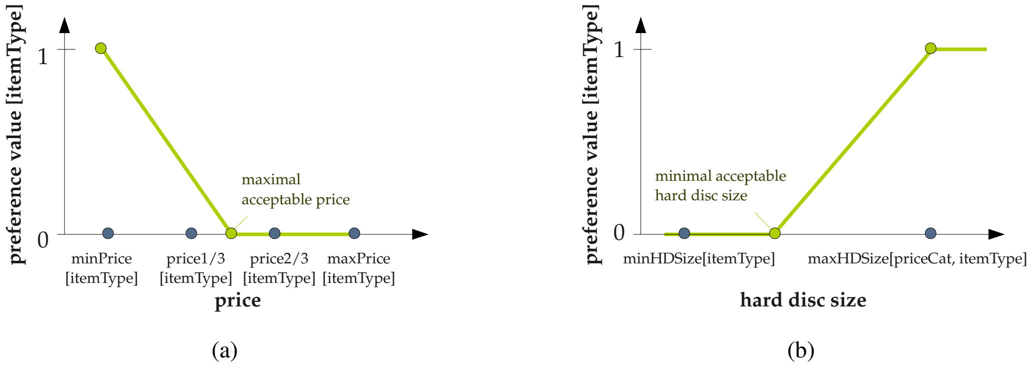


Figure 9.13.: Preference functions for the attributes *price* (a) and *hard disc size* (b)

In case of the *price* attribute, the preference value linearly decreased from 1 for the lowest possible price *minPrice*[*itemType*] (the lowest price value among those of the generated offers that referred to the requested computer item type) to 0 for the maximal acceptable price and remained 0 for higher prices (Figure 9.13(a)). The maximal acceptable price was uniformly chosen from a price range defined by the (item-type-dependent) price category of the request. The price ranges were defined as follows. *Price category 0* [*itemType*] contained prices in the range [*minPrice*[*itemType*], *price1/3*[*itemType*]), where *minPrice*$[itemType]$ was the lowest price value among those occurring in the generated offers that referred to the requested computer item type and $1/3$ of those service offers had a price lower than or equal to *price1/3*$[itemType]$. *Price category 1* $[itemType]$ comprised prices in the range $[$*price1/3*$[itemType]$, *price2/3*$[itemType])$, where $2/3$ of the created service offers that referred to the requested computer item type had a price lower than or equal to *price2/3*$[itemType]$. Finally, *price category 2* $[itemType]$ contained prices in the range $[$*price2/3*$[itemType]$, *maxPrice*$[itemType]]$, where *maxPrice*$[itemType]$ was the highest price value among those occurring in the generated offers that referred to the requested computer item type.

As a second example, consider the preference function for the attribute *hard disc size* (Figure 9.13(b)). The preference value linearly increased from 0 for a disc size value that was randomly chosen from the range $[minHDSize[itemType], maxHDSize[priceCat, itemType]]$ of hard disc size values defined by the disc size values of the generated service offers that referred to the requested computer item type and had a price that was lower than the upper bound of the desired price category *priceCat*, to 1 for the upper bound of this range. The lowest acceptable hard disc size for a given request, i.e. the lowest hard disc size with a non-zero preference value, was determined in a way which ensured that at least a certain percentage (*min attribute matches*) of the offers that referred to the requested computer item type and had a price lower than the upper bound of the desired price category had a larger hard disc size and thus a positive matching value with respect to the attribute *hard disc size* as specified in the request.

Preferences over nominal attributes were modeled by indicating acceptable attribute values with the *in*-operator for direct conditions (cf. Section 4.2). The set of acceptable values for a certain attribute comprised a fraction of those values for this attribute that were present in the generated offers which referred to the requested computer item type and whose price was lower than the upper bound of the desired price category. This fraction of attribute values was again chosen in a way which ensured that at least a certain percentage (*min attribute matches*) of the generated offers whose price was lower than the upper bound of the desired price category and that referred to the requested computer item type had a positive matching value with respect to the attribute. Preference values for acceptable attribute values were randomly assigned. While a certain percentage of those attribute values (depending on *min attribute matches*) received a matching degree of 1, the remaining attribute values received a matching degree uniformly chosen from the interval $(0, 1]$. The lower *min attribute matches*, the more of the acceptable attribute values were assigned a matching degree of 1[11].

Individual attribute matching degrees were aggregated according to the connecting strategies of type *weighted mean* (cf. Section 4.2), that were specified for each request attribute. Thereby, the importance of the price attribute was low for users which were willing to

[11] A percentage of $x = min(\textit{min attribute matches}, (1 - \textit{min attribute matches}))$ of the attribute values received a matching degree of 1, a percentage of $(\textit{min attribute matches} - x)$ of the values received a matching degree uniformly chosen from $(0, 1]$.

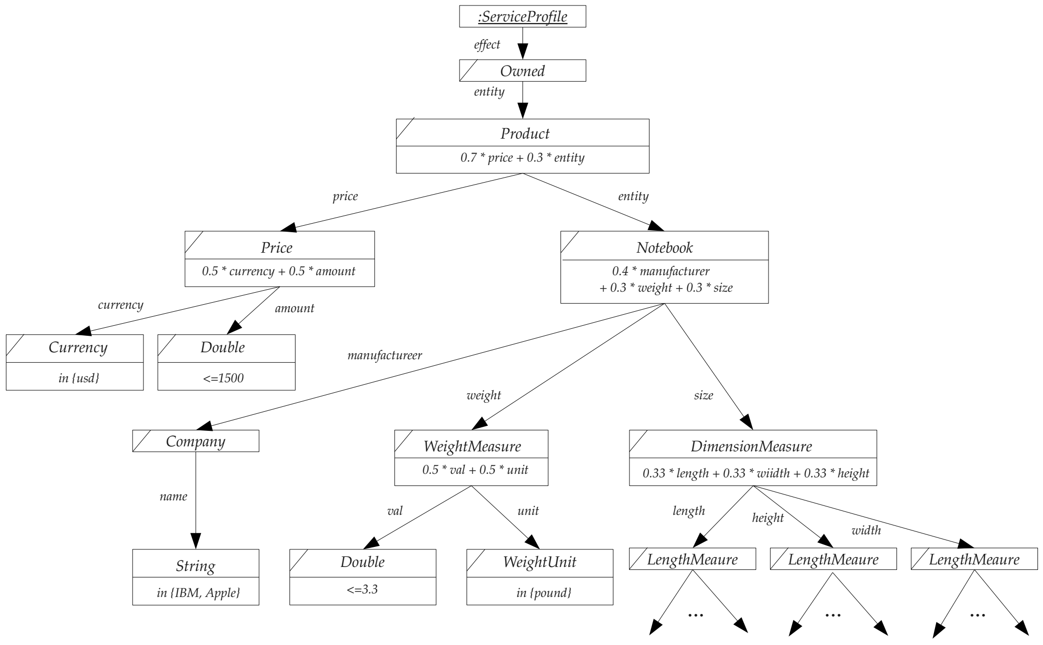


Figure 9.14.: Example of a generated DSD service request

accept high prices (requests of *price category 2*) and high for users that asked for a service that offers a cheap computer item (requests of *price category 0*). More specifically, the price weight for requests of *price category 0* was uniformly chosen from $[0.66, 1]$, the price weight for requests of *price category 1* was uniformly chosen from $[0.33, 0.66)$ and the price weight for requests of *price category 2* was uniformly chosen from $[0, 0.33)$. The weights for the other (specified) attributes were Gaussian distributed and normalized afterwards. Thus, we assigned medium weights with high probability and extremely high or low weights with low probability. Figure 9.14 shows a typical example of a generated DSD request, here for services that offer notebooks.

For our experiments, we generated two types of request sets, a *homogeneous* and a *heterogeneous request set*.

- The *homogeneous request set* contained 48 expert requests for services offering desktop PCs of *price category 2*[12].
- The *heterogeneous request set* contained an expert and a non-expert request per price category and considered service offer type ($= 2 \times 3 \times 8 = 48$ in total).

We would like to mention, that we are aware of the fact that this kind of test data is artificial to some degree. However, we argue that the service providers and requests contained in

[12]We chose expert requests, since they typically specify more attributes than non-expert requests and thus can be expected to be better suited to demonstrate the advantages of utilizing attribute-specific judgment information.

the generated test data set are much more diverse than those one would expect in reality and thus make it even harder rather than easier for our prediction algorithm to weight the single feedback contributions correctly and thus to make accurate predictions.

9.4.3. Test Settings

To evaluate the performance prediction procedure devised in this thesis as well as its variants, we ran a large number of tests with varying data sets and parameter configurations. The test data for each test comprised one of the request sets introduced above (either a *homogeneous* or *heterogeneous request set*) and a set of 5 providers, either all *homogeneous* or all *heterogeneous*, and, either all of type *weak deceiver* or all of type *heavy deceiver* ($= 2 \times 2 \times 2 = 8$ different data sets in total).

The basic procedure of a test was as follows. For each request and matching offer of the utilized test data, we determined the (partial) matching results (using the original service offer description) and noted the actual performance of the corresponding service, i.e. the ratings the requester would have given for each of the single service attributes after service execution, given by the (partial) matching results of the selected service offer's modified version. For 100 uniformly chosen pairs of request and matching service offer, we used the proposed prediction procedures to predict the corresponding service's performance, i.e. the ratings for the service with respect to the given request and the specified service attributes, thereby leveraging a (test-dependent) fraction of the generated test data ratings for the provider of the considered service. As a measure for the quality of the provided predictions, we utilized the most commonly used statistical accuracy metric in collaborative filtering, the Mean Absolute Error (MAE). More specifically, we compared the predicted rating for a request-service-pair with respect to a specific attribute with its actual rating, i.e. the matching value of the corresponding modified service instance and determined the absolute difference between those two values (*absolute prediction error*). The error was averaged over all 100 pairs of request and matching service offer (separately for each service attribute). The lower the mean absolute prediction error, the better the (mean) prediction quality. At this point, we want to note that the number of providers in the test data set had no influence on the prediction quality (just the number of offers per provider), since for a prediction only provider-specific feedback was considered. However, the number of service providers in a test data set had an impact on the number of test runs over which the prediction quality could be averaged. We would also like to note, that the described testing procedure is only valid under the assumption, that the service request (in case of a request model, the generated service request) that led to a service judgment accurately reflects the requester's actual service requirements (Assumption 6.1). In Section 9.4.5, we will introduce a modified version of the testing procedure in order to analyze how a relaxation of this assumption impacts the quality of the performance prediction.

We tested our prediction procedure with different parameter configurations. The following parameters (also cf. Sections 6.6.2 and 6.6.3) were varied:

- the *similarity measure* that was applied to determine the service and feedback provider similarity(s) of a feedback item with respect to a given request-offer-pair,
- the *maximum number of feedback items* that were used for a single rating's prediction,
- the *feedback aggregation function* that was used to aggregate the ratings that were contributed to a single rating's prediction,
- the *overall similarity measure*, i.e. the measure that was applied to calculate the overall similarity of a feedback item with respect to a given request-offer-pair, and
- the *relevance measure*, i.e. the measure that was used to determine the relevance of a single feedback contribution (its weight) for the prediction of a given request-offer-pair's judgment (with respect to a given attribute).

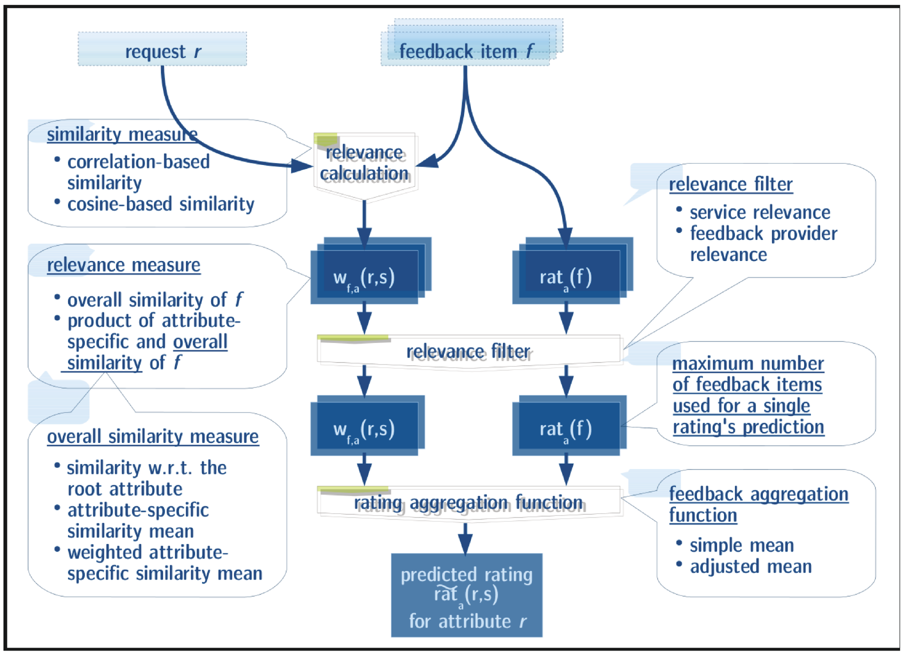

request r
feedback item f
similarity measure
• correlation-based similarity
• cosine-based similarity
relevance calculation
relevance filter
• service relevance
• feedback provider relevance
relevance measure
• overall similarity of f
• product of attribute-specific and overall similarity of f
$w_{f,a}(r,s)$
$rat_a(f)$
relevance filter
maximum number of feedback items used for a single rating's prediction
overall similarity measure
• similarity w.r.t. the root attribute
• attribute-specific similarity mean
• weighted attribute-specific similarity mean
$w_{f,a}(r,s)$
$rat_a(f)$
rating aggregation function
feedback aggregation function
• simple mean
• adjusted mean
predicted rating $\widetilde{rat}_a(r,s)$ for attribute r

Figure 9.15.: Parameters of the rating prediction procedure

Figure 9.15 illustrates the role of the different parameters in the prediction process. Table 9.3 provides an overview about possible values of those parameters and their abbreviations as used in the test results charts and their discussion. Please note, that the parameter configuration *AdjM-OvSim-RootSim* corresponds to the basic prediction procedure that is based on coarse-grained similarity information (Section 6.6.2), while all other parameter configurations correspond to variants of the prediction procedure that are based on fine-grained similarity information (Section 6.6.3).

In addition to the parameters outlined in Table 9.3, we applied *relevance filters* which ensured that only those judgment contributions, whose relevance with respect to the judged

Parameter	Value
similarity measure	• correlation-based similarity measure (Def. 6.19) • cosine-based similarity measure (Def. 6.20)
maximum number of feedback items	• any integer value (as far as the generated test data contained enough feedback)
feedback aggregation function	• simple mean (as a reference) - *SimpM* • adjusted weighted mean (Form. 6.21) - *AdjM*
overall similarity measure	• similarity w.r.t. the request's root attribute (Equ. 6.22) - *RootSim* • attribute-specific similarity mean (Equ. 6.29) - *AttrSimM* • weighted attribute-specific similarity mean (Equ. 6.30) - *WAttrSimM*
relevance measure	• overall similarity (Eqs. 6.22 and 6.28) - *OvSim* • product of attribute-specific and overall similarity (Equ. 6.26) - *SimProd*

Table 9.3.: Parameters of the prediction procedure, their abbreviations and possible values

service and the feedback provider exceeded certain thresholds, were used for the prediction (cf. Figure 9.15). In particular, a feedback contribution's *service relevance* was given as that factor of its relevance weight which referred to its service similarity. A feedback contribution's *feedback provider relevance* was given as that factor of its relevance weight which referred to its feedback provider similarity. That is, in cases where a feedback item's relevance with respect to a given request-service-pair and a given attribute was determined as its overall similarity (*OvSim*), its service relevance/feedback provider relevance was equal to the feedback item's overall service/feedback provider similarity (cf. Equations 6.22 and 6.28). In cases where a feedback item's relevance was calculated as the product of its attribute-specific similarity and its overall similarity (*SimProd*), service relevance/feedback provider relevance was determined as the product of the feedback item's overall service/ feedback provider similarity and its attribute-specific service/feedback provider similarity (cf. Equation 6.27). The generated test data as well as the test runs itself are reproducible, since the random number seed that was utilized by the random number generator which was used throughout the tests was provided as an additional test set generation and test parameter.

9.4.4. Results

In this section, we will present evaluation results that have been derived under the assumption, that a service request that leads to a service judgment accurately reflects the requester's actual service requirements (Assumption 6.1). We start with an analysis of our algorithm's performance when being used for the prediction of a given service's overall judgment with respect to a certain request. We will also investigate whether and to what extent the number of feedback items that are leveraged for this prediction affects the prediction quality. The final part of this section is dedicated to the evaluation of our algorithm's predictive performance when estimating attribute-specific ratings.

The Figures 9.16, 9.18, 9.20 and 9.22 illustrate the main results of our tests. Depicted is the *mean absolute prediction error* of the predicted overall rating, i.e. the error of the predicted rating for the request root attribute, accompanied with the 95% confidence interval as a measure for the reliability of the estimated prediction error and the significance of the results. While the left side of each chart shows the mean absolute prediction error when having used feedback contributions of arbitrary relevance for the prediction (*base prediction quality*), i.e. when having not applied any relevance filters, the right side depicts the lowest mean absolute prediction error that was achieved when having used only highly relevant feedback items for the prediction (*best prediction quality*), i.e. when having uniformly chosen the allowed number of feedback items from a fraction of the generated feedback items, whose service and feedback provider relevance exceeded certain thresholds. Remember, that the procedure which is used to determine service and feedback provider relevance differs for each parameter configuration of the prediction algorithm. Hence, the set of leveraged feedback items also differed among the parameter configurations. The thresholds that resulted in the lowest prediction error differed depending on the used parameter configuration and also depending on the used test data. This is due to the fact, that the distribution of the feedback items' relevance values is specific for each set of test data and each parameter configuration.

In our evaluation, we considered the following $4(\times 2)$ test data configurations:

- **configuration 1:** homogeneous requests and homogeneous providers (Figure 9.16),
- **configuration 2:** homogeneous requests and heterogeneous providers (Figure 9.18),
- **configuration 3:** heterogeneous requests and homogeneous providers (Figure 9.20) and
- **configuration 4:** heterogeneous requests and heterogeneous providers (Figure 9.22),

each with weak (upper part of each figure, i.e. figure parts (a) and (b)) or heavy deceiving providers (lower part of each figure, i.e. figure parts (c) and (d)). We ran tests with both, the correlation-based similarity measure (left side of each figure, i.e. figure parts (a) and (c)) and the cosine-based similarity measure (right side of each figure, i.e. figure parts (b) and (d)), each with several parameter configurations (Table 9.3). Parameter configurations are denoted by a code comprising of 3 dash-separated parts, the abbreviation for the value of the feedback aggregation function that was used, the abbreviation for the value of the relevance measure that was used and the abbreviation for the value of the overall similarity measure that was used (refer to Table 9.3 for an overview of possible parameter values and their abbreviations). As an example, consider the code *AdjM-OvSim-AttrSimM*. It refers to a test configuration that used adjusted weighted mean as the feedback aggregation function, a feedback item's overall similarity as its relevance measure and attribute-specific similarity mean as the overall similarity measure. Sometimes it was useful to refer to a class of parameter configurations that share certain parameter values and may differ with respect to others. We used the symbol * to indicate that members of the class may take any value of a considered parameter. For example, the code *AdjM-**-** refers to the class of all parameter configurations which have in common that they use adjusted mean as the feedback aggregation function. Tests with parameter configuration *SimpM* served as a baseline for our evaluation. In those cases, the predicted judgment was determined by simply averaging the single feedback item's rating contributions, i.e. no relevance information was considered. All depicted results are based on test runs performed with at most 5 feedback items used per prediction. This value was chosen, since, as we will show later on in this section, further increasing the number of feedback items that are leveraged for the prediction will not improve the best prediction quality that can be achieved when using the test data configuration that lead to the worst prediction quality (configuration 4).

The charts depicted in the Figures 9.17, 9.19, 9.21 and 9.23 complement those in the Figures 9.16, 9.18, 9.20 and 9.22. For each of the $4(\times 2)$ test data configurations, they depict two charts. Again, the upper part of each figure shows the results for the test data configuration with weak deceiving providers and the lower part of each figure presents the results for the test data configuration with heavy deceiving providers. The left charts (figure parts (a) and (c)) of each figure depict the mean absolute prediction error for fixed feedback provider relevance thresholds, depending on the minimal service relevance of the feedback items that have been used for the prediction. The right charts of each figure (figure parts (b) and (d)), show the mean absolute prediction error for fixed service relevance thresholds, depending on the minimal feedback provider relevance of the feedback items that have been used for the prediction. In each case, the depicted results refer to test runs that were based on the correlation-based similarity measure and were performed using adjusted weighted mean as the feedback aggregation function. The relevance weights were given by the overall similarity of each feedback item, determined as the mean of the single attribute similarities (parameter configuration *AdjM-OvSim-AttrSimM* in the Figures 9.16, 9.18, 9.20 and 9.22).

As can be seen in the Figures 9.16, 9.18, 9.20 and 9.22, the correlation-based measure almost always either outperformed the cosine-based similarity measure, i.e. produced lower prediction errors, or at least delivered equally good results. Hence, in the subsequent discussion, we will refer to the results produced by the correlation-based similarity measure.

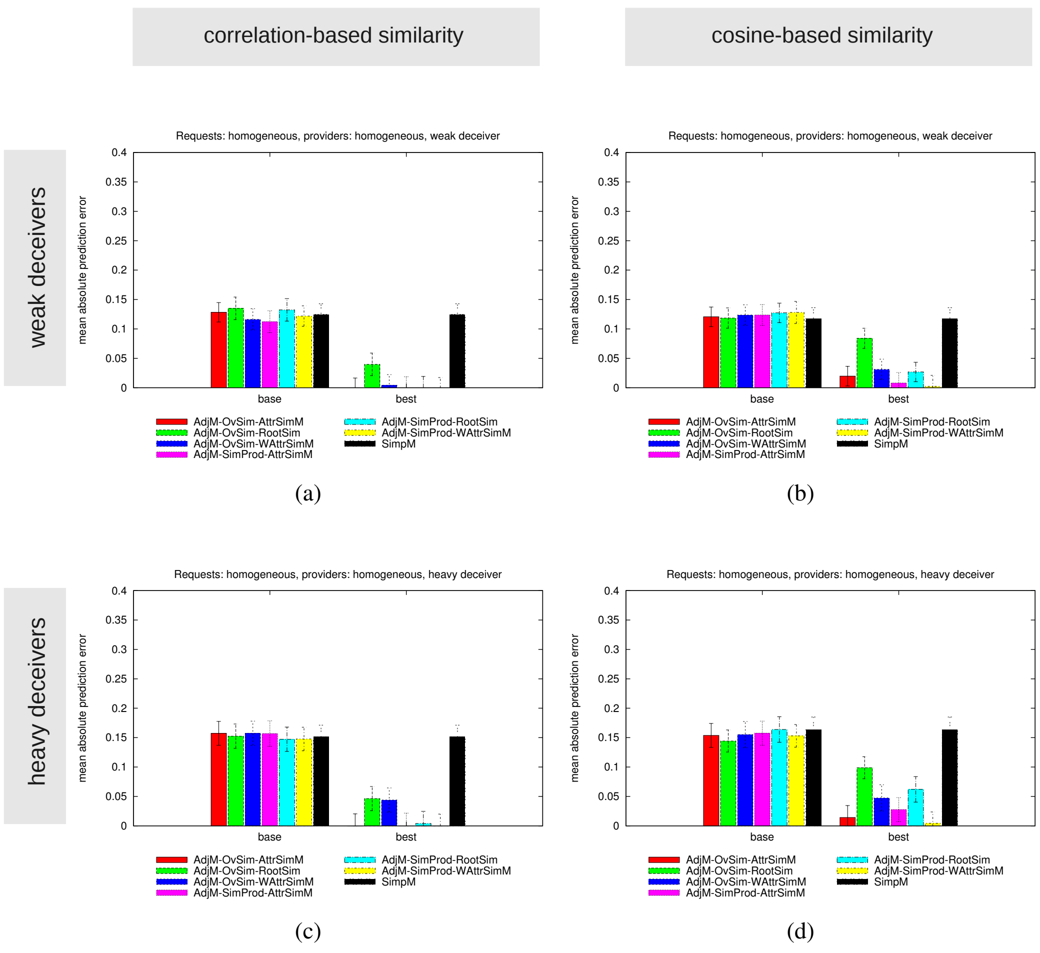


Figure 9.16.: Evaluation results for test data configuration 1 - homogeneous requests, homogeneous providers, weak deceivers (upper part)/heavy deceivers (lower part): mean absolute prediction error based on test runs performed with the correlation-based measure (left) and the cosine-based measure (right)

Configuration 1 - homogeneous requests, homogeneous providers Figure 9.16 depicts the main results for this test data setting. Figure 9.17 shows the corresponding relevance charts. As can be expected in a purely homogeneous setting, the base mean prediction error, i.e. when applying no relevance filters (left side of the charts in Figure 9.16), is relatively low (about 0.13 for the test data configuration with weak deceivers and 0.15 for the configuration with heavy deceivers). However, as can be seen, leveraging only those feedback items that have a high relevance with respect to the service and the request for which a rating shall be predicted, results in a much lower prediction error (right side of the charts in Figure 9.17). As Figure 9.17 illustrates, this is attributed to the fact, that though having homogeneous providers, both, service and feedback provider relevance,

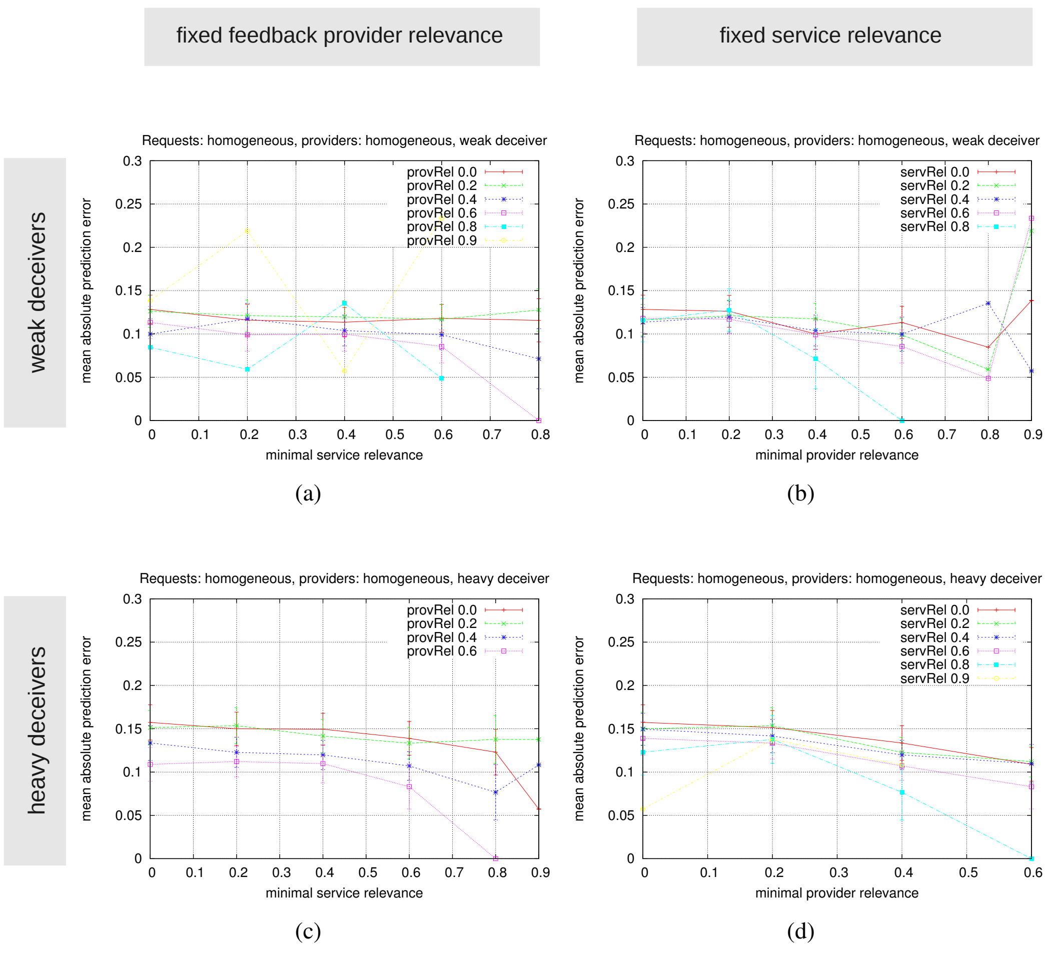


Figure 9.17.: Relevance charts for test data configuration 1 - homogeneous requests, homogeneous providers, weak deceivers (upper part)/heavy deceivers (lower part): mean absolute prediction error based on test runs performed with fixed feedback provider relevance thresholds depending on the minimal service relevance of the feedback items that have been used for the prediction (left), mean absolute prediction error based on test runs performed with fixed service relevance thresholds depending on the minimal feedback provider relevance of the feedback items that have been used for the prediction (right)

have a large impact on the prediction quality. The reason for this is, that the test data requests are homogeneous in the sense that they all refer to the same computer item type, but are still quite different with respect to the requirements specified for the single attributes. The same holds for the service relevance. As can be seen, the prediction quality increases with both, increasing threshold for feedback provider and service relevance. We also see, that just increasing both, the threshold for the feedback provider and the service relevance leads to the best prediction quality that can be achieved for this test data set. Hence we can conclude, that both types of relevance information, i.e. both types of context information, that have been elicited can be successfully leveraged to increase the prediction quality and complement each other. For very high relevance thresholds, we observe that the prediction error suddenly increases (e.g. Figure 9.17(b)). This is due to the fact that only a low number of feedback items yield such a high relevance value and thus less than 5 feedback items were used for the prediction. This in turn resulted in a worse prediction quality. The influence of the number of feedback items, that are leveraged for the prediction, on the prediction quality will be analyzed later on in this section. The results depicted in Figure 9.16, indicate that the parameter configurations **-*-AttrSimM* and **-*-WAttrSimM*, that used fine-grained, i.e. attribute-specific similarity information (Section 6.6.3), to determine the relevance of the single feedback items for the prediction, significantly outperformed parameter configurations *AdjM-OvSim-RootSim*, which leveraged only overall similarity information (Section 6.6.2). This does not hold for configuration *AdjM-OvSim-WAttrSimM*, which used the weighted attribute-specific similarity mean as an overall similarity measure. While tests performed with the parameter configurations *AdjM-OvSim-AttrSimM, AdjM-SimProd-AttrSimM, AdjM-SimProd-RootSim* and *AdjM-SimProd-WAttrSimM* achieved a reduction of the mean prediction error close to 0.0, the tests performed with the configurations *AdjM-OvSim-RootSim* and *AdjM-OvSim-WAttrSimM* reduced the mean error to about 0.05.

Configuration 2 - homogeneous requests, heterogeneous providers The main results for this test data configuration are depicted in Figure 9.18. Figure 9.19 shows the corresponding relevance charts. At first sight, it seems to be surprising that the base mean prediction error for this data set is even lower than that for the purely homogeneous configuration 1 (it is about 0.1), although the test data services are heterogeneous. The explanation for this is, that most of the test data offers do not fit very well to the test data requests and thus most of the request-offer pairs yield a very low matching degree (those who refer to different computer item categories). In fact, they just might match with respect to the price requirements specified in the request, but do not match with respect to the required computer item characteristics. Hence, the effect of cheating behavior on these pairs' actual matching degree, i.e. the consumer rating, is only marginally (in terms of absolute values). As a consequence, the consumer ratings for those request-offer pairs are all relatively low. This leads to low prediction errors, even when using no relevance information at all. The charts depicted in Figure 9.19 support this argumentation. As can be seen, the service relevance has only a low impact on the prediction quality (see Figures 9.19(b) and 9.19(d)). As in the case of test data configuration 1, the results for this test data configuration (Figure 9.18), indicate that parameter configurations that used fine-grained, i.e. attribute-specific similarity information, to determine the relevance of the single feedback items for the prediction, significantly outperformed those which leveraged only overall similarity information. Parameter configuration *AdjM-OvSim-WAttrSimM*, that used the weighted attribute-specific similarity mean as an overall similarity measure, again performed worse than all other parameter configurations that were based on fine-grained

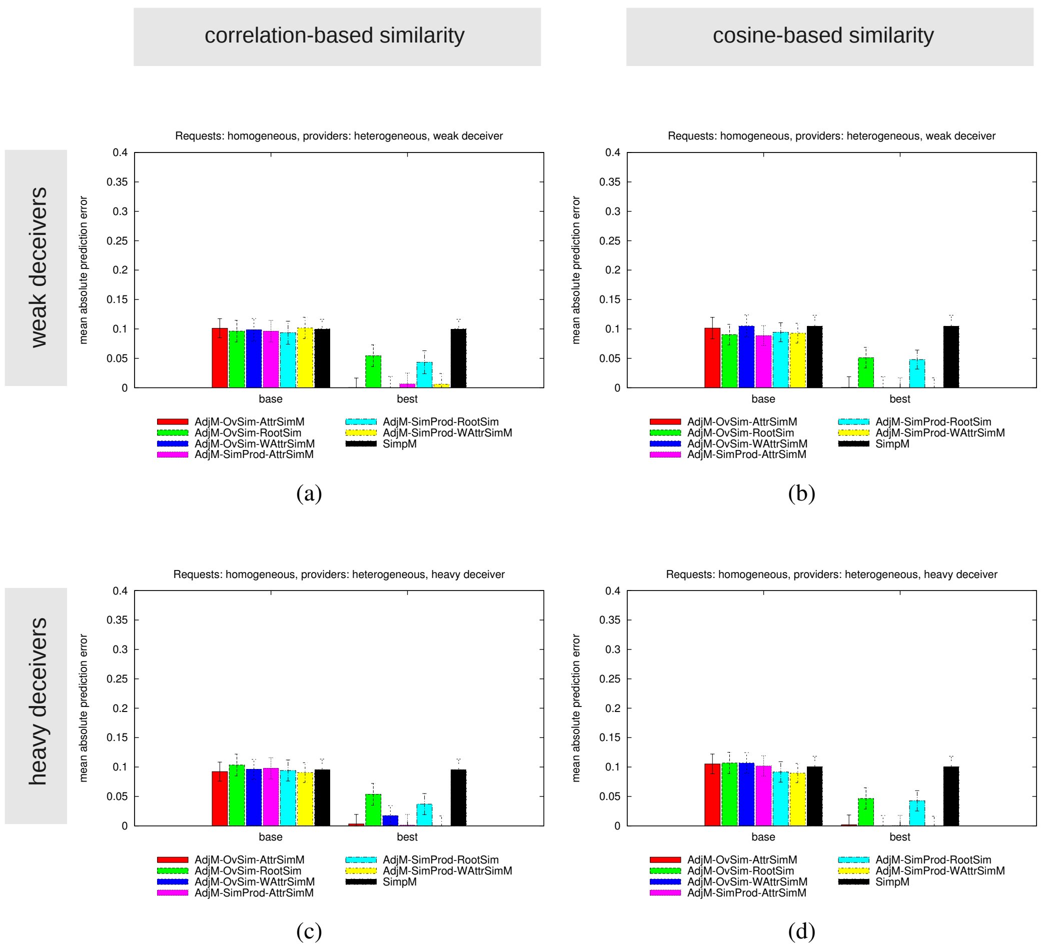


Figure 9.18.: Evaluation results for test data configuration 2 - homogeneous requests, heterogeneous providers, weak deceivers (upper part)/heavy deceivers (lower part): mean absolute prediction error based on test runs performed with the correlation-based measure (left) and the cosine-based measure (right)

similarity information. While tests performed with the parameter configurations *AdjM-*-AttrSimM* and *AdjM-SimProd-WAttrSimM* achieved a reduction of the mean prediction error close to 0.0, the tests performed with the configurations *AdjM-*-RootSim* reduced the mean error to about 0.05.

Configuration 3 - heterogeneous requests, homogeneous providers Figure 9.20 depicts the main results for this test data setting. Figure 9.21 shows the corresponding relevance charts. As in the results for test data configuration 2, the base mean prediction error is low (about 0.15). However, it is slightly higher for this test data setting. In

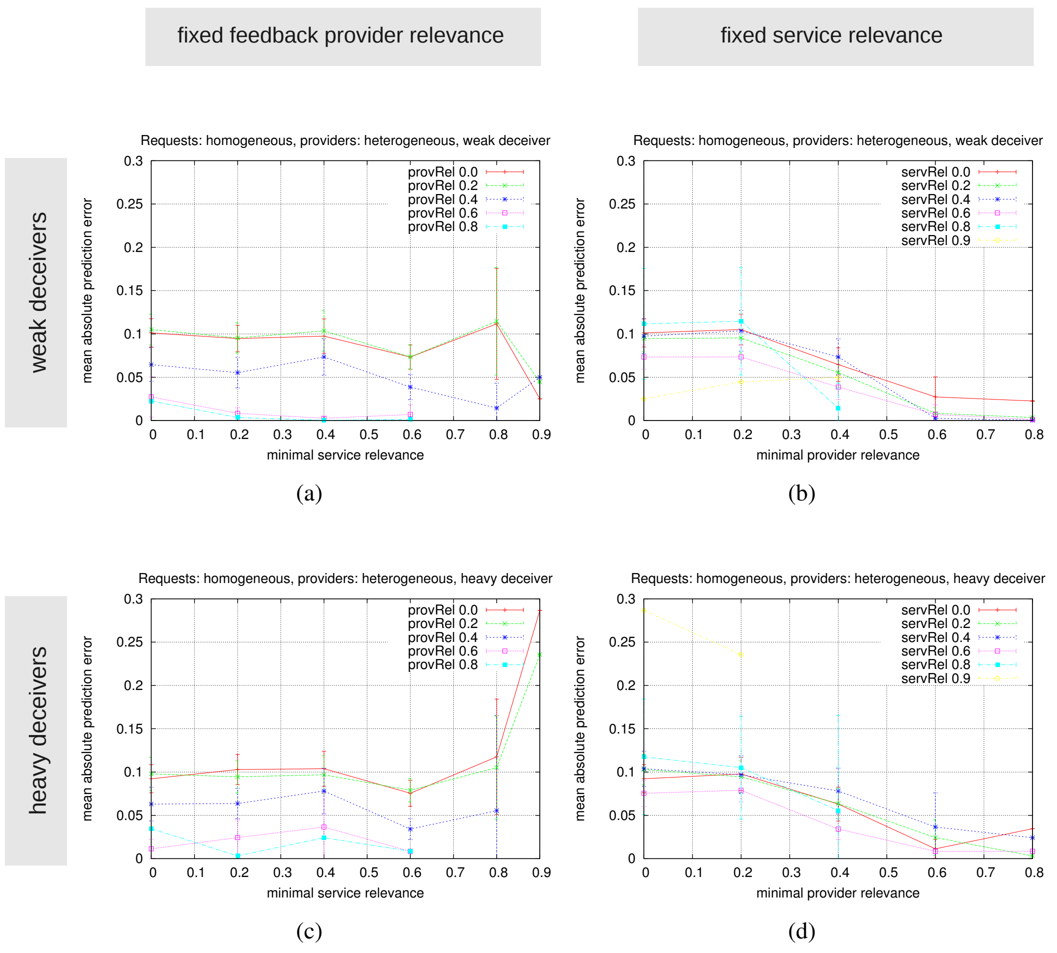


Figure 9.19.: Relevance charts for test data configuration 2 - homogeneous requests, heterogeneous providers, weak deceivers (upper part)/heavy deceivers (lower part): mean absolute prediction error based on test runs performed with fixed feedback provider relevance thresholds depending on the minimal service relevance of the feedback items that have been used for the prediction (left), mean absolute prediction error based on test runs performed with fixed service relevance thresholds depending on the minimal feedback provider relevance of the feedback items that have been used for the prediction (right)

fact, the situation is quite similar to that of configuration 2 (having heterogeneous requests/providers and homogeneous providers/requests). However, due to the heterogeneity of the test data requests, the price requirements specified in those requests vary much more than in test data configuration 2 and as a consequence, the consumer provided ratings are more diverse. This in turn results in both, a higher base mean prediction error and a higher mean prediction error when leveraging relevance information. Similar to the test

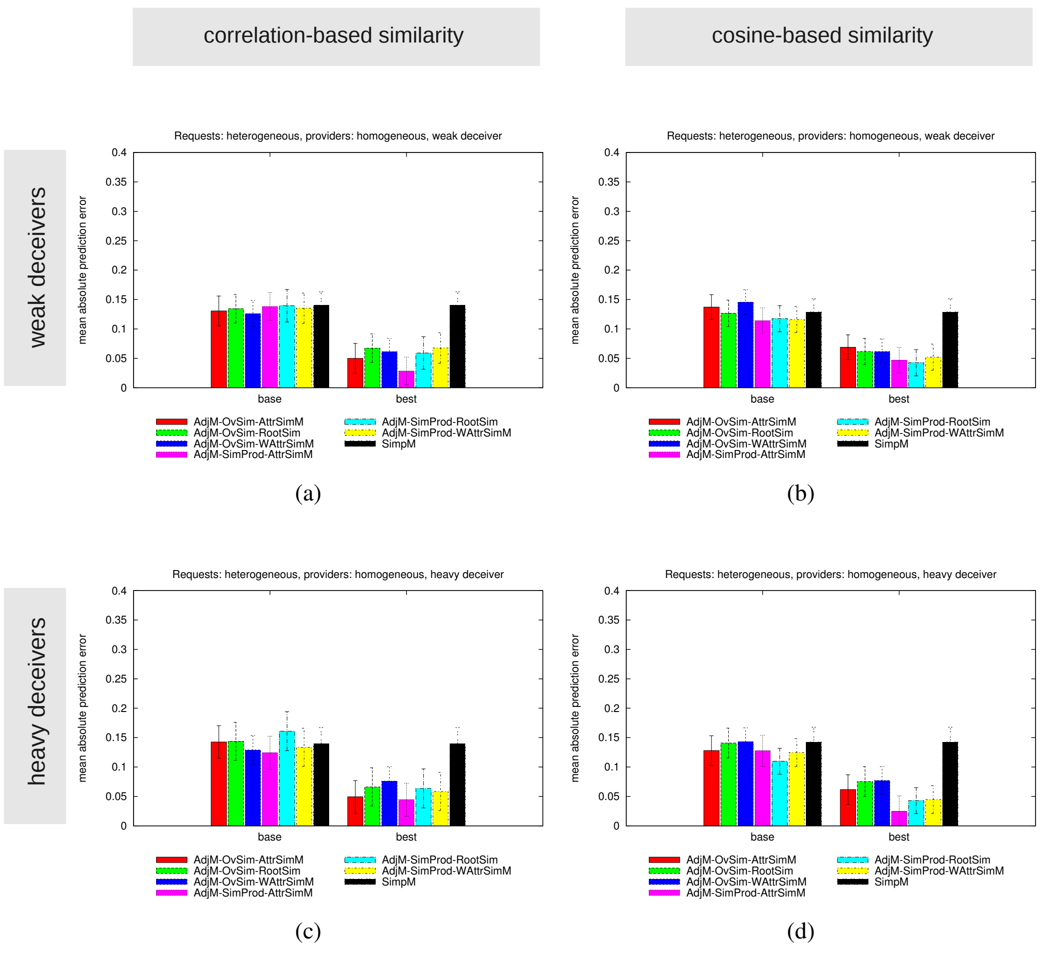


Figure 9.20.: Evaluation results for test data configuration 3 - heterogeneous requests, homogeneous providers, weak deceivers (upper part)/heavy deceivers (lower part): mean absolute prediction error based on test runs performed with the correlation-based measure (left) and the cosine-based measure (right)

data configurations 1 and 2, the test results indicate that parameter configurations using fine-grained, i.e. attribute-specific similarity information, to determine the relevance of the single feedback items for the prediction, outperformed those which leveraged only overall similarity information. Again, this does not hold for parameter configurations that used the weighted attribute-specific similarity mean as an overall similarity measure (*-*-*WAttrSimM*). However, in contrast to the previously considered test data configurations, the result was not significant and the achieved mean prediction errors for the parameter configurations *AdjM-*-AttrSimM* were higher (about 0.05). As can be also seen, there is no advantage of using the product of attribute-specific similarity and overall similarity as a relevance measure instead of simply using the overall similarity as an indicator for feed-

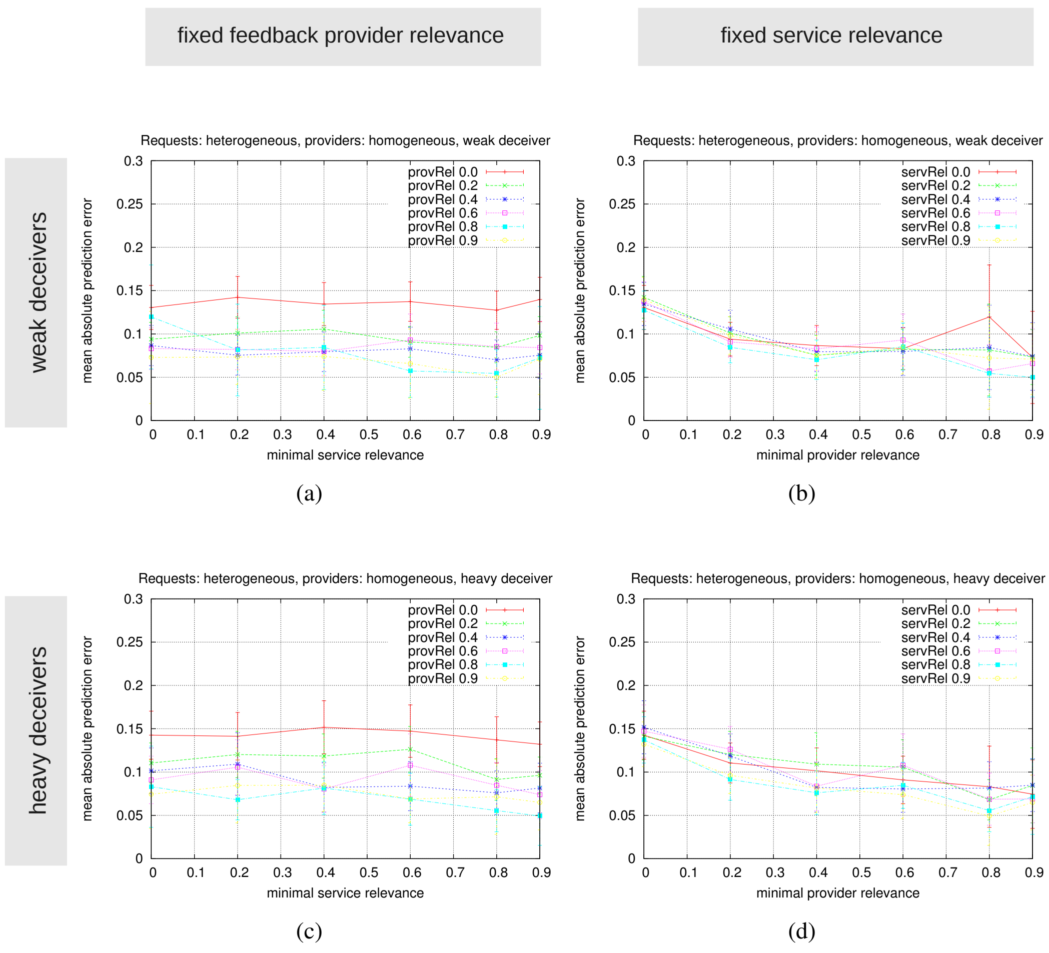


Figure 9.21.: Relevance charts for test data configuration 3 - heterogeneous requests, homogeneous providers, weak deceivers (upper part)/heavy deceivers (lower part): mean absolute prediction error based on test runs performed with fixed feedback provider relevance thresholds depending on the minimal service relevance of the feedback items that have been used for the prediction (left), mean absolute prediction error based on test runs performed with fixed service relevance thresholds depending on the minimal feedback provider relevance of the feedback items that have been used for the prediction (right)

back relevance. This was expected, since the charts depict the prediction error with respect to the predicted rating for the root attribute. Hence, the attribute-specific similarity of a feedback item, i.e. in this case its similarity with respect to the root attribute, coincides with the feedback item's overall similarity. Again, computing the overall similarity of a feedback item as the weighted attribute-specific similarity mean is not just not superior to determining it as the simple attribute-specific similarity mean, but even inferior (however, not significantly).

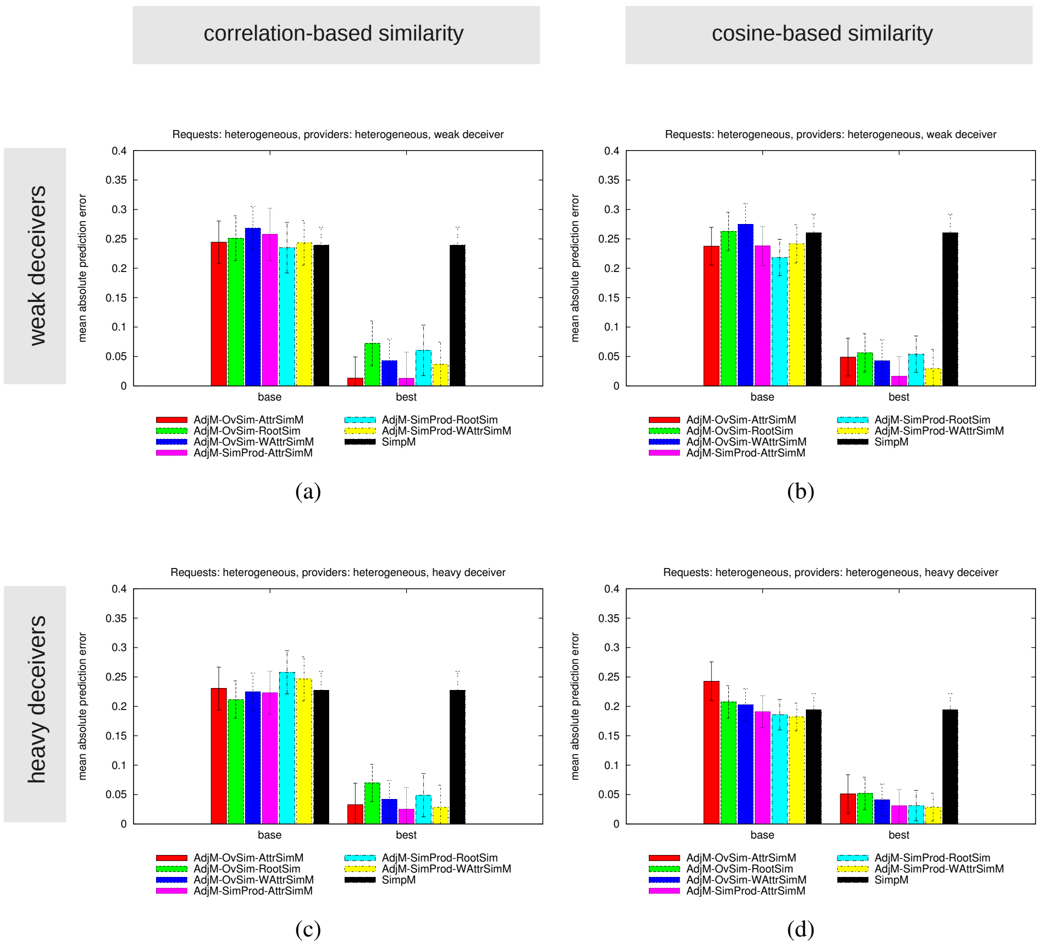


Figure 9.22.: Evaluation results for test data configuration 4 - heterogeneous requests, heterogeneous providers, weak deceivers (upper part)/heavy deceivers (lower part): mean absolute prediction error based on test runs performed with the correlation-based measure (left) and the cosine-based measure (right)

Configuration 4 - heterogeneous requests, heterogeneous providers The main results for this test data configuration are depicted in Figure 9.22. Figure 9.23 shows the corresponding relevance charts. As can be expected for a purely heterogeneous test data configuration, the base mean prediction error is much higher than for the other test data configurations (about 0.25). As for the test data configurations 1-3, we observe that the mean prediction quality increases with both, increasing threshold for feedback provider and service relevance. However, the impact of the service relevance on the prediction qual-

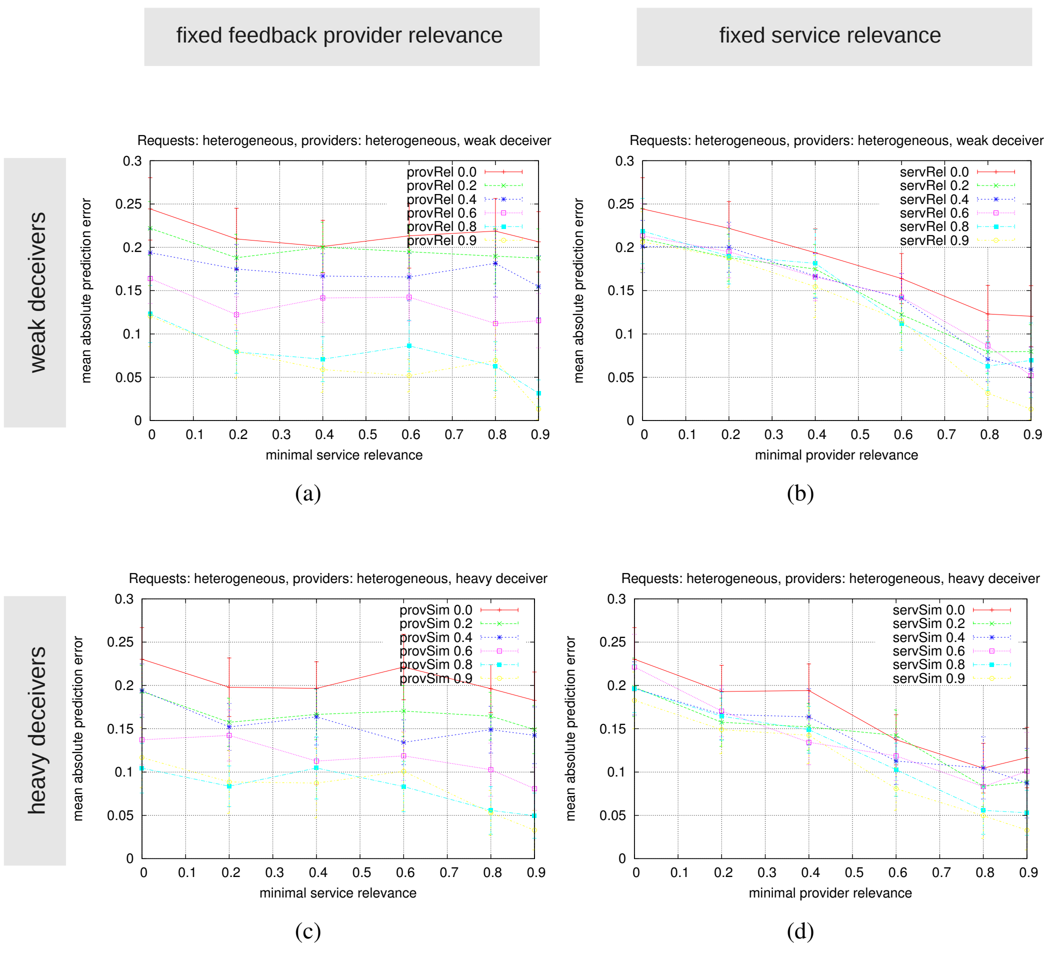
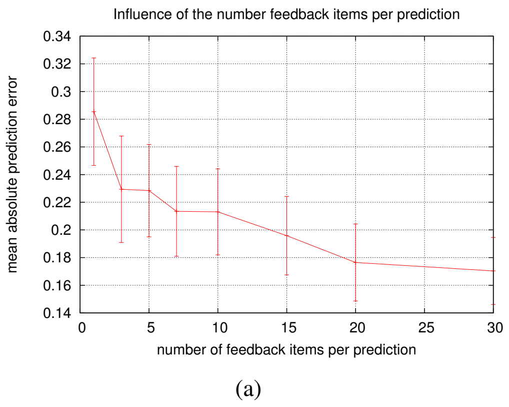


Figure 9.23.: Relevance charts for test data configuration 4 - heterogeneous requests, heterogeneous providers, weak deceivers (upper part)/heavy deceivers (lower part): mean absolute prediction error based on test runs performed with fixed feedback provider relevance thresholds depending on the minimal service relevance of the feedback items that have been used for the prediction (left), mean absolute prediction error based on test runs performed with fixed service relevance thresholds depending on the minimal feedback provider relevance of the feedback items that have been used for the prediction (right)

ity is much lower (e.g. cf. Figures 9.23(a) and 9.23(b)). Similar to the other test data configurations, the results indicate that parameter configurations that used fine-grained similarity information to determine the relevance of the single feedback items for the prediction, outperformed those which leveraged only overall similarity information. Again, this does not hold for parameter configurations of type *AdjM-*-WAttrSimM*, that used the weighted attribute-specific similarity mean as an overall similarity measure. However, this result is not significant. In particular, tests performed with the parameter configurations *AdjM-*-AttrSimM* achieved a reduction of the mean prediction error to about 0.01 for the test data configuration with weak deceivers and 0.03 for the configuration with heavy deceivers. Tests performed with the configurations *AdjM-*-RootSim* reduced the mean error to about 0.07 for the test data configurations with weak and heavy deceivers.

In any case, the quality of the rating prediction relies on the availability of sufficiently detailed similarity information. The more fine-grained the similarity information, the higher the prediction quality. Due to his privacy concerns, a user might not be willing to share fine-grained similarity information for each request attribute. Hence, the prediction quality that will be achieved in a real world scenario, will lie between the quality that was achieved based on the test setting *AdjM-OvSim-RootSim*, which leveraged just the overall matching degree for the similarity calculation (rating prediction based on coarse-grained similarity information (Section 6.6.2)), and the quality that was achieved based on the best test setting, *AdjM-OvSim-AttrSimM*, which assumed that similarity information for every single aspect of a service interaction (rating prediction based on fine-grained similarity information (Section 6.6.3)) is available. Hence, e.g. in case of test data configuration 4, the actual mean absolute prediction error would lie in the interval $[0.033-0.037, 0.070+0.032]$, i.e. with probability 0.95 will be below 0.102.



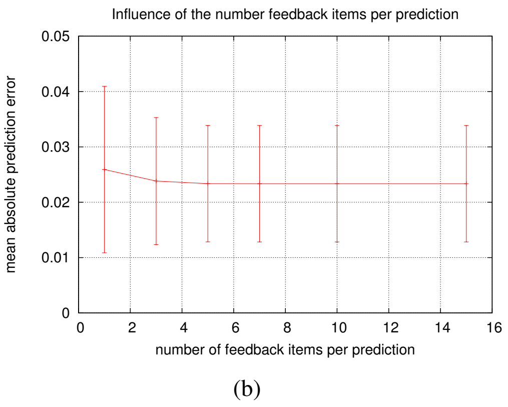


Figure 9.24.: Influence of the number of feedback items on the base mean prediction error (a) and the lowest mean prediction error that is achieved when using relevance filters (b)

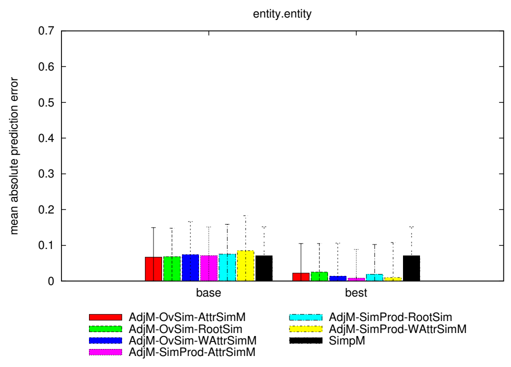
Influence of the number of feedback items on the prediction quality The charts depicted in Figure 9.24 illustrate the influence of the number of feedback items that are used for the prediction of a single rating on the prediction quality. They show the mean absolute prediction error for the overall judgment depending on the number of feedback items used for the prediction. Depicted are the results for test runs performed with the correlation-based similarity measure on test data configuration 4 (heterogeneous requests and providers) with heavy deceivers[13] and the parameter setting *AdjM-OvSim-AttrSimM*, which performed best for all test data configurations. While the chart depicted in Figure 9.24(a) shows the mean prediction error that was achieved when having used feedback contributions of arbitrary relevance for the prediction, the chart depicted in Figure 9.24(b) displays the lowest mean prediction error, that was achieved when having used only highly relevant feedback items for the prediction, i.e. when having used relevance filters.

As Figure 9.24(a) illustrates, weighting feedback contributions by their relevance had a positive influence on the prediction quality. The higher the number of feedback items, the higher the influence of the relevance weighting and thus the higher the prediction quality. However, just weighting feedback items according to their relevance was not sufficient in order to achieve a good prediction quality. The best prediction quality (mean prediction error of about 0.17) achieved just by weighting feedback contributions was for test runs that used 20 feedback items for the prediction. Further increasing the number of feedback items did not result in a better prediction quality. As can be seen in Figure 9.24(b), having only used highly relevant feedback items for the prediction, i.e. completely discarding non-relevant feedback items instead of assigning a low weight to them, much more efficiently reduced the mean prediction error (to about 0.025). As was expected, the number of feedback items had nearly no impact on the best prediction quality that was achieved. This is due to the fact that all leveraged feedback items were highly relevant. For a number of 5 feedback items, the best prediction quality was achieved and did not further improve by using more feedback contributions.

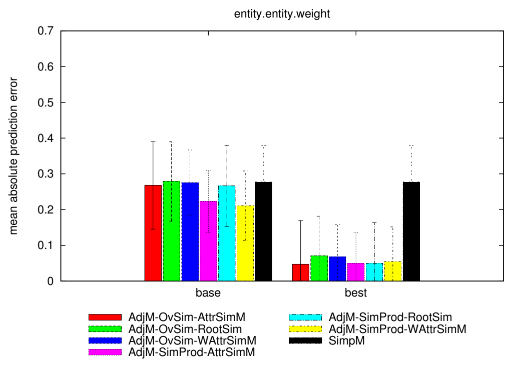
Attribute-specific prediction quality Figure 9.25 exemplarily illustrates the attribute-specific evaluation results and their characteristics. It depicts the attribute-specific prediction quality (the mean absolute prediction error) for the attributes *entity.entity*, *entity.entity.weight*, *entity.entity.model* and *entity.entity.manufacturer* (cf. Figure 9.14), performed with the correlation-based similarity measure on test data configuration 4 (heterogeneous requests and services) with heavy deceivers[13].

When looking at attribute-specific judgment prediction, the attribute-specific relevance of feedback items becomes an important factor for the prediction quality. This is evidenced by the attribute-specific evaluation results, depicted in Figure 9.25. As can be seen particularly in the charts for the attributes *entity.entity.model* and *entity.entity.manufacturer* (Figures 9.25(c) and 9.25(d)), the parameter configurations *AdjM-SimProd-**, that considered the attribute-specific similarity of feedback items, i.e. those that used the product of overall and attribute-specific similarity as a measure of feedback relevance, performed better than those that considered only the overall similarity of a feedback item (*AdjM-OvSim-**). This effect is less apparent for attributes that make up a large part of the request tree, such as *entity.entity* (cf. Figures 9.25(a) and 9.14), since in those cases the overall similarity is a good estimation for the attribute-specific similarity. Among the test settings that considered attribute-specific similarity when weighting the different feedback contributions,

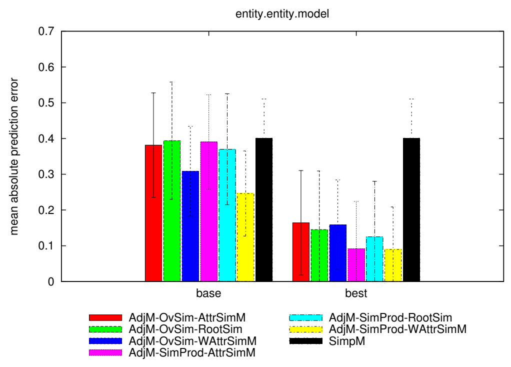
[13]We chose this test data configuration, since it resulted in the worst prediction quality for the predicted overall rating and thus seems to be the most challenging test data configuration.



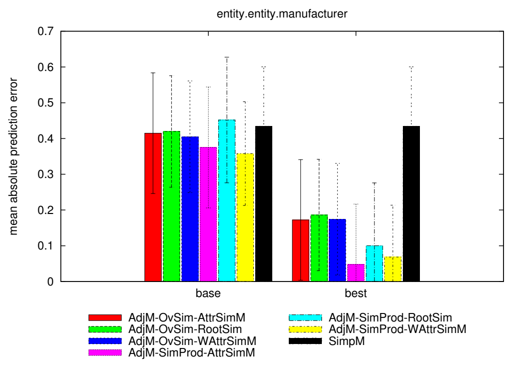
(a)



(b)



(c)



(d)

Figure 9.25.: Mean absolute prediction error for the attributes *entity.entity*, *entity.entity.weight*, *entity.entity.model* and *entity.entity.manufacturer*

test configuration *AdjM-SimProd-AttrSimM*, that used the mean of the attribute-specific similarities as an overall similarity measure resulted in the lowest prediction error.

Comparing the attribute-specific prediction errors with those that result when predicting the overall judgment for a given request-service-pair (i.e. the judgment the user would provide for the root attribute) based on the same test data set, we observe that attribute-specific predictions are subject to higher base mean prediction errors. This particularly holds for less complex attributes such as *entity.entity.weight*, *entity.entity.model* and *entity.entity.manufacturer* (cf. Figures 9.25(b), 9.25(c) and 9.25(d)), that have either a small number of subattributes or no subattributes at all (cf. Figure 9.14). For example, the base mean prediction error for the attribute *entity.entity.model* is about 0.4, while the base mean overall prediction error for the same test data is 0.25 (cf. Figure 9.22(c)). The reason for this is, that attribute-specific judgments for a given set of service requests and offers typically vary much more than their overall judgments and hence result in higher prediction errors when aggregated for the purpose of prediction. This is particularly true for non-complex attributes with nominal values. We illustrate this with an example. Consider the attribute *entity.entity.model* in a request for services that offer desktop PCs. Users explicitly list acceptable attribute values, i.e. acceptable models, and specify a matching value for each of those values. Then possible judgments for the attribute range from 0 for those offers that provide non-acceptable models to a typically high rating for those offers that provide acceptable models. This effect is less pronounced with increasing complexity of the attribute (increasing number of subattributes), since the judgments for the single subattributes are aggregated in various request-specific ways (see e.g. attribute *entity.entity* in Figure 9.25(a)).

When looking at nominal attributes, the observation that attribute-specific predictions are subject to higher prediction errors than the predictions of overall judgments, does not only hold for the base mean prediction error, but also for the best prediction error that was achieved when having used only a fraction of highly relevant feedback items (those that exceeded certain relevance thresholds). The reason for this lies in the way we modeled misbehavior and preferences (cf. Section 9.4.2). In the case of numerical attributes such as *entity.entity.weight* the modified values of service instances that have similar values, i.e. that have a high service relevance, are likely to be also similar and thus, due to the continuous nature of the preference functions for those attributes, are likely to be judged similarly by the same requester. This does not hold for nominal attributes, where the modified attribute value is uniformly chosen from the set of possible values and thus is rather arbitrary. This also holds for the resulting judgment. As a consequence, service relevance has no impact on the prediction accuracy for nominal attributes, which results in higher prediction errors. Nonetheless, modeling similar preferences and a similar type of misbehavior for similar service instances of nominal attributes is not an option. The reason for this is, that this would be rather hard and time-consuming, since it would require that we are provided with functions that indicate the similarity of different attribute values. Considering our test data, these functions would be rather hard to obtain, e.g. we would require a similarity function for different desktop PC models and different desktop PC manufacturers. Moreover, it would be unrealistic to assume that if we had such similarity functions at our disposal, they would be appropriate for each user. Consider for instance the attribute *entity.entity.manufacturer*. It is unlikely that any two service consumers share the same similarity function for desktop PC manufacturers, e.g. that any two consumers consider *Asus* to be similar to *Samsung*. Hence, modeling the test data in the desired way would require to have not only a realistic similarity function for each nominal attribute, but for each pair of request and nominal attribute.

Unfortunately, none of the presented results is statistically significant. The reason for this is that just a fraction of the requests, that were contained in the test data set, specified the depicted attributes. Hence, the results are based on a relatively low number of test runs (20 runs per parameter configuration).

Finally, we would like to comment on a fact that is related to the availability of relevant feedback, rather than to the characteristics of the prediction quality when sufficient feedback is available. Providing very detailed, i.e. attribute-specific feedback information implies a high judgment effort and reveals a high amount of personal information. As a consequence, it is less likely that a user will provide a large number of attribute-specific judgments and is willing to provide fine-grained, i.e. attribute-specific similarity information, particularly for attributes that refer to minor aspects of the consumer's service requirements. Hence, in a real world setting, we can expect to have a worse prediction quality for attribute-specific judgments than that achieved for our (semi-)artificial test data. However, this is not as unsatisfactory as it seems, since the less important a service aspect is, the less likely it is that the user is actually interested in the result of the judgment prediction.

9.4.5. On the Influence of Inaccurate Requirements Models

Performance predictions as introduced in this thesis are based on consumer judgments as well as on similarity information derived from the (partial) matching degrees between a judgment provider's service request and the available service offers. So far, we assumed, that a user's service request perfectly describes his service requirements. However, in our solution, service requests are derived from a system-maintained requirements model. As shown in Section 8.3, this model complies quite well with the user's actual service requirements, but does not perfectly describe them. This might affect the quality of the similarity information that is used for the judgment prediction and thus the quality of the performance predictions made by the system. In this section, we will therefore investigate this issue.

Evaluation methodology To account for the inaccuracy of requirements models and thus the inaccuracy of the derived service requests, we had to modify our original test data model that has been introduced in Section 9.4.2. Consider the original data model. So far, we assumed that the service request derived from the user's request model corresponds with the user's actual service requirements and thus both, the posed request and the actual requirements, could be modeled by the same, single request. Inaccuracy of service offer descriptions was simulated by deriving a modified offer description from each offer, which was considered to describe the actual outcome of the service described by the original offer description. The matching degree between the user's request and a given offer was then determined by matching the request with the original offer description. Consumer ratings for a certain service based on a given request were derived by matching the request with the modified service instance.

By abandoning the assumption that a user's request perfectly describes his service requirements, consumer ratings can no longer be determined by matching the request with the modified service instance. They rather have to be derived by matching the user's actual service requirements with the modified service offers. However, doing so would require to deeply analyze how the system-maintained model and the service consumer's actual service requirements differ and how this can be simulated. Particularly the latter is hardly possible and would introduce additional assumptions about the applicability of the evaluation results. Hence, rather than modeling a user's actual service requirements, we still determined consumer ratings by matching a user's service request with the judged offer's modified offer description and then accounted for the difference between this matching degree and the judgment the user would actually provide. This was done by randomly changing the determined matching degree by a certain amount to derive the desired consumer judgment.

However, to be able to perform the evaluation in this way, we had to know by which amount the matching degree between the service request derived from a user's final request model (i.e. the system-maintained request model at the time the user makes his selection) and a certain service offer differed from that of the user's actual service requirements and the considered offer. We acquired this information by leveraging the data retrieved from the test users when evaluating our requirements elicitation and service selection mechanism (Chapter 8). In particular, for each test person, we had the final request model maintained by the system (SM_{final} in Figure 8.1) and a model of his actual service requirements (MM_{use} in Figure 8.1). Using these models, we determined the mean absolute deviation of an offers matching degree with respect to SM_{final} and that with respect to MM_{use} and also the resulting standard deviation from this mean. The offer descriptions that were used to determine this value were the same as in the previous tests. As argued before (Section 8.3.3), MM_{use} does not perfectly comply with the user's actual service requirements, which is mainly due to the test user's inability to correctly indicate the importance of their single requirements by means of importance weights. While some of the user-provided importance weights were sufficiently consistent (and thus also the resulting requirements models), some were not. For determining the mean deviation of the matching degrees, we therefore restricted ourselves to the fairly consistent models (those with a consistency ratio ≤ 0.1, i.e. 5 models out of 10).

As it turned out, the mean matching degree deviation measured was 0.21 with a variance of 0.05 (i.e. a standard deviation of 0.23). Hence, when generating consumer ratings in our evaluation, we changed the matching degree of the generated request and a given modified service offer by adding a random value taken from a normal distribution with mean 0.21 and standard deviation 0.23. The effects of model inaccuracy on the prediction quality were studied for the test performed with the correlation-based similarity measure on test data configuration 4 (heterogeneous requests and services) with heavy deceivers. We chose this test data configuration, since it resulted in the worst prediction quality for the predicted overall rating and thus seems to be the most challenging test data configuration. The evaluation was performed for the prediction procedure that leveraged coarse-grained similarity information, i.e. the parameter configuration *AdjM-OvSim-RootSim*, since information about the matching value discrepancy with respect to single service aspects could not be retrieved. This is due to the fact that not all test persons considered all service aspects. Hence, we would get either no or no reliable information about the deviation of attribute-specific matching degrees.

Results As it turned out, when accounting for the inaccuracy of the system-maintained request model, the prediction error increased from 0.08 ± 0.08 to 0.14 ± 0.03, i.e. by 0.06 in average. This was expected, since by relying on inaccurate requirements models, the quality of the derived similarity information was reduced. This value is still acceptable in the sense that predictions of this quality still allow to assess the risk that is associated with the execution of a certain service at a relatively fine-grained level. Moreover, the quality of predictions that are based on fine-grained similarity information is even likely to be better, since their quality under the assumption of absolute model consistency was better. Moreover, the evaluation was performed for the test setting that resulted in the worst prediction quality. Hence, its results refer to the worst case. However, we lack means to verify these hypotheses. We also cannot assess how the inaccuracy of requirements models affects the prediction quality of attribute-specific judgments.

9.4.6. Prediction Confidence

Confidence in a prediction is the higher, the lower the prediction error is. The latter depends on several parameters such as the number of feedback items used per prediction and the similarity threshold for the leveraged feedback items (cf. Section 9.4.4). Devising a confidence measure requires a function for the prediction error depending on the indicated parameters. We determined such a function for the mean absolute prediction error of an overall judgment, exemplarily for test data configuration 4 (heterogeneous requests and services) with heavy deceivers.

Based on the evaluation results presented in Section 9.4.4, we assumed a logarithmic function of the form $-a \ln x + b$, $a, b \in \mathbb{R}$, for the dependency of the absolute prediction error from the number of feedback items used per prediction. As can be seen in Figure 9.24, the parameters of this function differed depending on the similarity threshold[14] that was set for the feedback items. We modeled this by assuming a linear dependency of the parameter values from the similarity threshold. This means, for the error function $err(t, n)$, returning the mean absolute prediction error depending on the chosen similarity threshold t and the number n of feedback items used per prediction, we assumed the following form

$$err(t, n) = -(a_1 t + a_2) \ln n + (b_1 t + b_2), \tag{9.2}$$

$a_1, a_2, b_1, b_2 \in \mathbb{R}$. We fitted this function to the values retrieved for the considered test data configuration using least-squares fitting. The resulting function as well as the original data (the measured mean prediction errors for the similarity thresholds 0.0 and 0.9 and various numbers of feedback items per prediction as depicted in Figure 9.24) is depicted in Figure 9.26. The fitted parameter values are shown in Table 9.4. As it turns out, the quality of the predicted prediction error is quite high. The mean absolute deviation of the predicted mean prediction error from the actual mean prediction error of the test data is 0.0032 ± 0.0020.

[14]the similarity threshold for both, the feedback provider and the service similarity

Parameter	**Fitted value**
a_1	-0.0348599 ± 0.003417 (9.803%)
a_2	0.0321678 ± 0.002005 (6.234%)
b_1	-0.281587 ± 0.0073 (2.593%)
b_2	0.27865 ± 0.004526 (1.624%)

Table 9.4.: Fitted parameter values

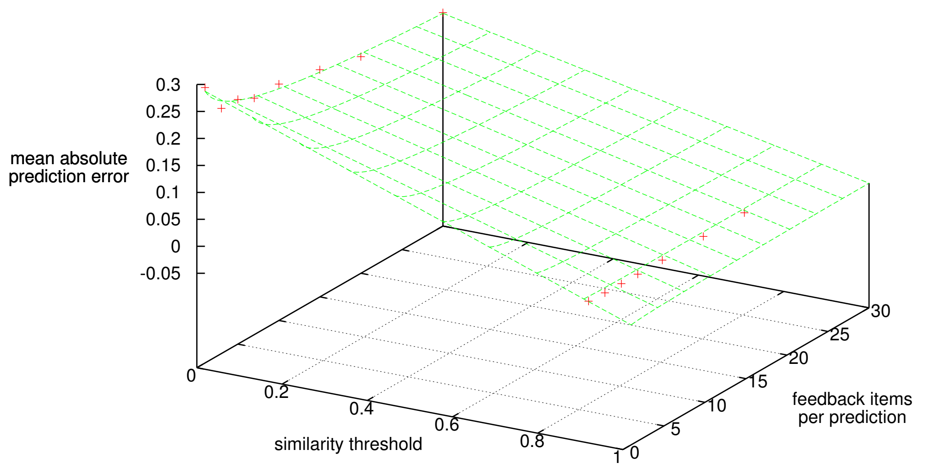


Figure 9.26.: Mean absolute prediction error depending on the similarity threshold and the number of feedback items used per prediction

A possible confidence measure $\mathit{conf}(t,n) \in [0,1]$ based on the prediction error is

$$\mathit{conf}(t,n) = 1 - \min(1, \mathit{err}(t,n)), \tag{9.3}$$

where a larger value indicates a higher confidence. Applying our approach to a given real world setting requires the determination of an appropriate parameter configuration for the suggested confidence measure. This has to be derived from example data acquired from the considered application domain.

Finally, we would like to remark, that the devised confidence measure provides the mean confidence in predictions with a given similarity threshold and a given number of leveraged feedback items. If needed, one can derive an upper bound for the confidence in an individual prediction by accounting for the variance of the prediction errors as measured for the test data. We would also like to note, that the prediction error might depend on other parameters than those that have been considered, such as the number of elements in the feedback matrix used to determine the similarity of an feedback item. Moreover, the prediction error depends on the attribute for which a judgment shall be predicted. This implies that one has to either provide an attribute-specific function for the prediction error or can derive a function for the upper bound of the prediction error. However, we did not investigate both of these aspects in this evaluation, since this would have gone beyond the scope of this thesis.

9.4.7. Discussion

Though we have extensively evaluated our performance prediction approach, we could not consider any aspects that might have an impact on the achieved prediction quality. In particular, we did not investigate the effect of context-dependent misbehavior of service providers. This would have required an intensive study of misbehavior in service provision, which would have gone beyond the scope of this thesis. Since the proposed prediction procedure takes the service and feedback provider context of leveraged feedback into account, we expect it to deal well with this kind of behavior. Hence, it would be worthwhile to investigate those issues more deeply.

Another issue concerning the generalizability of our evaluation results is, that the prediction quality that can be achieved by our approach depends on the similarity of the leveraged feedback information to the specific service selection task at hand. The more similar the judged service interactions are to the service selection task at hand, the better the prediction quality. Hence, whether the prediction quality that has been observed in our test runs can be achieved in a certain real world setting, depends on the availability of feedback that refers to service interactions that are sufficiently similar to a considered service interaction. This means, for each type of service interaction in a given real world setting, a sufficient number of service interactions should refer to a similar kind of service and should be based on similar service requirements. Whether this requirement can be fulfilled or not depends on the specific application scenario at hand.

A related issue is that our similarity-based approach to assess feedback relevance relies on a certain degree of stability in the set of available offers, i.e. the set of available services must not change very frequently. Otherwise, the feedback matrices derived for a certain feedback item would be scarce, which would have a negative effect on the quality of the similarity information inferred from these matrices. Hence, our approach to assess feedback relevance is not applicable in scenarios, where the entire set of services frequently changes. This is a problem it has is common with all collaborative-filtering-based mechanisms. However in contrast to these, our prediction procedure is able to derive meaningful context information for each service interaction that has been judged, whereas in collaborative-filtering systems those information can be only derived, if two considered users co-rated a sufficient number of services. This is a much more restrictive requirement.

Finally, it would be interesting to investigate whether our evaluation results can be confirmed for other datasets or possibly in real world settings. In any case, such a project would involve the determination of an appropriate parameter configuration for the suggested confidence measure. In case of a real world setting, this had to be derived from example data acquired from the considered application domain.

9.4.8. Summary

The key findings of our evaluation are:

- The mean prediction error for overall judgments was at most 0.07 (for the best parameter configuration).
- It was achieved by leveraging just 5 (relevant) feedback items.
- Since the quality of the rating prediction relies on the availability of fine-grained similarity information, the prediction quality achieved in a real world scenario, will be slightly worse. It will lie below 0.102.
- The mean prediction error for attribute-specific judgments using 5 feedback items was at most $0.09(+0.134)$ for the attributes that were considered in the test data. A larger test data set would be required to provide more accurate results.
- The low overall prediction error was achieved in a purely homogeneous setting as well as in settings with heterogeneous service providers and service requests.
- It was achieved for both, for service providers with heavy and weak deceiving behavior.
- Leveraging fine-grained similarity information to determine feedback relevance resulted in the lowest prediction errors.
- Weighting judgment contributions according to their relevance when making predictions had a positive but low impact on the prediction quality. Completely discarding non-relevant contributions was much more effective and resulted in low prediction errors.
- Both, information about the service relevance and the feedback provider relevance of the feedback items, that were used for the prediction, were successfully leveraged to increase the prediction quality and complemented each other. The higher the service relevance and the higher the feedback provider relevance of the leveraged feedback items, the better the prediction quality.
- Using the overall similarity of the feedback items that are used for the prediction as a relevance measure delivered good prediction results for the overall service judgment. When predicting attribute-specific judgments, the product of attribute-specific and overall similarity as a relevance measure delivered better prediction results.

- Weighting attribute-specific similarities to determine the overall similarity of a feedback item did not result in an improvement of the prediction quality.
- Relaxing the assumption of perfect compliance between the system-maintained request model and the user's actual requirements, results in a slight increase of the mean absolute prediction error for the overall judgment by 0.06, which is still acceptable. The result refers to the worst case scenario and the prediction procedure that is based on coarse-grained similarity information.

Summarizing these results, we conclude that the feedback mechanism that has been proposed in this thesis (Section 6.6) has been demonstrated to effectively exploit consumer feedback to predict the future performance of available services. This has been shown to be true, even if available feedback refers to service interactions that are diverse with respect to the services and requests (request models) that were involved (fulfillment of Requirement F.4). We also demonstrated, that this is enabled by both, leveraging detailed consumer feedback and taking the request and service (offer) context, in which a judgment was made, into account when using feedback to predict a service's future performance (fulfillment of the Requirements F.1 and F.5). Finally, using our evaluation results, we derived a confidence measure indicating the reliability of the predictions produced by the proposed prediction algorithm (fulfillment of Requirement F.6).

9.5 Summary and Conclusions

In the previous sections, we have shown that the requirements related to the feedback mechanism that has been proposed in this thesis (Requirements F.1 to F.8), which have been compiled in Section 6.1, are fulfilled.

- In particular, based on the concept of a feedback structure, we designed a model, that is capable of describing consumer feedback referring to multi-faceted service interactions and developed an effective mechanism for eliciting this kind of feedback. As has been shown, this elicitation mechanism flexibly and automatically adjusts to a consumer's willingness to provide feedback (**fulfillment of Requirement F.2** as shown in Section 9.3).
- We also demonstrated that both, the devised model and the elicitation procedure, ensure that elicited feedback is detailed, comprehensive, meaningful and appropriate in the context of a certain service interaction, even if the services and requests (request models) that are involved in the service interactions are diverse and refer to different application domains (**fulfillment of Requirement F.1** as shown in Section 9.2).
- By solely relying on the exchange of indirect information about the context of a judged service interaction, sharing of easily accessible personal information can be avoided. Moreover, the judgment providers themselves decide about the detailedness of the judgments they provide and thus about the amount of sensitive information that they are willing to share (**fulfillment of Requirement F.3** as shown in Section 9.2).

- We also provided a procedure that effectively exploits elicited consumer feedback to predict a service's future performance and thus to assess the risk that is associated with its execution. It accounts for the context-dependent nature of service performance and service judgments by taking the request and service context, in which a judgment was made, into account (**fulfillment of Requirement F.5** as shown in Section 9.4).
- As has been shown, the devised procedure is even effective, if available feedback refers to service interactions that are diverse with respect to the services and requests (request models) that were involved (**fulfillment of Requirement F.4** as shown in Section 9.4).
- We also provided a confidence measure, indicating how sure the system is about the predicted performance (**fulfillment of Requirement F.6** as shown in Section 9.4.6).
- Finally, we contributed means to effectively communicate feedback-derived knowledge to the user. In particular, feedback information are presented in a way that makes the user aware of the risk that is associated with the execution of a service (**fulfillment of Requirement F.7** as shown in Section 9.2.3), thereby accounting for different risk attitudes (**fulfillment of Requirement F.8** as shown in Section 9.2.3).

Hence, we conclude that **the operational goal of devising a feedback mechanism that is both, effective in terms of acquiring knowledge about the risk that is associated with the execution of a service and effective in terms of its ability to support service selection in the presence of this risk by effectively communicating this knowledge to the service consumer (Objective 2), is fulfilled**.

Part IV.

Final Considerations

10 Summary and Conclusions

We introduced Service Oriented Architectures (SOAs) as a powerful way of designing flexible and adaptive distributed applications based on loosely coupled, standalone services. We argued that beside the popularity of this principle for flexible, quick and low-cost application development in enterprises and for the integration of existing enterprise applications, there is recently an increasing trend of using Service Orientation based on Web Services as the underlying principle for offering and using functionality over the Internet. As has been indicated, experts expect this trend to continue and envision the emergence of an Internet of Services, a global, user-centric SOA based on easily accessible services offered over the Internet. In this thesis, we argued that the emerging Internet of Services has the potential to offer a new way for end-users, i.e. service consumers, to retrieve and use functionality over the Internet, that, particularly if complemented with semantic techniques, promises to be more effective and convenient than prevalent retrieval and usage schemes based on keyword search. However as discussed, existing approaches to (Semantic) Web Service retrieval are not well-suited to this new usage scenario, since they do not provide effective decision support for user-mediated service selection, which is required to enable well-founded service selection decisions at a large scale.

Thesis objective Therefore, the overall objective of this thesis was to develop a system that effectively supports service consumers, i.e. end-users, in efficiently making well-informed, balanced and consistent service selection decisions (Objective 1). In this context, we pointed out, that due to the fact that consumer requirements are constructive by nature and that information resources on the Internet are not necessarily trustworthy, the envisioned system has to account for the fact that, both, knowledge about the consumer's service requirements as well as knowledge about the capabilities of available service offers might be inaccurate and incomplete, i.e. is uncertain.

In Part I of the thesis, we argued that this thesis goal can be achieved by accomplishing the following two operational objectives:

- Firstly, dealing with uncertainty arising from inaccurate and incomplete knowledge about offered service capabilities requires the conception of a feedback mechanism that is both, effective in terms of acquiring knowledge about the risk that is associated with the execution of a service and effective in terms of its ability to support service selection in the presence of this risk (Objective 2).

- Secondly, a method for requirements elicitation and service selection that incrementally and interactively acquires knowledge about a user's evolving service requirements and by leveraging this knowledge effectively supports service consumers in making well-informed, balanced and consistent service selection decisions in an efficient manner, is required (Objective 3).

Contributions In the course of this thesis (Part II), we designed and detailedly described a system for requirements elicitation and service selection, that meets these objectives and thus fulfills the thesis goal. In particular, we devised an incremental and interactive approach to requirements elicitation and service selection, which alternates phases of intermediate service recommendation and phases of informal requirements specification based on the characteristics of the presented service alternatives or by using a modifiable graphical requirements representation. During that process, the user incrementally develops his service requirements and preferences and finally makes a selection decision. As part of this system, we described a model of the consumer's service requirements and preferences that is maintained to effectively support and guide the user in his tasks. We explained how uncertainty about the service consumer's true requirements and preferences, that is caused by missing and/or inaccurate knowledge, is explicitly represented within this model. Moreover, we detailed on how the model is continuously updated based on the user's interactions to accurately reflect the systems's growing knowledge about the user's service requirements and preferences. We also explained how suitable service offers can be retrieved based on this uncertain requirements model and how they can be appropriately displayed to enable effective decision making. Finally, we argued how model uncertainty can be effectively reduced to contribute to the efficiency of the requirements elicitation and service selection process.

In addition to that, we provided a feedback mechanism, that allows to assess the risk that is associated with the execution of a service. This includes the specification of a feedback model that is capable of detailedly and meaningfully describing consumer feedback related to service interactions, the development of a user-adaptive, flexible and privacy-aware method for the elicitation of such feedback and a prediction algorithm that effectively exploits available consumer feedback to assess the degree and kind of risk that is associated with the execution of an offered service. Thereby, the suggested solution accounts for the subjective and context-dependent nature of consumer feedback and exploits consumer experiences made in one context to infer knowledge about a service's behavior in another context. Finally, we explained how feedback-derived information can be leveraged to make users aware of the risk that is associated with the execution of a service and thereby accounting for different risk attitudes.

Implementation, evaluation and conclusion We also implemented and extensively evaluated our approach (Part III). In particular, we theoretically verified the appropriateness of the requirements model and the kind of requirements elicitation and service selection procedure that have been devised. We also provided the results of two extensive user studies that show the effectiveness of the devised requirements elicitation and service selection mechanism in terms of its ability to provide decision support for well-founded and efficient service selection (fulfillment of Objective 3). In addition to that, we evaluated the developed feedback mechanism. More specifically, we theoretically argued in favor of the appropriateness of the devised feedback model and demonstrated the effectiveness of the devised feedback elicitation procedure by means of a user study. Moreover, we simulatively demonstrated the effectiveness of the devised performance prediction procedure based on a real world data set, showed that these results are even valid if the model of the feedback provider's service requirements is inaccurate to some degree and derived a confidence measure for the predictions delivered by the procedure. Finally, we theoretically showed that feedback-derived information are presented in a way that makes the user aware of the risk that is associated with the execution of a service and thereby accounts for different risk attitudes (fulfillment of Objective 2).

As a result of these evaluations, we verified that **the two operational objectives of this thesis (Objective 2 and 3) are fulfilled (Sections 8.5 and 9.5) and thus the overall thesis objective of developing a system that effectively supports service consumers in efficiently making well-informed, balanced and consistent service selection decisions (Objective 1) has been achieved**. We hope, that by indicating a way to effective end-user support for Semantic Web Service selection, we could both, contribute to the realization of the emerging Internet of Services and to the practical adoption of Semantic Web Service technology.

11 Future Work

The aim of this thesis was to demonstrate the feasibility of end-user-mediated Semantic Web Service selection at Web-scale and by this means to illustrate the benefits and the potential of using Semantic Web Service technology. Fulfilling this task required to consider a large number of different research issues and involved the application of a wide range of techniques to approach these. Discussing all these issues in detail would have exceeded the scope of a PhD thesis by far. Hence, the contributions of this thesis have to be understood as proof of concept and as such can be extended and improved in several ways. In addition to that, they raise new, complementary research issues and open up possibilities for new application scenarios, which shall be discussed in this chapter.

11.1 Modeling and Elicitation of Service Requirements

A number of possible improvements and extensions to the proposed approach refer to the requirements model that has been introduced in Section 5.4 of this thesis. As already discussed, the request model implementation supports only weighted sum as a connecting strategy, i.e. presumes that a user's preferences with respect to a set of attribute conditions are mutually independent (cf. Section 5.4.3). Though this is a valid assumption for many real-world service selection scenarios [KR93, FGE05], there are service requirements that cannot be expressed using an additive model, e.g. a user looking for a flower delivery service might want to indicate that he does not consider a certain service offer as acceptable, if it cannot deliver at a given, user-provided day, even if it is very cheap. A possible extension to the proposed request model would therefore provide support for additional connecting strategies that allow to express interdependent preferences, e.g. by allowing to multiply preference values as possible in DSD.

Moreover, there might be other types of preference functions that better describe a user's preferences over a range of instances than those discussed in Section 5.4.2. This particularly holds, if another application scenario is considered. It would be thus worthwhile to investigate how human preferences in a certain domain can be appropriately modeled and to finally implement these models as an extension to the system. In addition to the mentioned preference-related features, there are other features offered by the Diane Service Description Language, that are worth to be considered in our implementation, but are not yet supported by our system. This includes e.g. support for specifying multiple effects or for specifying multi attribute conditions, i.e. direct conditions that reference several attributes.

Beside of those extensions that affect the expressiveness of the requirements model, there are possible enhancements concerning the model's ability to represent uncertainty about a service consumer's requirements. As discussed in Section 5.4, the introduced request model does not account for uncertainty about the type of acceptable service instances as well as for uncertainty about the user's preference for a certain instance that has been specified in an in-condition. Leveraging knowledge about these types of uncertainty has the potential to improve decision making support, but would also introduce additional complexity to the system and thus might affect its efficiency. Hence, a deeper investigation of these issues is required.

Another set of possible enhancements does not refer to the requirements model itself, but rather to the process of learning this model from the user's interactions with the system. The effectiveness of the latter strongly depends on both, the learning procedure's ability to infer knowledge about a user's service requirements from its various interactions with the system and its ability to interpret a system user's actions correctly, i.e. by appropriately modeling the likelihood of a certain interaction given a specific requirement. Hence, it would be worthwhile to analyze how a human's service requirements affect its interactions with the available system elements to further improve the behavioral models in terms of likelihood functions that have been used in the introduced approach (cf. Section 5.9). Moreover, it would be promising to consider interaction types other than those that have been discussed in this work (cf. Section 5.9) to infer additional knowledge about a user's service requirements. For instance, one could draw conclusions from the fact, that a user sorted the presented service offers with respect to a certain attribute, or from the fact, that a certain recommended tradeoff opportunity was not chosen.

11.2 Visualization of Service Requirements and Interaction Opportunities

By enhancing its visualization features, the usability and effectiveness of the proposed system for requirements elicitation and service selection can be further improved. This involves the consideration of other/additional information filtering and summarizing techniques to identify relevant information, the use of other/additional visualization techniques and metaphors as well as the implementation of additional visualization features such as zooming or overview. Specific visualization issues related to the graphical requirements representation include the question of how large requirements models can be presented in a clear way and how importance weights can be visually encoded to allow for their easy comparison (cf. Section 8.4). In addition to that, several improvements to the critiquing and tradeoff support offered by the system come to mind. These involve the consideration of other visualization techniques that further improve the comprehensibility of recommended tradeoff opportunities as well as the application of yet unconsidered data mining and filtering techniques to extract common properties of the matching service offers more effectively and efficiently. Moreover, the set of possible tradeoff types might be extended, e.g. to allow for more fine-grained critiques such as "slightly more" and "much more".

To further improve the system's support for requirements specification, additional features such as automatic completion, e.g. of unique attribute values, or disabling non-applicable interaction opportunities can be added. This could also involve support for automatic extension and refinement of the user's request model based on the current usage context as proposed and discussed in [Kla06].

11.3 Elicitation of Consumer Feedback

As discussed in Section 9.3.4, there are a number of issues related to human judgments and their effective elicitation, that promise to initiate a particularly interesting line of multidisciplinary research. Among others, these include an in-depth analysis of the situational and psychological aspects that affect a user's choice to judge a certain service aspect, such as the consumer's satisfaction with the invoked service's performance on that aspect or simply the fact, that this aspect has been recommended as a judgment target. In this context, it would be helpful to identify classes of judgment providers that share behavioral patterns (stereotypes), since this would dramatically simplify the recommendation of judgment targets and would allow for helpful recommendations, even if knowledge about a user's previous judgment behavior is missing. Other issues refer to the consistency of human judgments and the service consumers' willingness to provide detailed judgments. Since the outlined field of research touches on topics pertaining to both, human behavior and computer science, it should be jointly addressed by researchers from these fields.

11.4 Performance Prediction

A major advantage of the feedback mechanism that has been proposed in Chapter 6 is, that it elicits and shares meaningful consumer feedback while at the same time revealing only little personal information. In this context, the question arises, if the amount and quality of the shared judgment information can be further reduced while maintaining a similar prediction quality. Nonetheless, the quality of the performance predictions that can be achieved by our approach is strongly affected by the availability of consumer feedback that is sufficiently detailed, refers to similar kinds of services and that is based on similar service requirements (cf. Section 9.4.4). It should therefore be examined more closely whether this requirement can be fulfilled in a certain service domain and application scenario.

Finally, examining whether and how context-dependent misbehavior of service providers impacts on the prediction quality that can be achieved by the proposed approach would have gone beyond the scope of this thesis. Nonetheless, this is an issue worth to be considered more deeply, particularly, since the proposed prediction procedure takes the service and feedback provider context of leveraged feedback into account and thus can be expected to deal well with context-dependent behavior.

11.5 Practical Deployment and Social Implications

As indicated on many occasions throughout this thesis, it would be desirable to deploy our approach in a real world system. This would have several benefits. Firstly, it would allow us to evaluate our solution to Semantic Web Service selection in a real world setting, i.e. based on consumers' actual service needs, under realistic usage circumstances and with a larger number and wider range of test users. This not just applies to the approach presented in this thesis. A running application that uses our solution to Semantic Web Service selection would also furnish the test data that are urgently required to evaluate other approaches that are based on Semantic Web Service technology under realistic conditions. However, the semantic information produced by such a real world system would not just be valuable from a evaluation perspective, but could also serve as a foundation for new and innovative applications that leverage semantic data. Moreover, semantic information about consumers' service requirements as well as detailed, semantically described consumer feedback would deliver valuable knowledge about typical service needs in a given usage scenario, about human judgment behavior and about misbehavior in service provision. Finally, a real world application based on Semantic Web Service technology would demonstrate the benefits of using extensive semantical knowledge and thus would generate incentives for enterprises to invest time and money in the creation and use of semantic information. In this context, it would be desirable to investigate whether the approach proposed in this thesis can be applied to other semantic service description frameworks, whether this generates additional benefits or whether this is impracticable. This would help to assess the degree of semantics that is actually needed to enable "computers and people to work in cooperation" [BLHL01] as envisioned by Tim Berners-Lee.

The availability of extensive semantic data provides not just benefits, but also raises serious privacy issues. This is due to the fact, that semantic information can be interpreted and processed automatically, which makes it much easier to acquire and combine user-specific information, that otherwise had to be derived by analyzing unstructured data using imprecise data mining techniques. Discussing possible social implications of an increased use of semantic technologies should therefore be also part of future research.

References

[AA09] Dimitrios Skoutas Anna Averbakh, Daniel Krause. Recommend me a service: Personalized semantic web service matchmaking. In Andreas Nauerz David Hauger, Mirjam Köck, editor, *Workshop on Adaptivity and User Modeling in Interactive Systems (ABIS 2009)*, 2009.

[AAS09] Daniel Krause Anna Averbakh and Dimitrios Skoutas. Exploiting user feedback to improve semantic web service discovery. In *8th International Semantic Web Conference (ISWC 2009)*, 2009.

[Ado05] Gediminas Adomavicius. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems*, 23(1):103, 2005.

[Agr06] Gennady Agre. Infrawebs designer - a graphical tool for designing semantic web services. In Jérôme Euzenat and John Domingue, editors, *AIMSA 2006*, volume 4183, pages 275–289. Springer, 2006.

[AIS93] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington, D.C., FebruaryJune–FebruaryAugust 1993.

[AK06] Bilal M. Ayyub and George J. Klir. *Uncertainty Modeling and Analysis in Engineering and the Sciences*. Chapman and Hall/CRC, 2006.

[AMK11] Gediminas Adomavicius, Nikos Manouselis, and YoungOk Kwon. Multi-criteria recommender systems. In *Recommender Systems Handbook*, pages 769–803. 2011.

[AMLM07] Gennady Agre, Zlatina Marinova, Tomás Pariente Lobo, and András Micsik. Towards semantic web service engineering. In Tommaso Di Noia, Rubén Lara, Axel Polleres, Ioan Toma, Takahiro Kawamura, Matthias Klusch, Abraham Bernstein, Massimo Paolucci, Alain Leger, and David L. Martin, editors, *SMRR CEUR Workshop Proceedings*, volume 243. CEUR-WS.org, 2007.

[ARH97] Alfarez Abdul-Rahman and Stephen Hailes. A distributed trust model. In *NSPW '97: Proceedings of the 1997 workshop on New security paradigms*, pages 48–60, New York, NY, USA, 1997. ACM.

[ARH00] Alfarez Abdul-Rahman and Stephen Hailes. Supporting trust in virtual communities. *Hawaii International Conference on System Sciences*, 6, 2000.

[AS94] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, pages 487–499. Morgan Kaufmann, 1994.

[BAH$^+$06] Shlomo Berkovsky, Lora Aroyo, Dominik Heckmann, Geert-jan Houben, Er Kröner, Tsvi Kuflik, and Francesco Ricci. Predicting user experiences through cross-context reasoning. In *14TH WORKSHOP ON ADAPTIVITY AND USER MODELING IN INTERACTIVE SYSTEMS*, 2006.

[BC02] Ralph Bergmann and Padraig Cunningham. Acquiring customers' requirements in electronic commerce. *Artif. Intell. Rev.*, 18(3-4):163–193, 2002.

[BDB05] Alistair P. Barros, Marlon Dumas, and Peter D. Bruza. The move to web service ecosystems. *BPTrends*, 3(3), December 2005.

[Ber54] Daniel Bernoulli. Exposition of a new theory on the measurement of risk. *Econometrica*, 22(1):23, 1954.

[BF08] Daniel Bachlechner and Kerstin Fink. Semantic web service research: Current challenges and proximate achievements. *International Journal of Computer Science and Applications*, 5(3b):117–140, 2008.

[BHK98] John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. pages 43–52. Morgan Kaufmann, 1998.

[BHOC07] H. Billhardt, R. Hermoso, S. Ossowski, and R. Centeno. Trust-based service provider selection in open environments. In *22nd ACM Symposium on Applied Computing*, pages 1375–1380, Seoul, 2007.

[BHY96] Robin D. Burke, Kristian J. Hammond, and Benjamin C. Young. Knowledge-based navigation of complex information spaces. In *AAAI/IAAI, Vol. 1*, pages 462–468, 1996.

[BHY97] Robin D. Burke, Kristian J. Hammond, and Benjamin C. Young. The findme approach to assisted browsing. *IEEE Expert*, 12:32–40, 1997.

[BKR08] Shlomo Berkovsky, Tsvi Kuflik, and Francesco Ricci. Mediation of user models for enhanced personalization in recommender systems. *User Modeling and User-Adapted Interaction*, 18(3):245–286, aug 2008.

[BLHL01] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, May 2001.

[BLP98] James r. Bettman, Mary frances Luce, and John w. Payne. Constructive consumer choice processes. *Journal of Consumer Research*, 25(3):187–217, December 1998.

[BMUT97] Sergey Brin, Rajeev Motwani, Jeffrey D. Ullman, and Shalom Tsur. Dynamic itemset counting and implication rules for market basket data. In *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data*, pages 255–264, Tucson, Arizona, USA, May 1997.

[Bou02] Craig Boutilier. A pomdp formulation of preference elicitation problems. In *Eighteenth national conference on Artificial intelligence*, pages 239–246, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence.

[Bro02] Andrei Broder. A taxonomy of web search. *SIGIR FORUM*, 36(2):3–10, 2002.

[Bur02] Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, November 2002.

[Bur07] Robin Burke. Hybrid web recommender systems. pages 377–408. 2007.

[BVMC05] Alessio Bosca, Giuseppe Valetto, Roberta Maglione, and Fulvio Corno. Lecture notes in computer science. In Boualem Benatallah, Fabio Casati, and Paolo Traverso, editors, *ICSOC*, volume 3826, pages 588–593. Springer, 2005.

[BW03a] Wolf-Tilo Balke and Matthias Wagner. Cooperative discovery for user-centered web service provisioning. In *ICWS*, pages 191–197, 2003.

[BW03b] Wolf-Tilo Balke and Matthias Wagner. Towards personalized selection of web services. In *WWW (Alternate Paper Tracks)*, 2003.

[BW04] Wolf-Tilo Balke and Matthias Wagner. Through different eyes: assessing multiple conceptual views for querying web services. In *WWW (Alternate Track Papers & Posters)*, pages 196–205, 2004.

[CBGS06] A. Caballero, J. A. Botía, and A. F. Gómez-Skarmeta. A new model for trust and reputation management with an ontology based approach for similarity between tasks. In *MATES 2006: 4th German Conf. on Multiagent System Technologies, 19-20 Sep 2006, Erfurt. Germany*, pages 172–183, 2006.

[CBGS07] A. Caballero, J. A. Botía, and A. F. Gómez-Skarmeta. On the behaviour of the trsim model for trust and reputation. In *5th German Conf. on Multiagent System Technologies, Leipzig*, pages 182–193, 2007.

[Cha05] Joshua Chang. Online shopping: Advantages over the offline alternative. http://www.arraydev.com/commerce/JIBC/0311-07.htm, 2005.

[Che08] Li Chen. *User Decision Improvement and Trust Building in Product Recommender Systems*. PhD thesis, EPFL, Lausanne, Switzerland, Ecole Polytechnique Federale de Lausanne, August 2008.

[CK04] Y. H. Cho and J. K. Kim. Application of web usage mining and product taxonomy to collaborative recommendations in e-commerce. *Expert Systems with Applications*, 26(2):233–246, feb 2004.

[CKP00] Urszula Chajewska, Daphne Koller, and Ronald Parr. Making rational decisions using adaptive utility elicitation. In *AAAI/IAAI*, pages 363–369, 2000.

[Cle91] Robert T. Clemen. *Making Hard Decisions: An Introduction to Decision Analysis*. P.W.S.-Kent Publishing Co.,U.S., 1991.

[CMD02] J. Carbo, J. Molina, and J. Davila. Comparing predictions of sporas vs. a fuzzy reputation agent system. In *Third International Conference on Fuzzy Sets and Fuzzy Systems, Interlaken*, pages 147–153, 2002.

[CNS$^+$04] Simona Colucci, Tommaso Di Noia, Eugenio Di Sciascio, Marina Mongiello, and Francesco M. Donini. Concept abduction and contraction for semantic-based discovery of matches and negotiation spaces in an e-marketplace. In *ICEC '04: Proceedings of the 6th international conference on Electronic commerce*, pages 41–50, New York, NY, USA, 2004. ACM.

[CNS$^+$06] Simona Colucci, Tommaso Di Noia, Eugenio Di Sciascio, Francesco M. Donini, Azzurra Ragone, and Raffaele Rizzi. A semantic-based fully visual application for matchmaking and query refinement in b2c e-marketplaces. In *ICEC*, pages 174–184, 2006.

[Com08] The Nielsen Company. Trends in online shopping a global nielsen consumer report, February 2008.

[Com10] The Nielsen Company. Global trends in online shopping a nielsen global consumer report, June 2010.

[CP02] Giuseppe Carenini and David Poole. Constructed preferences and value-focused thinking: Implications for ai research on preference elicitation. In *AAAI'02 Workshop on Preferences in AI and CP: Symbolic Approaches, Edmonton, Canada*, 2002.

[CP05] Bernard C.K. Choi and Anita W.P. Pak. A catalog of biases in questionnaires. *Prev. Chronic Dis.*, 2(1), 2005.

[CP06] Li Chen and Pearl Pu. Evaluating critiquing-based recommender agents. In *AAAI*. AAAI Press, 2006.

[CP07a] Li Chen and Pearl Pu. The evaluation of a hybrid critiquing system with preference-based recommendations organization. In *RecSys*, pages 169–172, 2007.

[CP07b] Li Chen and Pearl Pu. Hybrid critiquing-based recommender systems. In *IUI*, pages 22–31, 2007.

[CP07c] Li Chen and Pearl Pu. Preference-based organization interfaces: Aiding user critiques in recommender systems. In *User Modeling*, pages 77–86, 2007.

[CS06] Jorge Cardoso and Amit Sheth, editors. *Semantic Web Services, Processes and Applications*. Springer-Verlag, Heidelberg, 2006.

[Daw79] Robyn M. Dawes. The robust beauty of improper linear models in decision making. *American Psychologist*, 34(7):571–582, 1979.

[Del02] Chrysanthos Dellarocas. Goodwill hunting: An economically efficient online feedback mechanism for environments with variable product quality. In *AMEC*, pages 238–252, 2002.

[DF02] Marek J. Druzdzel and Roger R. Flynn. *Encyclopedia of Library and Information Science*, chapter Decision Support Systems. Marcel Dekker, Inc., New York, 2002.

[DZN$^+$06] Honghua K. Dai, Lingzhi Zhao, Zaiqing Nie, Ji R. Wen, Lee Wang, and Ying Li. Detecting online commercial intention (oci). In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 829–837, New York, NY, USA, 2006. ACM.

[EC01] B. Esfandiari and S. Chandrasekharan. On how agents make friends: mechanisms for trust acquisition. In *Proceedings of the Fourth Workshop on Deception, Fraud and Trust in Agent Societies 2001*, pages 27–34, 2001.

[EDM$^+$05] Daniel Elenius, Grit Denker, David Martin, Fred Gilham, John Khouri, Shahin Sadaati, and Rukman Senanayake. The owl-s editor - a development tool for semantic web services. In *ESWC*, pages 78–92, 2005.

[eu206] Software, services and complexity research in the ist programme framework programme vi (2002-2006) - an overview, 2006.

[FA99] Hershey H. Friedman and Taiwoo Amoo. Rating the rating scales. *Journal of Marketing Management*, 9(3):114–123, 1999.

[FB08] A. Felfernig and R. Burke. Constraint-based recommender systems: technologies and research issues. In *ICEC '08: Proceedings of the 10th international conference on Electronic commerce*, pages 1–10, New York, NY, USA, 2008. ACM.

[FFST11] Dieter Fensel, Federico Michele Facca, Elena Paslaru Bontas Simperl, and Ioan Toma. *Semantic Web Services*. Springer, 2011.

[FGE05] J. Figueira, S. Greco, and M. Ehrgott. *Multiple Criteria Decision Analysis: State of the Art Surveys*. Springer Verlag, Boston, Dordrecht, London, 2005.

[Fie00] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, 2000.

[FK10] Birgitta König-Ries Friederike Klan. Supporting consumers in providing meaningful multi-criteria judgments. In *In Proceedings of the International Workshop on the Practical Use of Recommender Systems, Algorithms and Technologies (PRSAT 2010) in conjunction with RecSys2010, Barcelona, Spain*, 2010.

[FLP$^+$07] Dieter Fensel, Holger Lausen, Axel Polleres, Jos de Bruijn, Michael Stollberg, Dumitru Roman, and John Domingue. *Enabling Semantic Web Services: The Web Service Modeling Ontology*. Springer, 2007.

[FR01] E. J. Friedman and P. Resnick. The social cost of cheap pseudonyms. *Journal of Economics & Management Strategy*, 10(2):173–199, 2001.

[Fra96] Robert B. Frary. Hints for designing effective questionnaires. *Practical Assessment, Research and Evaluation*, 5(3), 1996.

[GA07] YoungOk Kwon Gediminas Adomavicius. New recommendation techniques for multi-criteria rating systems. *IEEE Intelligent Systems*, 22(3), 2007.

[Gam88] Diego Gambetta. Can we trust trust? In *Trust: Making and Breaking Cooperative Relations*, pages 213–237. Basil Blackwell, 1988.

[GDFB06] N. Gujral, D. DeAngelis, K. Fullam, and K. S. Barber. Modeling multi-dimensional trust. In *Proceedings of The Workshop on Trust in Agent Societies at The 5th International Conference on Autonomous Agents and Multi-agent Systems*, pages 35–41, 2006.

[GMP06] Stephan Grimm, Boris Motik, and Chris Preist. Matching semantic service descriptions with local closedworld reasoning. In *The Semantic Web: Research and Applications, 3rd European Semantic Web Conference, ESWC 2006, Budva*, pages 575–589. Springer, 2006.

[Gri05] Nathan Griffiths. Task delegation using experience-based multi-dimensional trust. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 489–496, New York, NY, USA, 2005. ACM.

[GS00] Tyrone Grandison and Morris Sloman. A survey of trust in internet applications. *IEEE Communications Surveys and Tutorials*, 3(4), 2000.

[HCG01] David Hawking, Nick Craswell, and Kathleen Griffiths. Which search engine is best at finding online services? In *WWW Posters*, 2001.

[Hol08] Clyde W. Holsapple. Dss architecture and types. In Frada Burstein and Clyde W. Holsapple, editors, *Handbook on Decision Support Systems 1*, pages 163–189. Springer Berlin Heidelberg, 2008.

[How88] Ronald A. Howard. Decision analysis: Practice and promise. *Management Science*, 34(6):679–695, 1988.

[Hub07] W. Douglas Hubbard. *How to measure anything finding the value of 'intangibles' in business*. John Wiley & Sons, Hoboken, N.J., 2007.

[Jac01] Paul Jaccard. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bulletin de la Société Vaudoise des Sciences Naturelles*, 37:547–579, 1901.

[JBXC08] Audun Jøsang, Touhid Bhuiyan, Yue Xu, and CliveCombining Cox. Combining trust and reputation managament for web-based services. In Steven Furnell, Sokratis K. Katsikas, and Antonio Lioy, editors, *TrustBus*, volume 5185, pages 90–99. Springer, 2008.

[JH07] Audun Josang and Jochen Haller. Dirichlet reputation systems. In *ARES '07: Proceedings of the The Second International Conference on Availability, Reliability and Security*, pages 112–119, Washington, DC, USA, 2007. IEEE Computer Society.

[JHP06] Audun Josang, Ross Hayward, and Simon Pope. Trust network analysis with subjective logic. In *ACSC '06: Proceedings of the 29th Australasian Computer Science Conference*, pages 85–94, Darlinghurst, Australia, Australia, 2006. Australian Computer Society, Inc.

[JIB07] Audun Josang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decis. Support Syst.*, 43(2):618–644, 2007.

[JLC08] Audun Josang, Xixi Luo, and Xiaowu Chen. Continuous ratings in discrete bayesian reputation systems. 263:151, 2008.

[Jøs01] Audun Jøsang. A logic for uncertain probabilities. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 9(3):279–311, 2001.

[Jøs02] Audun Jøsang. The beta reputation system. In *In Proceedings of the 15th Bled Electronic Commerce Conference*, 2002.

[Jos08] Audun Josang. Online reputation systems for the health sector. *Electronic Journal of Health Informatics*, 3(1), 2008.

[JvN53] Oskar Morgenstern John von Neumann. *Theory Of Games And Economic Behavior*. Princeton University Press., 1953.

[JWP93] Eric J. Johnson John W. Payne, James R. Bettman. *The adaptive decision maker*. Cambridge University Press, 1993.

[JZ09] Markus Jessenitschnig and Markus Zanker. A generic user modeling component for hybrid recommendation strategies. pages 337–344, jul 2009.

[Kau06] Frank Kaufer. Wsmo-mx: A logic programming based hybrid service matchmaker. In *2006 European Conference on Web Services (ECOWS 06)*, page 161, 2006.

[KBD07] N. Kokash, A. Birukou, and V. D'Andrea. Web service discovery based on past user experience. In *BIS*, volume 4439, pages 95–107. Springer, 2007.

[Kee92] Ralph L. Keeney. *Value-Focused Thinking - A Path to Creative Decisionmaking*. Harvard University Press, Cambridge, Massachusetts, London, England, 1992.

[Kee05] Ralph L. Keeney. *Advances in Decision Analysis: From Foundations to Applications*, chapter Developing Objectives and Attributes, pages 104–128. New York: Cambridge University Press, 2005.

[Ker06] M. Kerrigan. Web service selection mechanisms in the web service execution environment (wsmx). In *21st ACM Symposium on Applied Computing*, pages 1664–1668, Dijon, 2006.

[KFS06] Matthias Klusch, Benedikt Fries, and Katia Sycara. Automated semantic web service discovery with owls-mx. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 915–922, New York, NY, USA, 2006. ACM.

[KKK08] Matthias Klusch, Patrick Kapahnke, and Frank Kaufer. Evaluation of wsml service retrieval with wsmo-mx. In *Proceedings of the 2008 IEEE International Conference on Web Services*, ICWS '08, pages 401–408, Washington, DC, USA, 2008. IEEE Computer Society.

[KKR04] Michael Klein and Birgitta König-Ries. Coupled signature and specification matching for automatic service binding. In *Proc. of the European Conference on Web Services (ECOWS 2004)*, Erfurt, Germany, September 2004.

[KKR08] Friederike Klan and Birgitta König-Ries. A personalized approach to experience-aware service ranking and selection. In *Proceedings of the 2nd International Conference on Scalable Uncertainty Management (SUM 2008), Naples, Italy*, October 2008.

[KKR09] Ulrich Küster and Birgitta König-Ries. Semantic service discovery with diane service descriptions. In Charles Petrie, Tiziana Margaria, Holger Lausen, and Michal Zaremba, editors, *Semantic Web Services Challenge*, volume 8, chapter Semantic Web and Beyond, pages 199–216. Springer US, 2009.

[KKR10a] Friederike Klan and Birgitta König-Ries. Enabling Trust-Aware Semantic Web Service Selection - A Flexible and Personalized Approach. Jenaer Schriften zur Mathematik und Informatik, Math/Inf/02/10, Friedrich-Schiller-University Jena, August 2010.

[KKR10b] Friederike Klan and Birgitta König-Ries. Enabling trust-aware semantic web service selection - a flexible and personalized approach. In *Proceedings of the 12th International Conference on Information Integration and Web-based Applications & Services (iiWAS), Paris, France, November 2010*, 2010.

[KKR11] Friederike Klan and Birgitta König-Ries. A conversational approach to semantic web service selection. In *EC-Web*, pages 1–12, 2011.

[KKRKS07] Ulrich K"uster, Birgitta K"onig-Ries, Michael Klein, and Mirco Stern. Diane - a matchmaking-centered framework for automated service discovery, composition, binding and invocation. In *Proceedings of the 16th International World Wide Web Conference (WWW2007)*, Banff, Alberta, Canada, May 2007.

[KKRM05] M. Klein, B. König-Ries, and M. Müssig. What is needed for semantic service descriptions?: A proposal for suitable language constructs. *Int. J. of Web and Grid Services*, 1(3/4):328–364, 2005.

[Kla06] Friederike Klan. Context-aware service discovery, selection and usage. In *18th GI-Workshop on the Foundations of Databases, Wittenberg, Saxony-Anhalt*, 2006.

[Kle05] Michael Klein. *Automatisierung dienstorientierten Rechnens durch emantische Dienstbeschreibungen*. PhD thesis, Friedrich-Schiller-Universität Jena, 2005.

[KM01] Waikit Koh and Lik Mui. Ceur workshop proceedings. In Alexander Maedche, Steffen Staab, Claire Nedellec, and Eduard H. Hovy, editors, *Workshop on Ontology Learning*, volume 38. CEUR-WS.org, 2001.

[KMSF09] Mick Kerrigan, Adrian Mocan, Elena Simperl, and Dieter Fensel. Modeling semantic web services with the web service modeling toolkit. *Journal of Network and Systems Management*, 17(3):326, 2009.

[Kni21] Frank H. Knight. *Risk, Uncertainty and Profit*. University of Chicago Press, 1921.

[KR93] R. L. Keeney and H. Raiffa. *Decisions with Multiple Objectives: Preferences and Value Trade-offs*. Cambridge University Press, Cambridge, 1993.

[KTR05] Michael Kinateder, Ralf Terdic, and Kurt Rothermel. Strong pseudonymous communication for peer-to-peer reputation systems. In *Proceedings of the 2005 ACM Symposium on Applied Computing (SAC '05)*, pages 1570–1576, New York, NY, USA, 2005. ACM Press.

[Küs10] Ulrich Küster. *An Evaluation Methodology and Framework for Semantic Web Services Technologies*. PhD thesis, Friedrich-Schiller-University Jena, Jena, Germany, June 2010.

[KvW07] R. Keeney and D. von Winterfeldt. *Advances in Decision Analysis: From Foundations to Applications*, chapter Practical value models, pages 232–252. New York: Cambridge University Press, 2007.

[Lew82] C. H. Lewis. Using the "thinking aloud" method in cognitive interface design. Technical Report RC-9265, IBM, 1882.

[lik32] A technique for the measurement of attitudes. *Archives of Psychology*, 140:1–55, 1932.

[LK10] Tsoukiàs A. Lakiotaki K., Matsatsinis N. Multi-criteria user modeling in recommender systems. *IEEE Intelligent Systems*, 2010.

[LMRD09] Philipp Leitner, Anton Michlmayr, Florian Rosenberg, and Schahram Dustdar. Selecting web services based on past user experiences. In *APSCC*, pages 205–212, 2009.

[LPN$^+$07] Tomás Pariente Lobo, Alejandro Lopez Perez, Joachim Nern, Gennady Agre, Zlatina Marinova, Tatiana Atanasova, Andràs Micsik, James Scicluna, Janne Saarela, and Elpida Tzafestas. Infrawebs integrated framework user guide, version 2.0, dated 21/02/2007, 2007.

[LR05] Fabiana Lorenzi and Francesco Ricci. Case-based recommender systems: A unifying view. In Bamshad Mobasher and Sarabjot S. Anand, editors, *Lecture Notes in Computer Science, ITWP*, volume 3169, pages 89–113. Springer, 2005.

[LS09] Fabián P. P. Lousam and Eduardo Sánchez. View-based recommender systems. In *Proceedings of the third ACM conference on Recommender systems*, RecSys '09, pages 389–392, New York, NY, USA, 2009. ACM.

[LWG08] Qiudan Li, Chunheng Wang, and Guanggang Geng. Improving personalized services in mobile commerce by a novel multicriteria rating approach. In *Proceeding of the 17th international conference on World Wide Web*, WWW '08, pages 1235–1236, New York, NY, USA, 2008. ACM.

[Mar78] J. G. March. Bounded rationality, ambiguity, and the engineering of choice. *The Bell Journal of Economics*, 9(2):587–608, 1978.

[Mar94] Stephen Paul Marsh. *Formalising Trust as a Computational Concept*. PhD thesis, University of Stirling, April 1994.

[MBBW07] Bamshad Mobasher, Robin Burke, Runa Bhaumik, and Chad Williams. Toward trustworthy recommender systems. *ACM Transactions on Internet Technology*, 7(4):23, 2007.

[MC07] Nikos Manouselis and Constantina Costopoulou. Experimental analysis of design choices in multiattribute utility collaborative filtering. *IJPRAI*, 21(2):311–331, 2007.

[Min98] Jack Minker. An overview of cooperative answering in databases. In *FQAS '98: Proceedings of the Third International Conference on Flexible Query Answering Systems*, pages 282–285, London, UK, 1998. Springer-Verlag.

[MMA$^+$01] L. Mui, M. Mohtashemi, C. Ang, P. Szolovits, and A. Halberstadt. Ratings in distributed systems: A bayesian approach, 2001.

[MMS05] Glenn Mahoney, Wendy J. Myrvold, and Gholamali C. Shoja. Generic reliability trust model. In *PST*, 2005.

[MP05] U. S. Manikrao and T. V. Prabhakar. Dynamic selection of web services with recommendation system. In *Intl. Conf. on Next Generation Web Services Practices*, pages 117–121, Washington, DC, 2005. IEEE Computer Society.

[MRMS04] K. McCarthy, J. Reilly, L. McGinty, and B. Smyth. Thinking positively - explanatory feedback for conversational recommender systems. In P. Cunningham and D. McSherry, editors, *European Conference on Case-Based Reasoning (ECCBR-04) Explanation Workshop*, pages 115–124, 2004. Madrid, Spain.

[MS02] E. M. Maximilien and M. P. Singh. Conceptual model of web service reputation. *SIGMOD Rec.*, 31(4):36–41, 2002.

[MS05] E. Michael Maximilien and Munindar P. Singh. Agent-based trust model involving multiple qualities. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 519–526, New York, NY, USA, 2005. ACM.

[Mui02] Lik Mui. *Computational Models of Trust and Reputation: Agents, Evolutionary Games, and Social Networks*. PhD thesis, MIT, 2002.

[nes07] Nessi strategic research agenda: Vol. 2 strategy to build nessi, March 2007.

[nes09] Nessi strategic research agenda: Nessi research priorities for fp7, May 2009.

[NHC08] Syavash Nobarany, Mona Haraty, and Dan Cosley. Geputtis: General purpose transitive trust inference system for social networks. In *Proceedings of the AAAI Spring Symposium on Social Information Processing (AAAI-SIP-08)*, Menlo Park, California, 2008. AAAI.

[Nie93] J. Nielsen. *Usability Engineering*. Academic Press, 1993.

[NLL$^+$06] Olaf Noppens, Marko Luther, Thorsten Liebig, Matthias Wagner, and Massimo Paolucci. Ontology-supported preference handling for mobile music selection. In *Proceedings of the Multidisciplinary Workshop on Advances in Preference Handling*, Riva del Garda, Italy, august 2006.

[NLS07] O. Noppens, T. Liebig, and P. Schmidt. Mobixpl - a svg-based mobile user interface for semantic service discovery. In *Proceedings of the 5th International Conference on Scalable Vector Graphics (SVG Open 2007)*, 2007.

[NM90] Jakob Nielsen and Rolf Molich. Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Empowering people*, CHI '90, pages 249–256, New York, NY, USA, 1990. ACM.

[OAS06] OASIS. Reference model for service oriented architecture 1.0. OASIS Standard, October 2006.

[PB07] Michael J. Pazzani and Daniel Billsus. Content-based recommendation systems. In Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, editors, *The Adaptive Web: Methods and Strategies of Web Personalization*, volume 4321, chapter Content-Based Recommendation Systems, pages 325–341. Springer, Berlin, Heidelberg, 2007.

[PBJ92] J. W. Payne, J. R. Bettman, and E. J. Johnson. Behavioral decision research - a constructive processing perspective. *Annual Review of Psychology*, 43:87–131, 1992.

[PBS99] John W. Payne, James R. Bettman, and David A. Schkade. Measuring constructed preferences: Towards a building code. *Journal of Risk and Uncertainty*, 19(1-3):243–70, December 1999.

[PBTL99] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Discovering frequent closed itemsets for association rules. In *ICDT*, pages 398–416, 1999.

[PC06] Pearl Pu and Li Chen. Trust building with explanation interfaces. In *IUI*, pages 93–100, 2006.

[PC07] P PU and L CHEN. Trust-inspiring explanation interfaces for recommender systems. *Knowledge-Based Systems*, 20(6):542, 2007.

[Phi84] Lawrence D. Phillips. A theory of requisite decision models. *Acta Psychologica*, 56:29–48, 1984.

[RE03] M. Rodriguez and M. Egenhofer. Determining semantic similarity among entity classes from different ontologies. *IEEE Trans. on Knowledge and Data Eng.*, 15(2):442–456, 2003.

[Rei05] James Reilly. Incremental critiquing. *Knowledge-Based Systems*, 18(4-5):143, 2005.

[Res10] TechTarget/Forrester Research. State of soa survey for 2010, 2010.

[Rie09] Sebastian Ries. Extending bayesian trust models regarding context-dependence and user friendly representation. In *SAC '09: Proceedings of the 2009 ACM symposium on Applied Computing*, pages 1294–1301, New York, NY, USA, 2009. ACM.

[RIS$^+$94] P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. pages 175–186, Chapel Hill, North Carolina, 1994. ACM.

[RKM$^+$01] Naren Ramakrishnan, Benjamin J. Keller, Batul J. Mirza, Ananth Y. Grama, and George Karypis. Privacy risks in recommender systems. *IEEE Internet Computing*, 5(6):54–62, nov 2001.

[RMMS04] James Reilly, Kevin McCarthy, Lorraine McGinty, and Barry Smyth. Dynamic critiquing. In *ECCBR*, pages 763–777, 2004.

[RNSS08] Michele Ruta, Tommaso Di Noia, Eugenio Di Sciascio, and Floriano Scioscia. A semantic-based fully visual application for context-aware matchmaking and request refinement in ubiquitous computing. In *ICCSA (2)*, pages 259–274, 2008.

[RRRJ07a] Steven Reece, Stephen Roberts, Alex Rogers, and Nicholas R. Jennings. A multi-dimensional trust model for heterogeneous contract observations. In *AAAI*, pages 128–135, 2007.

[RRRJ07b] Steven Reece, Alex Rogers, Stephen Roberts, and Nicholas R. Jennings. Rumours and reputation: evaluating multi-dimensional trust within a decentralised reputation system. In *AAMAS '07: Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, pages 1–8, New York, NY, USA, 2007. ACM.

[RSMM05] James Reilly, Barry Smyth, Lorraine McGinty, and Kevin McCarthy. Critiquing with confidence. In Héctor Muñoz-Avila and Francesco Ricci, editors, *Lecture Note IN Computer Science, ICCBR*, volume 3620, pages 436–450. Springer, 2005.

[RZM$^+$07] James Reilly, Jiyong Zhang, Lorraine McGinty, Pearl Pu, and Barry Smyth. Evaluating compound critiquing recommenders: a real-user study. In *EC '07: Proceedings of the 8th ACM conference on Electronic commerce*, pages 114–123, New York, NY, USA, 2007. ACM.

[Saa08] Thomas L. Saaty. Relative measurement and its generalization in decision making: Why pairwise comparisons are central in mathematics for the measurement of intangible factors - the analytic hierarchy process. *RACSAM*, 102(2):251–318, 2008.

[Sab05] Jordi Sabater. Review on computational trust and reputation models. *Artificial Intelligence Review*, 24(1):33, 2005.

[SAM04] J. Scicluna, C. Abela, and M. Montebello. Visual modelling of owl-s services. In *IADIS International Conference WWW/Internet*, 2004.

[sec06] Secse deliverable a2.d5 secse requirements process v2.0, 2006.

[SFHS07] J. Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. *Collaborative Filtering Recommender Systems*, volume 4321 of *Lecture Notes in Computer Science*, pages 291–324. Springer, Berlin, Heidelberg, 2007.

[Sha48] C. E. Shannon. A mathematical theory of communication. *Bell system technical journal*, 27, 1948.

[Shi01] Hideo Shimazu. Expert clerk: Navigating shoppers' buying process with the combination of asking and proposing. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI'01)*, pages 1443–1450, August 2001.

[Shi02] Hideo Shimazu. Expertclerk: A conversational case-based reasoning tool fordeveloping salesclerk agents in e-commerce webshops. *Artif. Intell. Rev.*, 18(3-4):223–244, 2002.

[Shn98] Ben Shneiderman. *Designing the User Interface - Strategies for Effective Human-Computer Interaction.* Addison-Wesley Longman, Reading, MA, 3rd edition, 1998.

[Sho08] D. Sholler. 2008 soa user survey: Adoption trends and characteristics, 2008. No. G00161125.

[SJ04] Jean-Marc Seigneur and Christian D. Jensen. Trading privacy for trust. pages 93–107. 2004.

[SJ07] Christoph Schroth and Till Janner. Web 2.0 and soa: Converging concepts enabling the internet of services. *IT Professional*, 9(3):36–41, may 2007.

[SKDC06] Nachiketa Sahoo, Ramayya Krishnan, George Duncan, and James P. Callan. Collaborative filtering with multi-component rating for recommender systems. In *16th WI Workshop on Information Technologies and Systems*, Milwaukee, 2006.

[SLH06] N. Shadbolt, Tim B. Lee, and W. Hall. The semantic web revisited. *IEEE Intelligent Systems*, 21(3):96–101, 2006.

[Slo72] P. Slovic. *Limitations of the Mind of Man: Implications for decision making in the nuclear age*. Los Alamos Scientific Laboratory, 1972.

[Slo95] P. Slovic. The construction of preference. *American Psychologist*, 50(5):364–371, May 1995.

[SMB10] Ahu Sieg, Bamshad Mobasher, and Robin Burke. Improving the effectiveness of collaborative recommendation with ontology-based user profiles. In *Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems*, HetRec '10, pages 39–46, New York, NY, USA, 2010. ACM.

[Smy07] Barry Smyth. Case-based recommendation. In *The Adaptive Web*, pages 342–376, 2007.

[soa08] Soa4all deliverable d2.2.1 service consumption platform design, 2008.

[SPg$^+$07] M. Sensoy, F. C. Pembe, H. Zirtilou glu, P. Yolum, and A. Bener. Experience-based service provider selection in agent-mediated e-commerce. *Eng. Appl. Artif. Intell.*, 20(3):325–335, 2007.

[SS01] J. Sabater and C. Sierra. Regret: Reputation in gregarious societies. In *AGENTS '01: Proceedings of the 5th Intl. Conf. on Autonomous agents*, pages 194–195, New York, NY, USA, 2001.

[SS02] Jordi Sabater and Carles Sierra. Reputation and social network analysis in multi-agent systems. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 475–482, New York, NY, USA, 2002. ACM.

[SSN01] Hideo Shimazu, Akihiro Shibata, and Katsumi Nihei. Expertguide: A conversational case-based reasoning tool for developing mentors in knowledge spaces. *Appl. Intell.*, 14(1):33–48, 2001.

[Tin07] Nava Tintarev. A survey of explanations in recommender systems. In *2007 IEEE 23rd International Conference on Data Engineering Workshop*, page 801, 2007.

[TLU06] Santtu Toivonen, Gabriele Lenzini, and Ilkka Uusitalo. Context-aware trust evaluation functions for dynamic reconfigurable systems. In *MTW*, 2006.

[TP98] Ellen Taylor-Powell. Questionnaire design: Asking questions with purpose. G3658-2 program development and evaluation, University of Wisconsin, 1998.

[Tsa95] Edward Tsang. *Foundations of Constraint Satisfaction*. Academic Press, 1995.

[TT08] Richard Taylor and Chris Tofts. Enabling the web of services. Hpl-2008-14, 2008.

[Tve77] A. Tversky. Features of similarity. *Psychological Review*, 84:327–352, 1977.

[UKA04] Takeaki Uno, Masashi Kiyomi, and Hiroki Arimura. Lcm ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets. In *FIMI*, 2004.

[VFP06] Paolo Viappiani, Boi Faltings, and Pearl Pu. Preference-based search using example-critiquing with suggestions. *Journal of Artificial Intelligence Research (JAIR)*, 27:465–503, 2006.

[VHA05] L. H. Vu, M. Hauswirth, and K. Aberer. Qos-based service selection and ranking with trust and reputation management. In *Intl. Conf. on Cooperative Information Systems*, volume 3760(1), pages 446–483, Agia Napa, 2005.

[VPF07] Paolo Viappiani, Pearl Pu, and Boi Faltings. Conversational recommenders with adaptive suggestions. In *RecSys '07: Proceedings of the 2007 ACM conference on Recommender systems*, pages 89–96, New York, NY, USA, 2007. ACM.

[VPF08] Paolo Viappiani, Pearl Pu, and Boi Faltings. Preference-based search with adaptive recommendations. *AI Communications*, 21(2-3):155–175, April 2008.

[VPHA07] L. H. Vu, F. Porto, M. Hauswirth, and K. Aberer. An extensible and personalized approach to qos-enabled service discovery. In *11th Intl. Database Engineering & Applications Symposium*, Banff, 2007.

[WLH07] H. C. Wang, C. S. Lee, and T. H. Ho. Combining subjective and objective qos factors for personalized web service selection. *Expert Syst. Appl.*, 32(2):571–584, 2007.

[WLN$^+$04] Matthias Wagner, Thorsten Liebig, Olaf Noppens, Steffen Balzer, and Wolfgang Kellerer. Towards semantic-based service discovery on tiny mobile devices. In *Proceedings of the Workshop on Semantic Web Technology for Mobile and Ubiquitous Applications*, pages 90–101, Hiroshima, Japan, Novemer 2004.

[WV03] Yao Wang and Julita Vassileva. Bayesian network-based trust model. In *WI '03: Proceedings of the IEEE/WIC International Conference on Web Intelligence*, Washington, DC, USA, 2003. IEEE Computer Society.

[YS02] Bin Yu and Munindar P. Singh. Distributed reputation management for electronic commerce. *Computational Intelligence*, 18(4):535–549, 2002.

[Yu07] Liyang Yu. *Introduction to the Semantic Web and Semantic Web Services*. Chapman & Hall/CRC, 1 edition, 2007.

[Zac99] G. Zacharia. Collaborative reputation mechanisms for online communities. Master's thesis, Masschusetts Institute of Technology, 1999.

[Zan09] Markus Zanker. Case-studies on exploiting explicit customer requirements in recommender systems. *User Modeling and User-Adapted Interaction*, 19(1-2):133, 2009.

[ZM08] Konstantinos Zachos and Neil Maiden. Inventing requirements from software: An empirical investigation with web services. In *Proceedings of the 2008 16th IEEE International Requirements Engineering Conference*, pages 145–154, Washington, DC, USA, 2008. IEEE Computer Society.

[ZMHM08] Konstantinos Zachos, Neil A. M. Maiden, and Rhydian Howells-Morris. Discovering web services to improve requirements specifications: Does it help? In *REFSQ*, pages 168–182, 2008.

[ZVB06] Weiliang Zhao, Vijay Varadharajan, and George Bryan. General methodology for analysis and modeling of trust relationships in distributed computing. *JCP*, 1(2):42–53, 2006.

[ZZMJ06] Konstantinos Zachos, Xiaohong Zhu, Neil Maiden, and Sara Jones. Seamlessly integrating service discovery into uml requirements processes. In *SOSE '06: Proceedings of the 2006 international workshop on Service-oriented software engineering*, pages 60–66, New York, NY, USA, 2006. ACM.

Appendix

A
Ontology for the Computer Items Domain

This appendix provides the ontology for the computer items domain (in f-dsd notation [Kle05]), which we created for the evaluation of the performance prediction procedure (Chapter 6 and Section 9.4) as well as for the evaluation of the approach to requirements elicitation and service selection that have been introduced in this thesis (Chapter 5 and Chapter 8). It was used to generate service descriptions from the computer items that have been extracted from a major online seller.

```
{ontology domain.computer}

entityclass Device extends PhysicalEntity at top
[
        size : DimensionMeasure at domain.measure
        manufacturer : Company at domain.economy
]

entityclass Computer extends Device
[
        model : Model
]

entityclass AllPurposeComputer extends Computer
[
        processor : Processor,
        platform : OS,
        memory : Memory
]

entityclass OnePurposeComputer extends Computer []
entityclass StationaryComputer extends AllPurposeComputer
[
        numberOfProcessors : Integer,
        floppyDisc : FDD,
        graphicsCard : GraphicsCard,
        hardDisc : HDD,
        numberOfMemorySlots : Integer,
        systemBus : SystemBus,
        secondaryCache : SecondaryCache
]

entityclass PortableComputer extends AllPurposeComputer
[
        battery : Battery,
        display : Display,
        modem : Modem
]

entityclass DesktopPC extends StationaryComputer
[
        display : Display,
        modem : Modem,
        ethernet : Ethernet
]

entityclass Server extends StationaryComputer []

entityclass Notebook extends PortableComputer
[
        floppyDisc : FDD,
        graphicsCard : GraphicsCard,
        hardDisc : HDD,
        numberOfMemorySlots : Integer,
        numberOfProcessors : Integer,
        secondaryCache : SecondaryCache,
        systemBus : SystemBus
]

entityclass PDA extends PortableComputer []

entityclass DigitalWatch extends OnePurposeComputer []

entityclass EBookReader extends OnePurposeComputer
[
        battery : Battery,
        platform : OS,
        memory : Memory
]
```

```
entityclass ElectronicDictionary extends OnePurposeComputer []

entityclass Organizer extends OnePurposeComputer
[
        battery : Battery,
        platform : OS,
        memory : Memory
]

public entityclass Processor extends Device
[
        clockSpeed : FrequencyMeasure at domain.measure,
        type : String
]

entityclass GraphicsCard extends Device
[
        graphicsCardMemorySize : DataCapacityMeasure at domain.measure
]

entityclass HDD extends Device
[
        hddSize : DataCapacityMeasure at domain.measure
]

entityclass Memory extends Device
[
        memorySize : DataCapacityMeasure at domain.measure,
        memoryType : String
]

entityclass SystemBus extends Device
[
        busSpeed : FrequencyMeasure at domain.measure
]
```

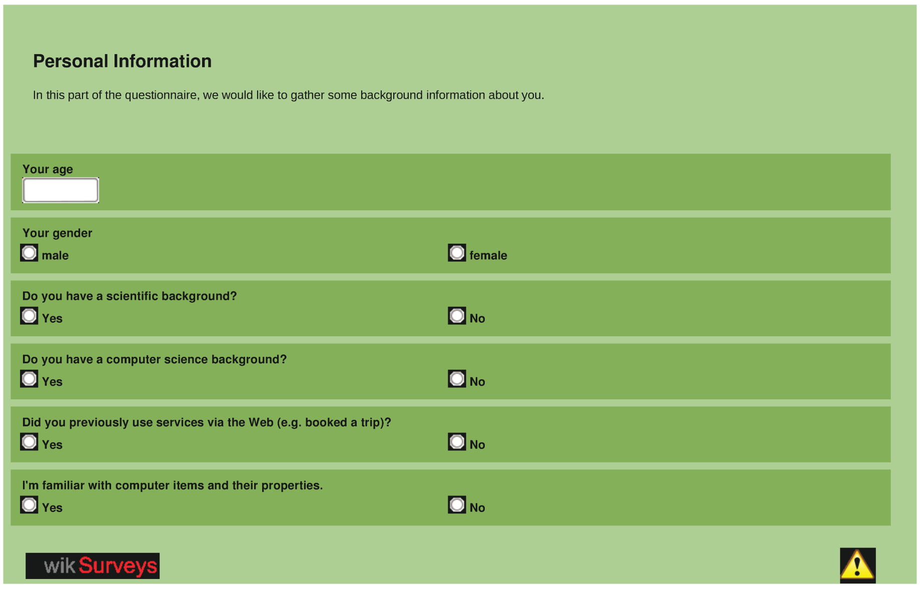
```
entityclass SecondaryCache extends Device
[
        secondaryCacheSize : DataCapacityMeasure at domain.measure
]

entityclass OS extends AbstractEntity at top
[
        producer : Company at domain.economy,
        type : String
]

entityclass Display extends Device
[
        displaySize : LengthMeasure at domain.measure,
        resX : Integer,
        resY : Integer
]

entityclass Modem extends Device at domain.computer
[
        isOnBoard : Boolean
]

entityclass Ethernet extends Device at domain.computer
[
        isOnBoard : Boolean
]

entityclass FDD extends Device at domain.computer
[
        isOnBoard : Boolean
]

entityclass Battery extends PhysicalEntity at top
[
        isIncluded : Boolean
]

entityclass Model extends AbstractEntity at top
[
        modelName : String
]
```

B
Questionnaire for the Evaluation of the Requirements Elicitation and Service Selection Mechanism

This appendix contains the questionnaires that have been used to evaluate the requirements elicitation and service selection mechanism that has been proposed in Chapter 5. These are the Pre-System-Usage Questionnaire (Appendix B.1), the Post-System-Usage Questionnaire (Appendix B.2), the Post-Browsing Questionnaire (Appendix B.3) and the questionnaire for the evaluation of the graphical requirements representation (Appendix B.4). The questionnaires have been created using the free Web-based survey tool Kwik Surveys[1], which enabled online completement of the questionnaires, IP tracking as well as result download and export.

[1]http://kwiksurveys.com

B.1 Pre-System-Usage Questionnaire



Personal Information

In this part of the questionnaire, we would like to gather some background information about you.

Your age

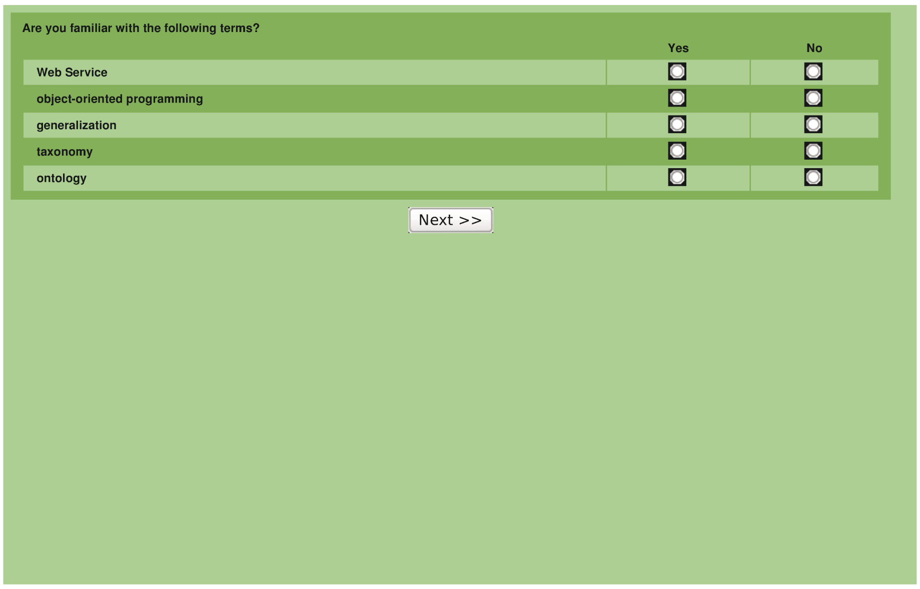
Your gender
male female

Do you have a scientific background?
Yes No

Do you have a computer science background?
Yes No

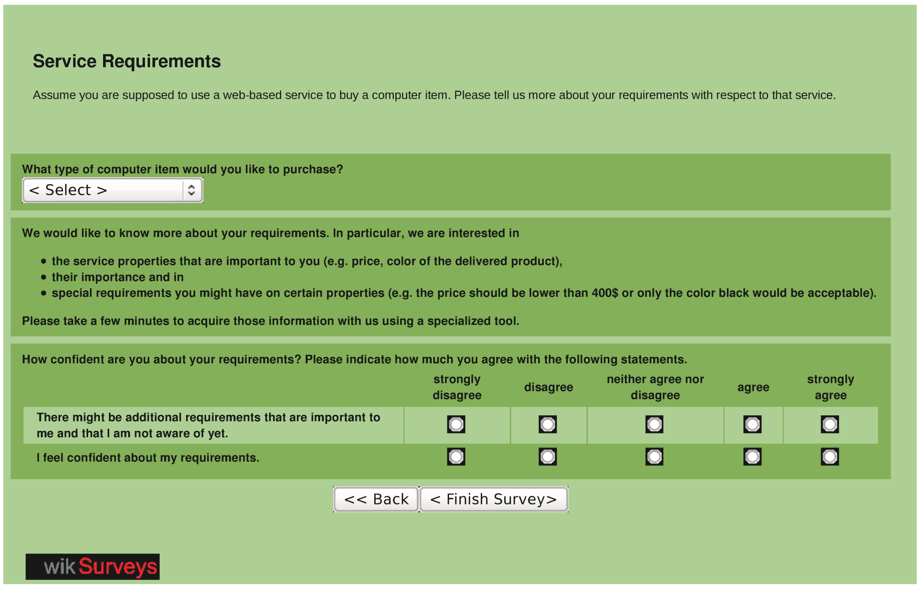
Did you previously use services via the Web (e.g. booked a trip)?
Yes No

I'm familiar with computer items and their properties.
Yes No



Are you familiar with the following terms?

	Yes	No
Web Service		
object-oriented programming		
generalization		
taxonomy		
ontology		



Service Requirements

Assume you are supposed to use a web-based service to buy a computer item. Please tell us more about your requirements with respect to that service.

What type of computer item would you like to purchase?

< Select >

We would like to know more about your requirements. In particular, we are interested in

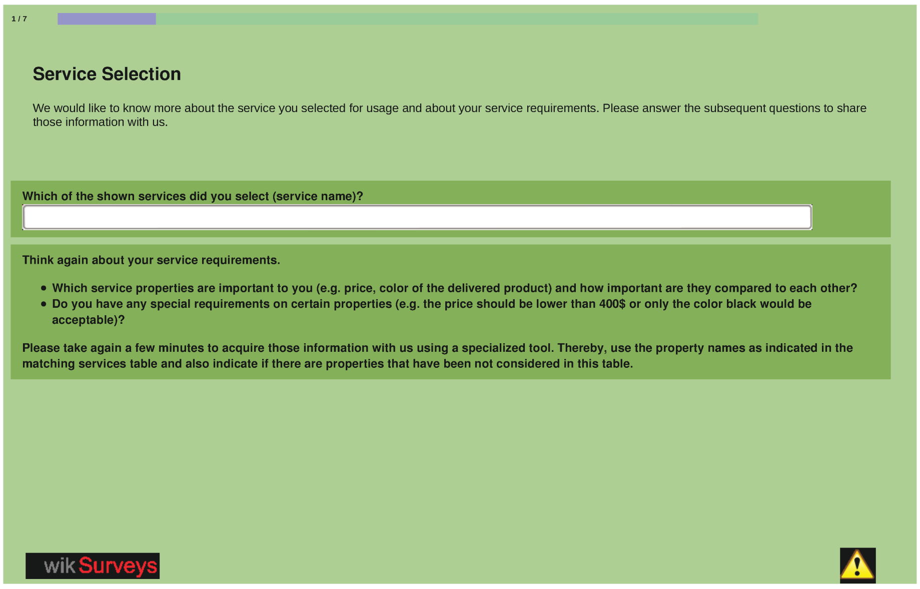
- the service properties that are important to you (e.g. price, color of the delivered product),
- their importance and in
- special requirements you might have on certain properties (e.g. the price should be lower than 400$ or only the color black would be acceptable).

Please take a few minutes to acquire those information with us using a specialized tool.

How confident are you about your requirements? Please indicate how much you agree with the following statements.

	strongly disagree	disagree	neither agree nor disagree	agree	strongly agree
There might be additional requirements that are important to me and that I am not aware of yet.	○	○	○	○	○
I feel confident about my requirements.	○	○	○	○	○

B.2 Post-System-Usage Questionnaire


1 / 7

Service Selection

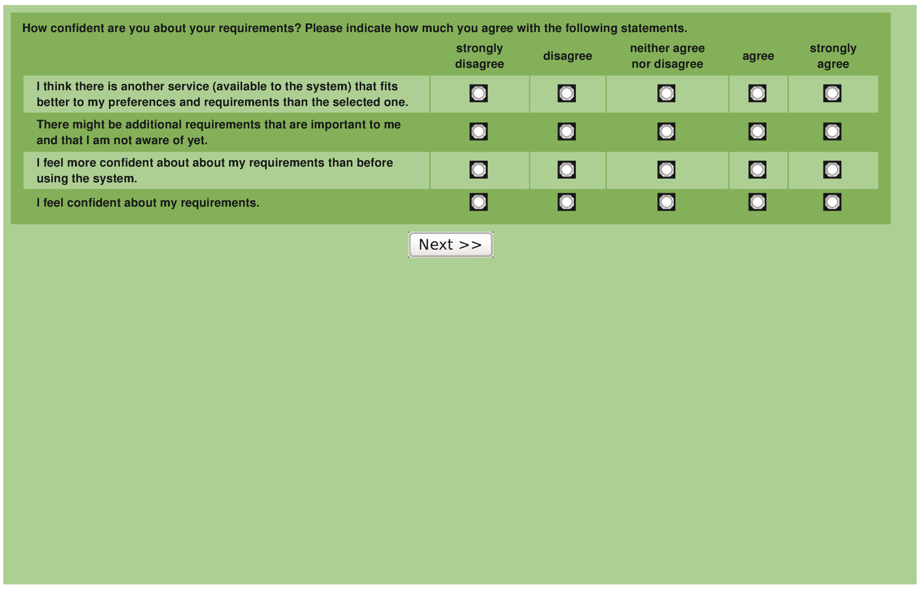
We would like to know more about the service you selected for usage and about your service requirements. Please answer the subsequent questions to share those information with us.

Which of the shown services did you select (service name)?

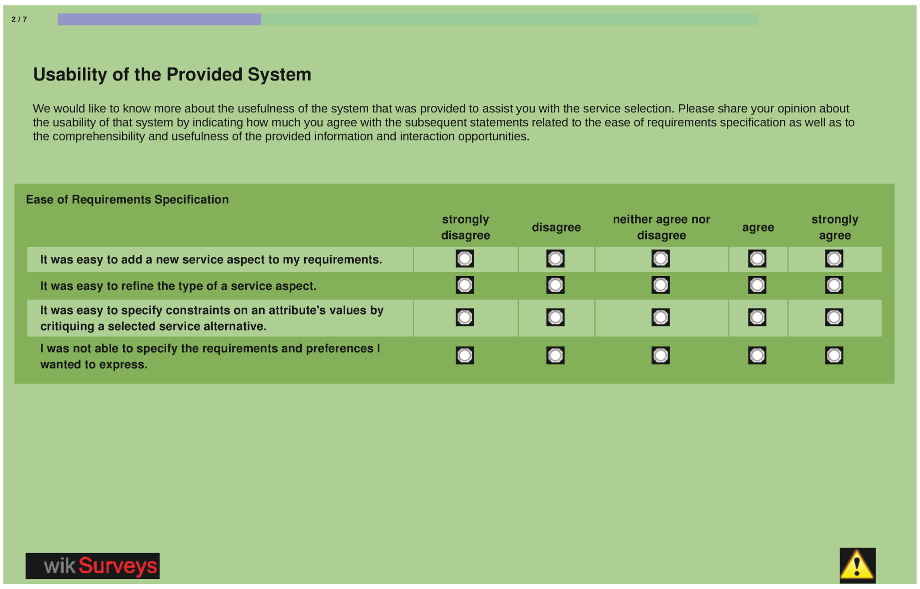
Think again about your service requirements.

- **Which service properties are important to you (e.g. price, color of the delivered product) and how important are they compared to each other?**
- **Do you have any special requirements on certain properties (e.g. the price should be lower than 400$ or only the color black would be acceptable)?**

Please take again a few minutes to acquire those information with us using a specialized tool. Thereby, use the property names as indicated in the matching services table and also indicate if there are properties that have been not considered in this table.


How confident are you about your requirements? Please indicate how much you agree with the following statements.

	strongly disagree	disagree	neither agree nor disagree	agree	strongly agree
I think there is another service (available to the system) that fits better to my preferences and requirements than the selected one.	○	○	○	○	○
There might be additional requirements that are important to me and that I am not aware of yet.	○	○	○	○	○
I feel more confident about about my requirements than before using the system.	○	○	○	○	○
I feel confident about my requirements.	○	○	○	○	○



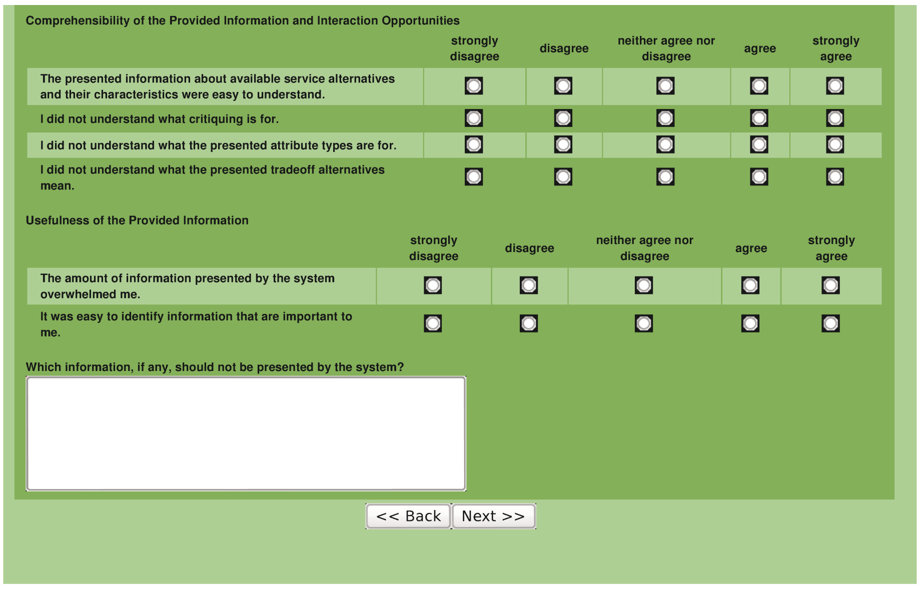
2 / 7

Usability of the Provided System

We would like to know more about the usefulness of the system that was provided to assist you with the service selection. Please share your opinion about the usability of that system by indicating how much you agree with the subsequent statements related to the ease of requirements specification as well as to the comprehensibility and usefulness of the provided information and interaction opportunities.

Ease of Requirements Specification

	strongly disagree	disagree	neither agree nor disagree	agree	strongly agree
It was easy to add a new service aspect to my requirements.	○	○	○	○	○
It was easy to refine the type of a service aspect.	○	○	○	○	○
It was easy to specify constraints on an attribute's values by critiquing a selected service alternative.	○	○	○	○	○
I was not able to specify the requirements and preferences I wanted to express.	○	○	○	○	○



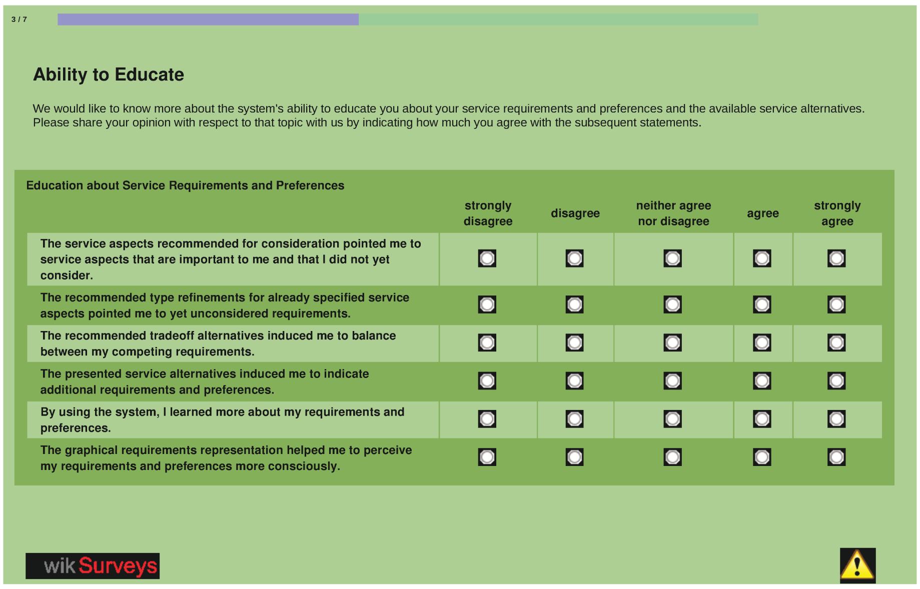
Comprehensibility of the Provided Information and Interaction Opportunities

	strongly disagree	disagree	neither agree nor disagree	agree	strongly agree
The presented information about available service alternatives and their characteristics were easy to understand.	○	○	○	○	○
I did not understand what critiquing is for.	○	○	○	○	○
I did not understand what the presented attribute types are for.	○	○	○	○	○
I did not understand what the presented tradeoff alternatives mean.	○	○	○	○	○

Usefulness of the Provided Information

	strongly disagree	disagree	neither agree nor disagree	agree	strongly agree
The amount of information presented by the system overwhelmed me.	○	○	○	○	○
It was easy to identify information that are important to me.	○	○	○	○	○

Which information, if any, should not be presented by the system?



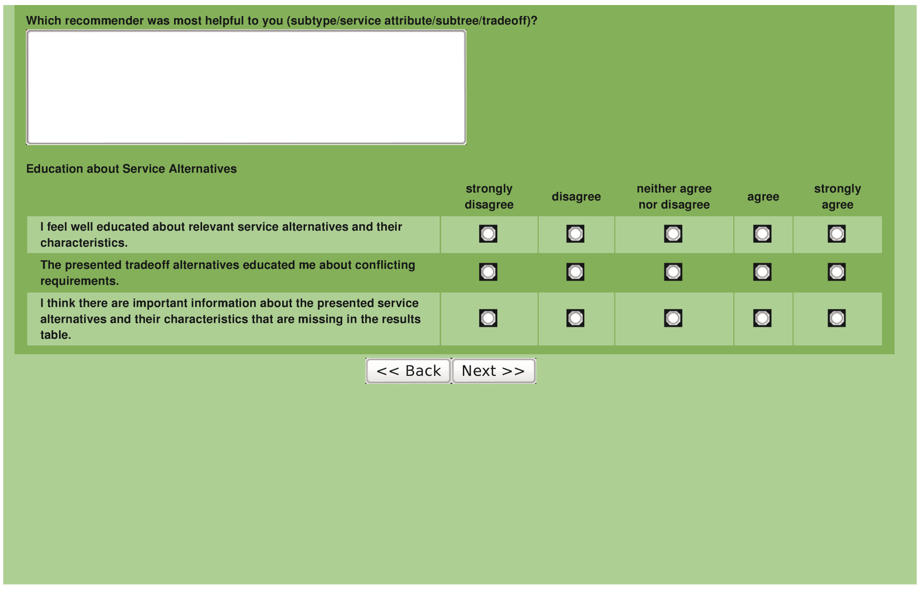
3 / 7

Ability to Educate

We would like to know more about the system's ability to educate you about your service requirements and preferences and the available service alternatives. Please share your opinion with respect to that topic with us by indicating how much you agree with the subsequent statements.

Education about Service Requirements and Preferences

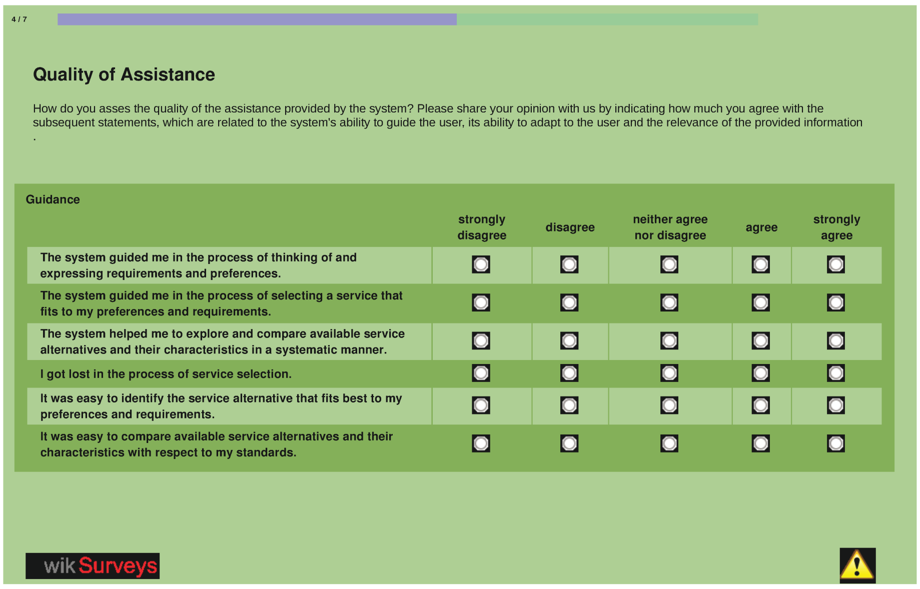
	strongly disagree	disagree	neither agree nor disagree	agree	strongly agree
The service aspects recommended for consideration pointed me to service aspects that are important to me and that I did not yet consider.	○	○	○	○	○
The recommended type refinements for already specified service aspects pointed me to yet unconsidered requirements.	○	○	○	○	○
The recommended tradeoff alternatives induced me to balance between my competing requirements.	○	○	○	○	○
The presented service alternatives induced me to indicate additional requirements and preferences.	○	○	○	○	○
By using the system, I learned more about my requirements and preferences.	○	○	○	○	○
The graphical requirements representation helped me to perceive my requirements and preferences more consciously.	○	○	○	○	○



Which recommender was most helpful to you (subtype/service attribute/subtree/tradeoff)?

Education about Service Alternatives

	strongly disagree	disagree	neither agree nor disagree	agree	strongly agree
I feel well educated about relevant service alternatives and their characteristics.	○	○	○	○	○
The presented tradeoff alternatives educated me about conflicting requirements.	○	○	○	○	○
I think there are important information about the presented service alternatives and their characteristics that are missing in the results table.	○	○	○	○	○



4 / 7

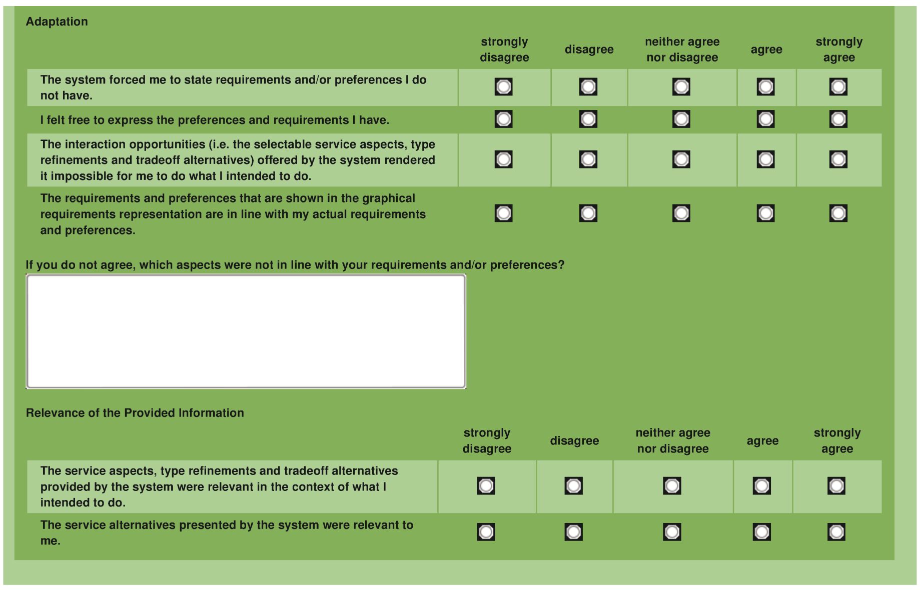
Quality of Assistance

How do you asses the quality of the assistance provided by the system? Please share your opinion with us by indicating how much you agree with the subsequent statements, which are related to the system's ability to guide the user, its ability to adapt to the user and the relevance of the provided information .

Guidance

	strongly disagree	disagree	neither agree nor disagree	agree	strongly agree
The system guided me in the process of thinking of and expressing requirements and preferences.	○	○	○	○	○
The system guided me in the process of selecting a service that fits to my preferences and requirements.	○	○	○	○	○
The system helped me to explore and compare available service alternatives and their characteristics in a systematic manner.	○	○	○	○	○
I got lost in the process of service selection.	○	○	○	○	○
It was easy to identify the service alternative that fits best to my preferences and requirements.	○	○	○	○	○
It was easy to compare available service alternatives and their characteristics with respect to my standards.	○	○	○	○	○

wikSurveys



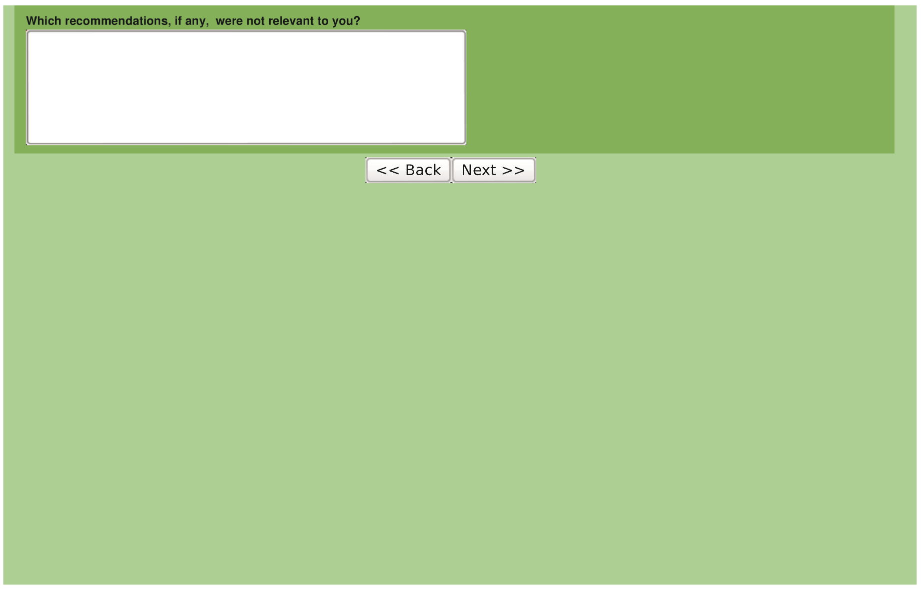
Adaptation

	strongly disagree	disagree	neither agree nor disagree	agree	strongly agree
The system forced me to state requirements and/or preferences I do not have.	○	○	○	○	○
I felt free to express the preferences and requirements I have.	○	○	○	○	○
The interaction opportunities (i.e. the selectable service aspects, type refinements and tradeoff alternatives) offered by the system rendered it impossible for me to do what I intended to do.	○	○	○	○	○
The requirements and preferences that are shown in the graphical requirements representation are in line with my actual requirements and preferences.	○	○	○	○	○

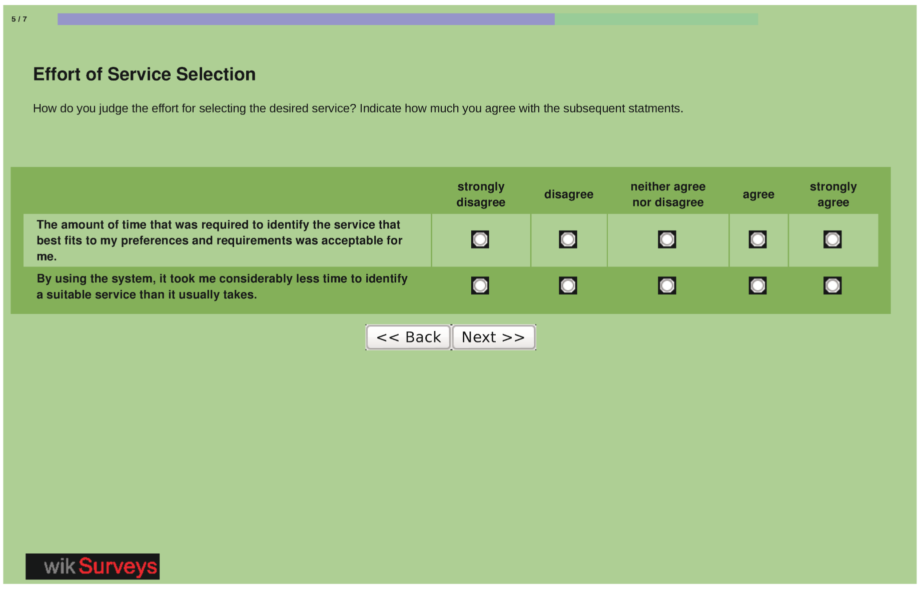
If you do not agree, which aspects were not in line with your requirements and/or preferences?

Relevance of the Provided Information

	strongly disagree	disagree	neither agree nor disagree	agree	strongly agree
The service aspects, type refinements and tradeoff alternatives provided by the system were relevant in the context of what I intended to do.	○	○	○	○	○
The service alternatives presented by the system were relevant to me.	○	○	○	○	○



Which recommendations, if any, were not relevant to you?

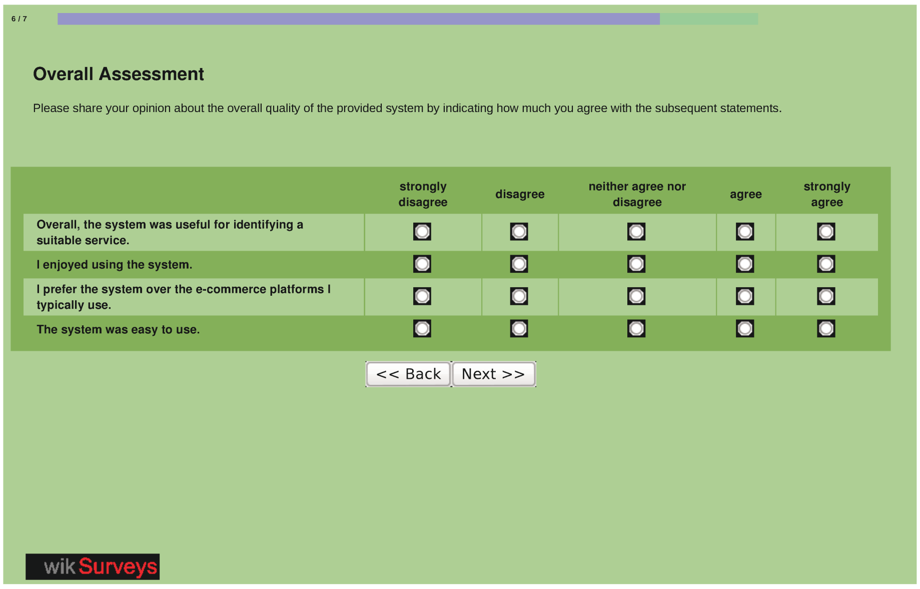


5 / 7

Effort of Service Selection

How do you judge the effort for selecting the desired service? Indicate how much you agree with the subsequent statments.

	strongly disagree	disagree	neither agree nor disagree	agree	strongly agree
The amount of time that was required to identify the service that best fits to my preferences and requirements was acceptable for me.	○	○	○	○	○
By using the system, it took me considerably less time to identify a suitable service than it usually takes.	○	○	○	○	○

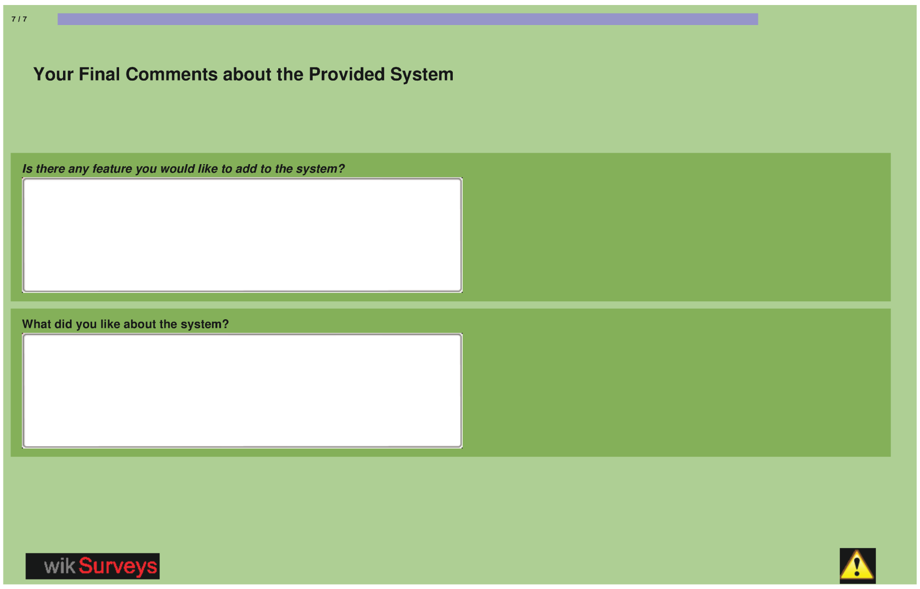


6 / 7

Overall Assessment

Please share your opinion about the overall quality of the provided system by indicating how much you agree with the subsequent statements.

	strongly disagree	disagree	neither agree nor disagree	agree	strongly agree
Overall, the system was useful for identifying a suitable service.	○	○	○	○	○
I enjoyed using the system.	○	○	○	○	○
I prefer the system over the e-commerce platforms I typically use.	○	○	○	○	○
The system was easy to use.	○	○	○	○	○

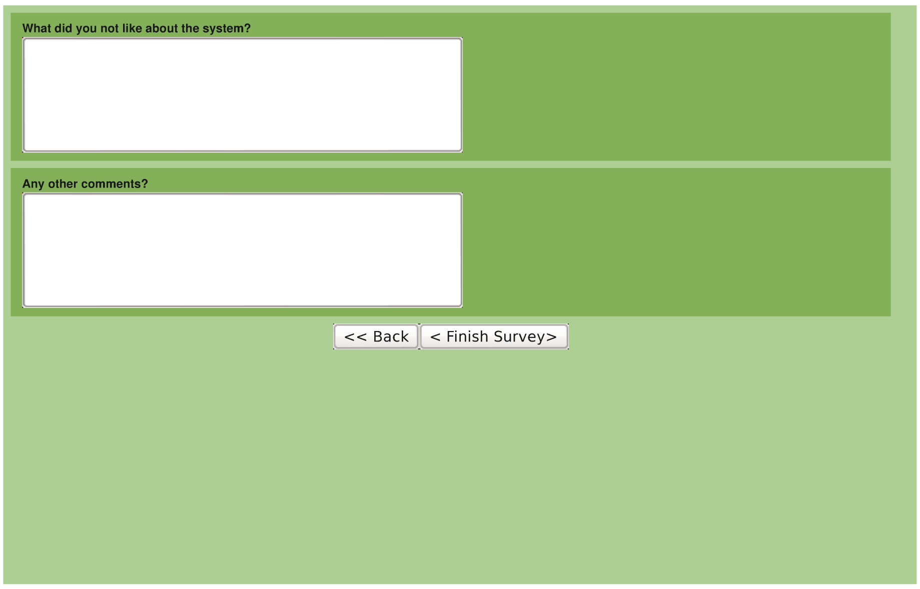


7 / 7

Your Final Comments about the Provided System

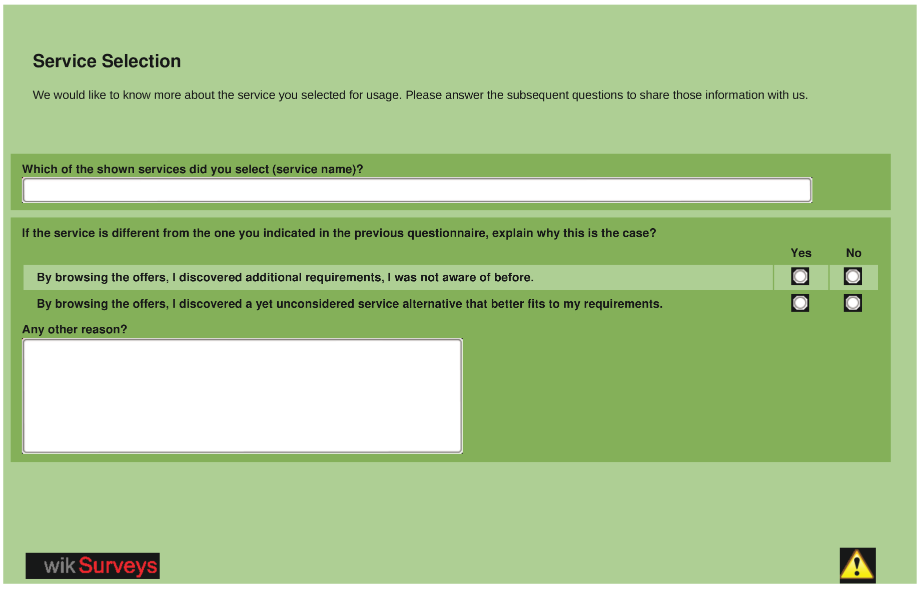
Is there any feature you would like to add to the system?

What did you like about the system?


What did you not like about the system?

Any other comments?

B.3 Post-Browsing Questionnaire



Service Selection

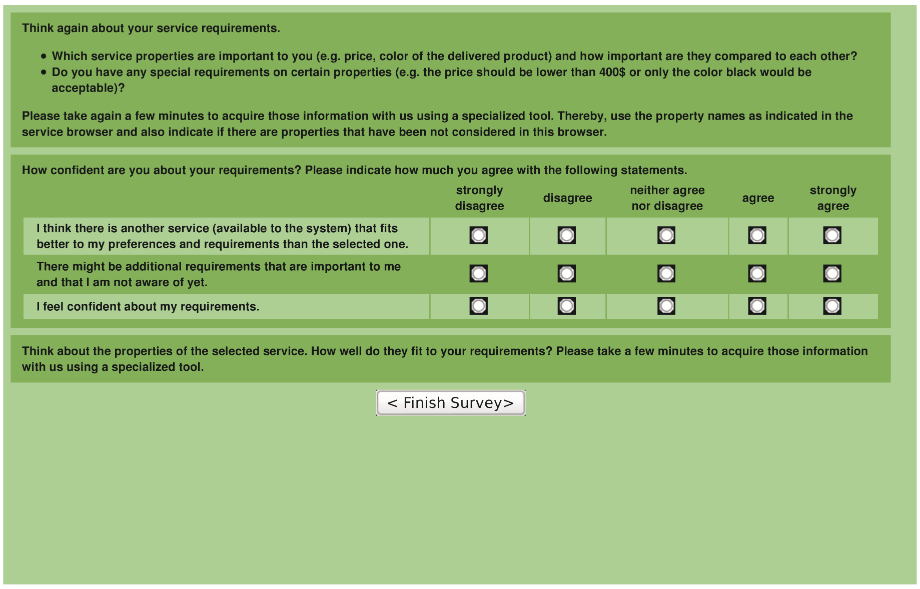
We would like to know more about the service you selected for usage. Please answer the subsequent questions to share those information with us.

Which of the shown services did you select (service name)?

If the service is different from the one you indicated in the previous questionnaire, explain why this is the case?

	Yes	No
By browsing the offers, I discovered additional requirements, I was not aware of before.	○	○
By browsing the offers, I discovered a yet unconsidered service alternative that better fits to my requirements.	○	○

Any other reason?



Think again about your service requirements.

- Which service properties are important to you (e.g. price, color of the delivered product) and how important are they compared to each other?
- Do you have any special requirements on certain properties (e.g. the price should be lower than 400$ or only the color black would be acceptable)?

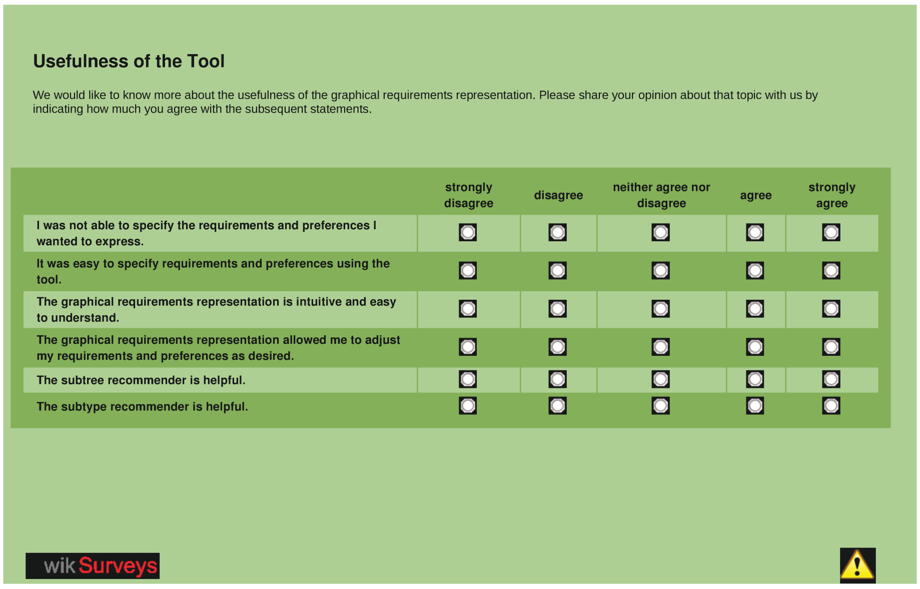
Please take again a few minutes to acquire those information with us using a specialized tool. Thereby, use the property names as indicated in the service browser and also indicate if there are properties that have been not considered in this browser.

How confident are you about your requirements? Please indicate how much you agree with the following statements.

	strongly disagree	disagree	neither agree nor disagree	agree	strongly agree
I think there is another service (available to the system) that fits better to my preferences and requirements than the selected one.	○	○	○	○	○
There might be additional requirements that are important to me and that I am not aware of yet.	○	○	○	○	○
I feel confident about my requirements.	○	○	○	○	○

Think about the properties of the selected service. How well do they fit to your requirements? Please take a few minutes to acquire those information with us using a specialized tool.

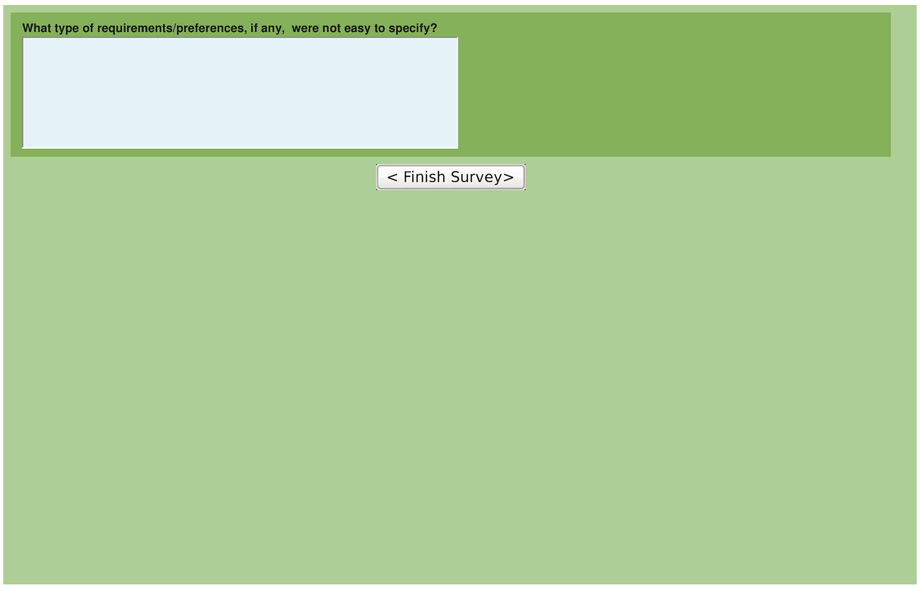
B.4 Questionnaire Requirements Representation



Usefulness of the Tool

We would like to know more about the usefulness of the graphical requirements representation. Please share your opinion about that topic with us by indicating how much you agree with the subsequent statements.

	strongly disagree	disagree	neither agree nor disagree	agree	strongly agree
I was not able to specify the requirements and preferences I wanted to express.	○	○	○	○	○
It was easy to specify requirements and preferences using the tool.	○	○	○	○	○
The graphical requirements representation is intuitive and easy to understand.	○	○	○	○	○
The graphical requirements representation allowed me to adjust my requirements and preferences as desired.	○	○	○	○	○
The subtree recommender is helpful.	○	○	○	○	○
The subtype recommender is helpful.	○	○	○	○	○



What type of requirements/preferences, if any, were not easy to specify?

C Questionnaire for the Evaluation of the Judgment Recommendation Mechanism

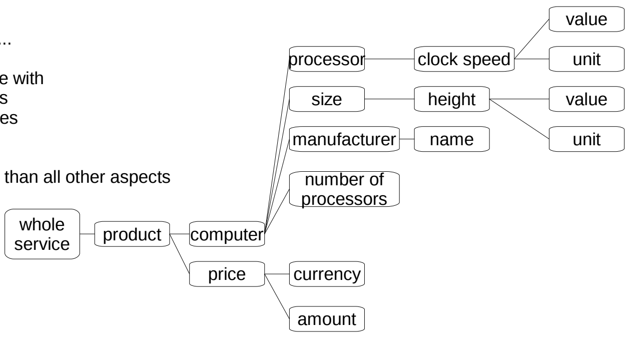
This appendix contains the questionnaire that was used to elicit the test data for the evaluation of the mechanism for judgement target recommendation that has been presented in Section 6.5.

Desktop PCs

Desktop Request Model 1
You want to buy a computer, namely a ...

Desktop PC from IBM, Lenovo or Apple with
at least two 2.0 GHz processors
and a height of at most 20 inches
for at most 2000 $

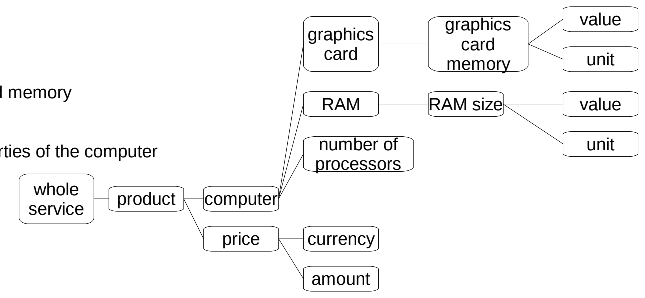
the price is very important to you, more than all other aspects



Desktop Request Model 2
You want to buy a computer, namely a ...

Desktop PC with
at least 3 processors
at least 1GB RAM
At least 350MB graphics card memory
for at most 2000 $

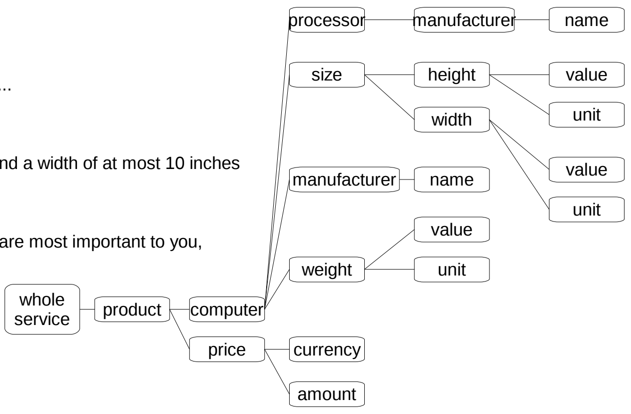
both, the price and the desired properties of the computer are equally important to you



Desktop Request Model 3
You want to buy a computer, namely a ...

Desktop PC from Apple with
an Intel processor
a height of at most 20 inches and a width of at most 10 inches
a weight of at most 15 pounds
for at most 3500 $

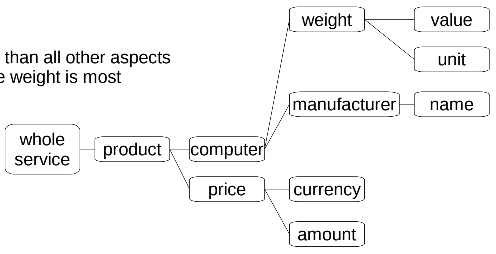
the desired properties of the computer are most important to you, more than the price



Desktop Request Model 4
You want to buy a computer, namely a ...

Desktop PC from IBM with
a weight of at most 17 pounds
for at most 1000 $

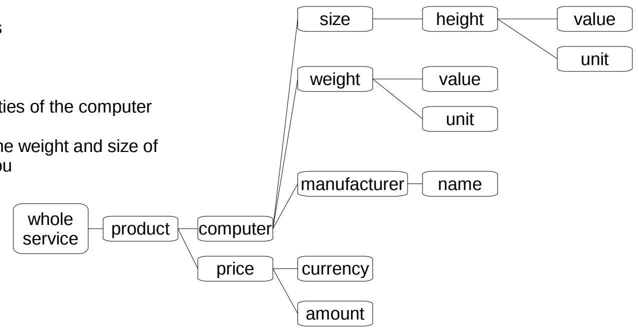
the price is very important to you, more than all other aspects
among the computer characteristics the weight is most
important to you



Desktop Request Model 5
You want to buy a computer, namely a ...

Desktop PC from Dell with
a weight of at most 17 pounds
a height of at most 20 inches
for at most 1000 $

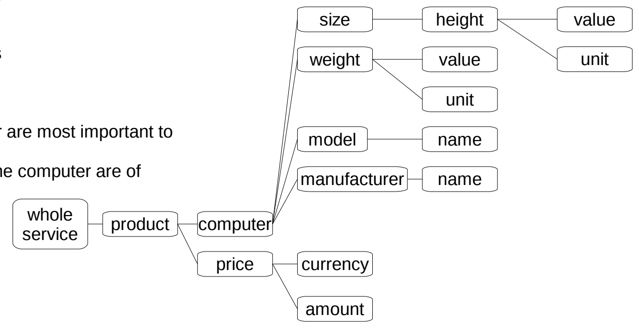
both, the price and the desired properties of the computer
are equally important to you
among the computer characteristics the weight and size of
the computer are most important to you



Desktop Request Model 6
You want to buy a computer, namely a ...

Desktop PC C315 from Lenovo with
a weight of at most 17 pounds
a height of at most 20 inches
for at most 1000 $

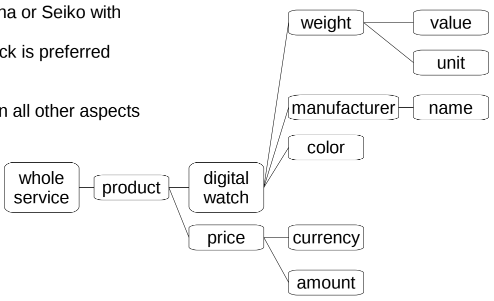
the desired properties of the computer are most important to
you, more than the price
the model and the manufactuerer of the computer are of
particular importance to you



Digital Watches

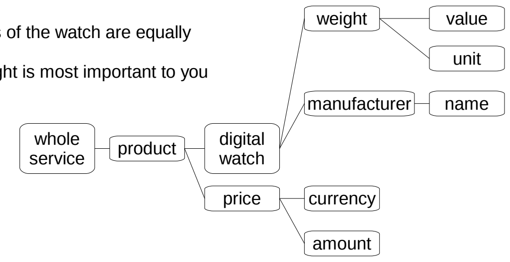
Digital Watch Request Model 1
You want to buy a digital watch from Karenna or Seiko with
a weight of at most 40 g
that is either silver or black, but black is preferred
for at most 100 $

the price is very important to you, more than all other aspects



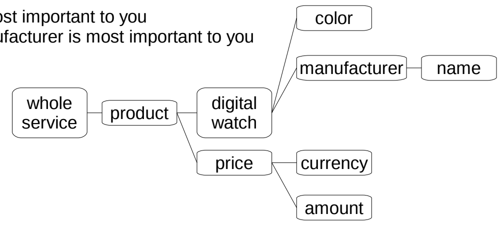
Digital Watch Request Model 2
You want to buy a digital watch from Schylling, Gametime or Nike with
a weight of at most 45 g
for at most 200 $

both, the price and the desired properties of the watch are equally important to you
among the watch characteristics the weight is most important to you



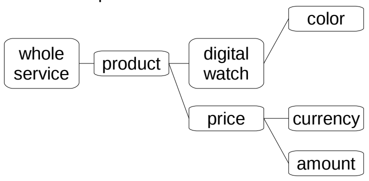
Digital Watch Request Model 3
You want to buy a black digital watch from Nike
for at most 200 $

the desired properties of the watch are most important to you
among the watch characteristics the manufacturer is most important to you



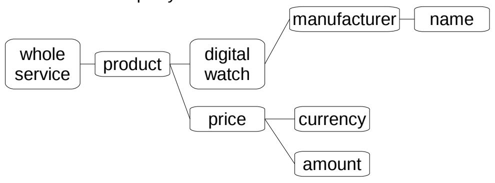
Digital Watch Request Model 4
You want to buy a silver digital watch
for at most 100 $

the price is very important to you, more than all other aspects



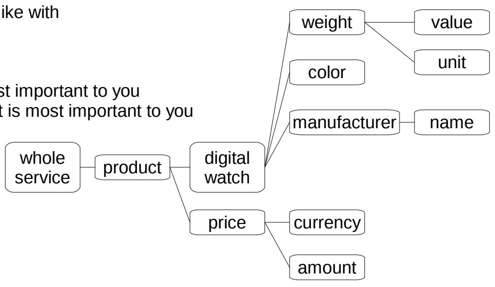
Digital Watch Request Model 5
You want to buy a digital watch from Xonic, Gametime or Nike
for at most 200 $

both, the price and the desired properties of the watch are equally important to you



Digital Watch Request Model 6
You want to buy a pink digital watch from Nike with
a weight of at most 45 g
for at most 200 $

the desired properties of the watch are most important to you
among the watch characteristics the weight is most important to you


weight
value
unit
color
manufacturer
name
whole service
product
digital watch
price
currency
amount