Scrutable Adaptivity in Community-Enabled Web Portals

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 M.Eng. Fedor Bakalov

geboren am 02. März 1979 in Sosnovka, Kirgisistan

Gutachter

1. Prof. Dr. Birgitta König-Ries
   Friedrich-Schiller-Universität Jena, D-07743 Jena

2. Prof. Dr. Martin Welsch
   Friedrich-Schiller-Universität Jena, D-07743 Jena, Deutschland
   IBM Deutschland Research & Development GmbH, D-71032 Böblingen

3. Prof. Dr. Peter Brusilovsky
   University of Pittsburgh, Pittsburgh, PA 15260, USA

Tag der öffentlichen Verteidigung: 17. September 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 eigener 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, 17. Juli 2012

[Fedor Bakalov]

Wissenschaftliche Publikationen des Promovenden

[1] F. Bakalov, M.-J. Meurs, B. König-Ries, B. Sateli, R. Witte, G. Butler, and A. Tsang. Personalized semantic assistance for curation of biochemical literature. In *IEEE Int. Conf. on Bioinformatics and Biomedicine*, 2012.

[2] F. Bakalov, B. Sateli, R. Witte, M.-J. Meurs, and B. König-Ries. Natural language processing for semantic assistance in web portals. In *6th IEEE Int. Conf. on Semantic Computing*, 2012.

[3] I.-H. Hsiao, G. Julio, D. Parra, F. Bakalov, B. König-Ries, and P. Brusilovsky. Comparative social visualization for personalized e-learning. In *Proc. of the 11th Int. Working Conf. on Advanced Visual Interfaces*, 2012.

[4] F. Bakalov, I.-H. Hsiao, P. Brusilovsky, and B. König-Ries. Progressor: Personalized visual access to programming problems. In *Proc. of the IEEE Symposium on Visual Languages and Human-Centric Computing*, 2011.

[5] F. Bakalov, I.-H. Hsiao, P. Brusilovsky, and B. König-Ries. Visualizing student models for social learning with Parallel IntrospectiveViews. In *Proc. of the Workshop on Visual Interfaces to the Social and Semantic Web held in conj. with the Int. Conf. on Intelligent User Interfaces*, 2011.

[6] F. Bakalov, B. König-Ries, T. Hennig, and G. Schade. Usability study of a semantic user model visualization for social networks. In *Workshop on Visual Interfaces to the Social and Semantic Web held in conj. with the Int. Conf. on Intelligent User Interfaces*, 2011.

[7] I.-H. Hsiao, F. Bakalov, P. Brusilovsky, and B. König-Ries. Open social student modeling: Visualizing student models with Parallel IntrospectiveViews.

In *Proc. of the 19th Int. Conf. on User Modeling, Adaptation, and Personalization*, 2011.

[8] F. Bakalov, B. König-Ries, A. Nauerz, and M. Welsch. IntrospectiveViews: An interface for scrutinizing semantic user models. In *Proc. of the 18th Int. Conf. on User Modeling, Adaptation, and Personalization*, 2010.

[9] F. Bakalov, B. König-Ries, A. Nauerz, and M. Welsch. Scrutinizing user interest models with IntrospectiveViews. In *Adjunct proc. of the 18th Int. Conf. on User Modeling, Adaptation, and Personalization*, 2010.

[10] O. Schimratzki, F. Bakalov, A. Knoth, and B. König-Ries. Semantic enrichment of social media resources for adaptation. In *Proc. of the Workshop on Adaptation in Social and Semantic Web held in conj. with the 18th Int. Conf. on User Modeling, Adaptation, and Personalization*, 2010.

[11] F. Bakalov, B. König-Ries, A. Nauerz, and M. Welsch. Automating mashups for next-generation enterprise portals. *IT Professional*, 11(4):6–9, 2009.

[12] F. Bakalov, B. König-Ries, A. Nauerz, and M. Welsch. A hybrid approach to identifying user interests in web portals. In *Proc. of the 9th Int. Conf. on Innovative Internet Community Systems*, Jena, Germany, 2009.

[13] F. Bakalov, B. König-Ries, A. Nauerz, and M. Welsch. A user modeling server for determination of semantically-enriched user interests. In *Proc. of the Int. Workshop on Ubiquitous User Modeling held in conj. with the 1st and 17th Int. Conf. on User Modeling, Adaptation and Personalization*, 2009.

[14] T. Fischer, F. Bakalov, B. König-Ries, A. Nauerz, and M. Welsch. An evolutionary algorithm for automatic composition of information web services in mashups. In *Proceedings of the 7th IEEE European Conference on Web Services*, 2009.

[15] T. Fischer, F. Bakalov, and A. Nauerz. An overview of current approaches to mashup generation. In *Proc. of the Int. Workshop on Knowledge Services and Mashups*, 2009.

[16] A. Kreiser, A. Nauerz, F. Bakalov, B. König-Ries, and M. Welsch. A web 3.0 approach for improving tagging systems. In *Proc. of the Int. Workshop on Web 3.0: Merging Semantic Web and Social Web (in conjunction with the 20th Int. Conf. on Hypertext and Hypermedia 2009)*, 2009.

[17] A. Nauerz, F. Bakalov, M. Welsch, and B. König-Ries. New tagging paradigms for content recommendation in web 2.0 portals. In *Proc. of the Int. Workshop*

on Adaptation and Personalization for Web 2.0 held in conj. with the 1st and 17th Int. Conf. on User Modeling, Adaptation and Personalization, 2009.

[18] A. Nauerz, M. Brück, M. Welsch, F. Bakalov, and B. König-Ries. New tagging paradigms for enhancing collaboration in web 2.0 communities. In *Proc. of the 17th Workshop on Adaptivity and User Modeling in Interactive Systems*, 2009.

[19] A. Nauerz, M. Welsch, F. Bakalov, and B. König-Ries. Link clouds and user-/community-driven dynamic interlinking of resources. In *Proc. of the 17th Workshop on Adaptivity and User Modeling in Interactive Systems*, 2009.

[20] F. Bakalov, B. König-Ries, A. Nauerz, and M. Welsch. Ontology-based multidimensional personalization modeling for the automatic generation of mashups in next-generation portals. In *Proc. of the 1st Int. Workshop on Ontologies in Interactive Systems held in conj. with HCI*, 2008.

[21] T. Fischer, F. Bakalov, and A. Nauerz. Towards an automatic service composition for generation of user-sensitive mashups. In *Proc. of the 16th Workshop on Adaptivity and User Modeling in Interactive Systems*, 2008.

[22] A. Nauerz, F. Bakalov, B. König-Ries, and M. Welsch. Adaptive portals: Adapting and recommending content and expertise. In *Proc. of the 16th Workshop on Adaptivity and User Modeling in Interactive Systems*, 2008.

[23] A. Nauerz, F. Bakalov, B. König-Ries, and M. Welsch. Personalized recommendation of related content based on automatic metadata extraction. In *Proc. of the 18th Int. Conf. on Computer Science and Software Engineering*, 2008.

[24] A. Nauerz, S. Schindler, and F. Bakalov. Adaptive treemap based navigation through web portals. In *Proc. of the 16th Workshop on Adaptivity and User Modeling in Interactive Systems*, 2008.

[25] F. Bakalov. An ontology-based approach to designing information architecture of websites. Master's thesis, Asian Institute of Technology, 2007.

Patent Applications

[26] F. Bakalov, B. König-Ries, A. Nauerz, and M. Welsch. An interactive ring-shaped interface for visualization of semantically-enriched user interest models. Patent Application DE9-2010-014 (under evaluation), 2010.

[27] F. Bakalov, B. König-Ries, A. Nauerz, and M. Welsch. A visual rich-interaction approach to user-controlled personalization in web portals. Patent Application DE9-2010-0095 (under evaluation), 2010.

Deutsche Zusammenfassung

Personalisierte Informationssysteme sind als Gegenmaßnahme zu Informationsüberflutung und steigender Komplexität der Navigation im Informationsraum entstanden. Solche Systeme sind in der Lage, über die Informationsbedürfnisse des Benutzers zu lernen, um Inhalte, Funktionen und Benutzeroberfläche an den jeweiligen Benutzer anzupassen. Webportale sind eine der Erstanwender der Personalisierungstechnologie. Beispiele für personalisierte Webportale reichen von e-Commerce Portalen, die anhand der bisherigen Einkäufe des Benutzers für ihn interessante Produkte empfehlen, bis hin zu Nachrichtenportalen, die die vom Benutzer besuchten Seiten verwenden, um neue Nachrichten nach individuellen Interessen des Benutzers zu sortieren. Das Hauptziel solcher Verfahren ist es, dem Benutzer zu helfen, relevante Inhalte schneller und effizienter zu finden.

Um eine solche adaptive Verhaltensweise zu erreichen, benötigt das System etliche Personalisierungsmodelle. Insbesondere braucht es ein Benutzermodell, das bestimmte Informationen über den Benutzer beinhaltet, z.B. seine Interessen, Ziele, Kenntnisse, usw. Zusätzlich braucht es Metadaten von Inhalten. Um die automatische Auswahl von Inhalten, die den Benutzerbedürfnisse entsprechen, zu er-reichen, müssen das Benutzermodell und Metadaten in derselben Sprache definiert werden, was ein Domainmodell fordert. Schließlich braucht das System Personalisierungsregeln, die die Logik der Anpassung von Inhalten bestimmen.

In den meisten personalisierten Systemen sind diese Modelle und der Anpassungsprozess dem Benutzer verborgen. Er sieht nur die Auswirkung der Personalisierung, kann aber weder auf die persönlichen Informationen, die das System über ihn sammelt, zugreifen, noch Einfluss auf die Anpassungslogik nehmen. Dadurch können u.a. veraltete oder falsche Informationen zu falschen oder irrelevanten Empfehlungen führen. Das Verdecken des Zugangs zum Benutzermodell steht ggf. auch im Konflikt mit dem gesetzlich verbrieften Recht auf informationelle Selbstbestimmung in Deutschland bzw. der EU [1, 80, 117]. Darüber hinaus werden mehrere anerkannte Usability-Prinzipien, u.a., Transparenz, Steuerbarkeit und Vorhersehbarkeit, verletzt bzw. beeinträchtigt [67, 70].

Um diese Probleme zu vermeiden, sollte der Benutzer volle Kontrolle über das Benutzermodell und den Personalisierungsprozess erhalten. In dieser Dissertation präsentieren wir eine Lösung der o.g. Probleme. Wir beschreiben ein Framework für benutzerkontrollierte Personalisierung in Community-getriebenen Webportalen. Dies Framework bietet vier Modelle, die ein personalisiertes Portal braucht: (1) ein Benutzermodell für die Darstellung von Informationen über individuelle Bedürfnisse der Benutzer, (2) ein Metadaten-Repository für Darstellung von Annotationen der Portalinhalte, (3) ein Domainmodell für Darstellung von maschinenlesbarem Wissen, das als Vokabular für Definition der Benutzerinformationen und Metadaten dient und (4) Personalisierungsregeln, die die Logik von Adaptionen definieren. Darüber hinaus bietet das Framework Methoden für eine ständige Aktualisierung, Vervollständigung und Kontrolle auf Korrektheit für diese vier Modelle. Unter anderem nutzen diese Methoden die Technologie für Textanalyse und die Bereitschaft der Benutzer-Community, Inhalte zu annotieren. Außerdem bietet das Framework graphische Benutzeroberflächen und Interaktionsmuster, die dem Benutzer ermöglichen, Informationen dieser Modelle leicht zu verstehen und anzupassen.

Einer der wichtigsten Beiträge dieser Arbeit ist eine neuartige Benutzeroberfläche zur Darstellung von großen und komplexen Benutzermodellen IntrospectiveViews. Für die Visualisierung verwendet es eine Metapher kreisförmiger Zonen, die in Sektoren gesplittet sind. Dabei repräsentieren die Zonen die Interessengruppen der dargestellten Objekte und die Sektoren bezeichnen die semantischen Typen der Objekte. Die Visualisierung bietet Funktionen für Zoomen, Filtern, Navigation und Suche. Außerdem kann der Benutzer über IntrospectiveViews sein Benutzermodell schnell und effizient ändern. Er kann Interessen einfügen und löschen, den Interessensgrad ändern, neue Kategorien erstellen und Objekte semantisch verknüpfen. In dieser Dissertation präsentieren wir Ergebnisse der drei Benutzerstudien von IntrospectiveViews. Die Ergebnisse zeigen, dass Benutzer die Visualisierung für intuitiv, benutzerfreundlich, visuell attraktiv und motivierend befinden.

Ein anderer wichtiger Beitrag dieser Arbeit ist eine visuelle Methode zur Benutzerinspektion und Steuerung von Personalisierungsregeln und Personalisierungseffekten in Webportalen. Unser Verfahren ermöglicht, dass der Benutzer einen Über-blick über alle Adaptionen, die das Portal durchführt, bekommen kann. Außerdem ermöglich es dem Benutzer, diese Adaptionen zu beeinflussen. Das Werkzeug für Personalisierungssteuerung, das wir in dieser Arbeit entwickelt haben, wird direkt neben den personalisierten Inhalten angezeigt. Der Benutzer kann damit die Personalisierungseffekte nach Bedarf ein- und ausschalten, sowie sein Interessenprofil ändern. Jede Änderung von Personalisierungseffekten oder Benutzerinteressen wird sofort auf den personalisierten Inhalt projiziert. Dies hilft dem Benutzer, den Einfluss von seinem Interessenprofil und Personalisierungsregeln auf den personalisierten Inhalten besser zu verstehen. Das ermöglicht ihm auch die adaptive Verhaltensweise des Portals an seine Präferenzen und Bedürfnisse anpassen. Unser

Verfahren kann für eine gezielte Personalisierungssteuerung auf der Portlet-Ebene verwendet werden. Der Benutzer kann in jedem einzelnen Portlet die Personalisierungsregeln ändern und, wenn nötig, die Personalisierung komplett ausschalten.

Zuletzt präsentieren wir in dieser Dissertation Ergebnisse einer Benutzerstudie, in der wir die Auswirkung unseres Verfahrens auf die Benutzerempfindung von adaptiven Webportalen evaluiert haben. In dieser Studie haben wir ein adaptives Portal ohne Unterstützung von Personalisierungssteuerung mit einem identischen Portal mit Personalisierungssteuerung gemäß unseres Verfahrens vergleichen. Die Ergebnisse dieser Studie zeigen, dass sich unser Verfahren signifikant auf viele Aspekte personalisierter Webportale auswirkt. Dadurch wird die adaptive Verhaltensweise transparenter und verständlicher. Unser Verfahren steigert auch den Nutzwert, Benutzerfreundlichkeit und Benutzerzufriedenheit personalisierter Systeme.

Abstract

Personalized systems emerged as a solution to the problem of steadily growing amounts of information and constantly increasing complexity of navigation in the information space that overwhelms users. These systems are able to learn about the needs of individual users and to tailor the content, appearance, and behavior to the user needs. Web portals are one of the earliest adopters of personalization technology. Examples of personalized portals range from news aggregating portals that recommend interesting news stories identified based on the stories that the user read in the past to e-commerce portals recommending products identified based on the user's previous purchases. The aim of such adaptive behavior is to help users to find relevant content easier and faster.

To achieve this type of adaptivity, an adaptive system needs a number of personalization models. More precisely, it needs a user model providing information about users, e.g., about their interests, expertise, background, or traits. It also needs metadata of information resources. To achieve automatic selection of the resources that match the user's individual needs, both the user model and metadata should use the same vocabulary, which requires a domain knowledge model represented in a formalism that can be interpreted by the portal. Finally, the system needs some logic or rules that govern how the resources must be delivered given the user model and metadata.

In most personalized systems, these models and the process of adaptation are hidden from users. Users see only the resulting personalization effects. But they do not have direct access to the information the system collects about them and do not have control over the personalization behavior. This results in a number of grave usability and privacy problems. It violates such principles as predictability, transparency, controllability, and unobtrusiveness [67, 70]. Also, it violates privacy legislation and negatively impacts the trustworthiness of adaptive systems [1, 80, 117]. Finally, preventing users from controlling personalization may cause that the models that the system uses for adaptation are incomplete or contain obsolete or

inaccurate information. This, in its turn, leads to wrong adaptive behavior and irrelevant recommendations.

In this thesis, we propose an approach to solve the above-mentioned problems. We present a framework for scrutable adaptivity in community-enabled web portals. This framework provides four models that an adaptive system needs for personalization: (1) a user model representing information about individual users, (2) a metadata repository providing annotations of content, (3) domain model defining machine-processable semantics of the domain knowledge used for modeling user features and annotating content, and (4) personalization rules defining the logic of adaptation. Also, the framework provides methods for keeping these models up-to-date, complete, and accurate. Among others, these methods leverage the technology for Natural Language Processing and the willingness of the user community to contribute and annotate content. Furthermore, the framework provides graphical user interfaces and interaction patters for allowing users to view and adapt these models.

One of the key contributions of this research is a novel interface for scrutinizing large and complex semantic user models IntrospectiveViews. The interface visualizes such models leveraging a metaphor of circular zones partitioned into slices, where each zone represents items of certain interest degree and each slice represents items of a specific type. The visualization provides functions for getting overview, zooming, filtering, navigation, and search. It also displays relevant content and semantic relations among items. In addition to viewing, IntrospectiveViews allows users to edit the model in an easy-to-use and efficient manner. It allows adding and deleting items, changing status of items, organizing items by type, defining user own types, and creating semantic relations among items. In this thesis, we report on several users studies of IntrospectiveViews. Results of these studies show that users deem the visualization simple-to-understand, user friendly, visually attractive, and engaging.

Another important contribution of our research is a visual method for scrutinizing personalization rules and effects in web portals. Our approach gives users an overview of all adaptations that the portal makes. Also, it allows users to influence these adaptations by either changing personalization rules or editing the user model. The tool for scrutinizing personalization is placed in the direct proximity to the personalized content. Any change that the user makes on personalization rules or the user model takes effect immediately on the personalized content. This helps users to understand the connection between these two components and the end personalization effects. Also, it allows them to easily adjust the adaptive behavior to their needs and preferences. Our approach can be used for fine-tuning of personalization effects at the portlet level. Users can customize personalization rules and, if necessary, deactivate personalization completely for each portlet individually.

Finally, we report on a user study evaluating impacts of our approach as a whole on the user's perception of personalized portals. In this study, we compared a personalized portal without support for scrutable adaptivity with an identical personalized portal in which such support was provided following our approach. Results of this study show that our approach makes significant impacts on a number of aspects of personalized portals. It makes the adaptivity more transparent and understandable. It improves the usefulness of personalized portals and makes users be more satisfied with them.

Acknowledgments

I would like to thank all those who gave me the possibility to complete this dissertation. I would have never been able to finish it without their support, contributions, and care.

Above all, I am sincerely grateful to my supervisor, Prof. Dr. Birgitta König-Ries, who supported me throughout the entire course of my Ph.D. with her knowledge and guidance. She was always responsive, encouraging, and helpful in many ways. I admire her style of supervision – giving students valuable advices and feedback whilst leaving them the room to work in their own way.

Also, I sincerely thank Prof. Dr. Martin Welsch for co-supervising my thesis. His experience in both academia and industry helped me to find a balance between research-oriented and practical aspects of my dissertation. I thank him for his guidance and support as well as for his hospitability and care that he provided during my visits to the IBM Lab in Böblingen.

Besides I would like to acknowledge the support of two other professors, Prof. Peter Brusilovsky and Assistant Professor Dr. René Witte. I thank Prof. Brusilovsky for his dedication to review my thesis and his valuable comments. I thank Dr. Witte for hosting me several times at the Semantic Software Lab at Concordia University in Montreal and providing me a wonderful opportunity to conduct parts of my research there.

I thank all my colleagues and project partners who helped me during my doctoral research. Special thanks go to Ulrich Küster and Andreas Nauerz. Ulrich, a colleague and a good friend, was a source of help and aspiration from the very beginning of my Ph.D. He helped me not only to set high standards for conducting research, but also to enjoy and have fun doing Ph.D. I thank Andreas for sharing his expertise in portal technology, providing feedback on my research, and helping me during my stays at the IBM Lab in Böblingen. Besides I express my gratitude to Marie-Jean Meurs and Bahar Sateli for helping me during my visits to Concordia University. I also thank a colleague of mine Aygul Gabdulkhakova for helping me

to settle things down upon my arrival in Jena and encouraging me throughout my Ph.D.

I express my gratitude and appreciation to IBM, especially to the IBM's Center for Advanced Studies and Ph.D. Fellowships Program, under which my research was funded. I would have not been able to complete my Ph.D. without the provided financial support and the access to IBM knowledge, technology, software, and infrastructure. I would also like to thank Friedrich Schiller University of Jena for the academic, technical, and financial support.

I am grateful to those who participated in the studies reported in this thesis for their time, patience, and valuable feedback. Special thanks go to researchers and students of Heinz-Nixdorf Endowed Chair for Distributed Information Systems of the University of Jena, students of Erfurt University of Applied Sciences, and researchers of the Center for Structural and Functional Genomics of Concordia University in Montreal.

Last but in no means least I thank my family and close friends for supporting me throughout my doctoral research. I especially thank my mother, Valentina Bakalova, for her support and encouragement to reach higher goals.

Contents

CHAPTER 1

Introduction

In the early history of the World Wide Web, websites were designed following the "one-size-fits-all" principle. Their look-and-feel, content organization, and navigation structures were designed considering an average user: All users of a website had the same content and navigation mechanisms irrespective of their individual information needs. However, due to the constantly growing amount of information on the Web, it became difficult and time consuming for users to navigate through such websites. The problem of being disoriented and being unable to quickly and efficiently find relevant information, also known as "lost in hyperspace" phenomenon [47], strengthened the need of adapting websites to the user's individual information needs.

In the early 1990s, Adaptive Hypermedia (AH) emerged as a new research field to address the challenges of the "lost in hyperspace" phenomenon. According to Brusilovsky [29], the goal of this research area is to enhance the functionality of hypermedia through personalization and adaptation. This area embraces the systems that build a model of the user's goals, knowledge, and preferences and automatically adapt their presentation and navigation structures based on this model.

One of the earliest adopters of adaptation and personalization techniques were web portals. Personalization in the first portals was present primarily in the form of *adaptability*, i.e., users were able to modify the portal's color scheme as well as add, delete, rearrange, and configure portlets on a page. However, as the field of Adaptive Hypermedia was maturing, a number of proposals were made to introduce *adaptivity* in web portals to achieve system-generated adaptation effects. Users of an adaptive web portal may encounter different kinds of adaptation effects, e.g., the portal's front page displaying recently added resources that the user is interested in, a side portlet recommending relevant documents, a document augmented with additional information that matches current interests of the user, and so on.

In order to achieve such adaptation effects, the following four requirements must be met (see Section 3.1). First, the portal must have a *user model* containing information about users, e.g., their interests, expertise, traits, goals, and so on. Second, portal resources must be described with *metadata* representing machine-processable semantics of the resources. Third, to achieve automatic selection of the resources that match the user's individual needs, both the user model and metadata should use the same vocabulary, which requires a *domain knowledge model* represented in a formalism that can be interpreted by the portal. Forth, the portal needs *personalization rules* governing what adaptation effects should be made for a certain user, when, and how.

1.1. Motivation

In most adaptive portals users can only see the end adaptation effects. The mechanism of the adaptation process, user and domain knowledge models, and adaptation rules are often hidden from the user. This, however, results in a number of grave *usability and privacy* problems and may cause the information the system uses for adaptation to become *obsolete, inaccurate, or incomplete*.

Usability and Privacy Challenges of Adaptive Systems

Hiding the information that is being used for adaptation decreases the system's transparency and predictability [67] and as a result may outweigh the benefits of adaptation. Also, inability of the user to access and control this information violates the privacy legislation [80]. Jameson [70] identified five challenges of adaptive systems with respect to the usability and privacy issues:

- **Challenge 1: Predictability and transparency.** Predictability refers to the system's property that allows the user to anticipate the system's behavior, whereas transparency is the extent to which the system's status and actions are visible and understandable. Both properties can be negatively affected by hiding the information that the system uses for adaptation from the user and/or by not explaining why and based on what information a certain adaptation action has been performed.
- **Challenge 2: Controllability.** Controllability is the property that allows users to bring the system in the state they want. Inability of the user to control the adaptation process might lead to unwanted and inappropriate adaptation effects.
- **Challenge 3: Obtrusiveness of adaptive behavior.** Unobtrusiveness refers to the style of adaptive behavior that does not disturb the user from

performing her primary tasks. Even though the main aim of adaptive systems is to assist the user in performing her tasks, it is still very unlikely to achieve 100% correctness of the system's proactive behavior. Also, the system's adaptation effects can be perceived differently: some users might appreciate the system's proactive behavior, whereas some may perceive it as a disturbance.

- **Challenge 4: Privacy.** Most user-adaptive systems gather, maintain, and use for adaptation certain information about the user. Preventing the user form accessing this information is a serious violation of privacy laws [80, 117]. For instance, according to the German Act on the Protection of Personal Data Used in Teleservices (Gesetz über den Datenschutz bei Telediensten),

  > "§ 7 User's right to information
  > The user shall be entitled at any time to inspect, free of charge, stored data concerning his person or his pseudonym at the provider's. The information shall be given electronically if so requested by the user." [1]

- **Challenge 5: Breadth of experience.** Another serious problem of adaptive systems is that by filtering out irrelevant or uninteresting (from the system's point of view) content, adaptive systems narrow user experience. In an adaptive system, user's knowledge about domains not captured by the user model is poorer than it would be in a non-adaptive system.

Challenges related to the completeness and correctness of personalization models

Preventing users from accessing the personalization models, including the user and domain models, resource metadata, and adaptation rules, might also result in incorrect adaptive behavior of the system. This problem can be intensified in community-enabled portals, where the content is generated more dynamically. With this respect, the following challenges exist and should be taken into account.

- **Challenge 6: Complete and accurate user model.** Inability of users to access and edit the user model might result in the system having wrong and/or incomplete understanding about them. This might lead to issuing incorrect recommendations and/or not recommending relevant resources.

- **Challenge 7: Complete and accurate domain knowledge.** Hiding the domain knowledge model from users might lead to the state when the model is not anymore up-to-date or contains information that does not reflect the domain knowledge of a certain group of users. Since up-to-date domain knowledge is an important condition for inference about user features and for

recommendations, having this information incomplete may cause the system to make wrong assumptions about users or resources and as a result make irrelevant recommendations.

- **Challenge 8: Timely and accurate annotation of resources.** Having metadata on the resources in an adaptive system is an important requirement for achieving automatic selection of relevant documents and recommending them to the user. However, in the systems where the content is generated by the user community, it is difficult, if not impossible, to achieve timely description of resources by the administrator due to the dynamics of user-generated content.

- **Challenge 9: User-tailored adaptation rules.** Due to the dynamics of user-generated content it is also difficult to come up with one set of adaptation rules that serve the entire community of users. Different users might like different adaptation effects. For example in a news portal, some users might wish to have news stories sorted by interest, from the most interesting to the least interesting, whereas other users, might wish that the stories are sorted chronologically, but the most interesting ones are highlighted. Therefore, users must have an option to create own adaptation rules or select appropriate ones from the list of predefined rules.

Challenges of Scrutable Adaptivity

To mitigate the usability and privacy problems of adaptive systems and to ensure correctness and completeness of the personalization models, a number of solutions have been proposed recently. Researchers have been experimenting with scrutable user models that can be inspected and altered by the user. Also, a number of adaptive systems have been proposed to provide the user with access to the models representing the domain knowledge. However, introducing scrutability in an adaptive system has also a number of challenges.

- **Challenge 10: Lack of awareness of scrutability.** Since scrutability has not yet become an integral feature of all adaptive systems an average user might not have developed the mental model of affordance to control adaptation [50].

- **Challenge 11: Lack of motivation to scrutinize.** Scrutinizing adaptation is certainly not the main task the user wants to accomplish in an adaptive system. Therefore, the user might not have sufficient motivation to use scrutinizing tools, especially if it requires training or certain skills.

- **Challenge 12: Complexity of adaptation.** In case of the systems applying complex adaptation models and rules, it might be difficult for users to understand the adaptation process and adjust it to their personal needs.
- **Challenge 13: Obtrusiveness of scrutability.** The presence of tools for scrutinization might be perceived by some users as an obtrusion and distract them from completing their primary tasks.

Portal Technology Related Challenges

Web portals provide a single-point of access to a broad spectrum of content sources and applications. For instance, in an enterprise portal users can get access to all the company's information, such as news, documents, forms, and databases. Also, in such a portal they can access the company's applications and legacy systems, such as accounting systems, applications for Supply Chain Management, Customer Relationship Management, etc. All the information and functions in a portal are delivered to users through so called portlets (see Section 2.2). Portal technology enables aggregating multiple portlets with different functions and content on one page to support users in carrying out complex tasks. This feature imposes the following challenge on scrutable adaptivity in web portals.

- **Challenge 14: Portlet-level personalization.** Personalization effects occurring in portlets may vary from portlet to portlet. For instance, a portlet displaying search result may sort the hits by the relevance to the user interest profile. A portlet that displays news stories may highlight the most interesting for the user stories by color. Moreover, a portlet may support multiple personalization effects. Different users might wish different personalization effects for the same portlet. Also, on a portal page with multiple portlets, a user might wish that only certain portlets are personalized, but not all of them.

1.2. Objectives

In this document, we propose an approach to scrutable adaptivity in community-enabled web portals. The main objectives of this approach are to overcome the usability and privacy problems of adaptive portals in general and to mitigate the problems of incompleteness and incorrectness of personalization models in community-enabled adaptive portals. Also, it aims to solve the issues specific to scrutable adaptivity, namely, insufficient awareness of an average user about the existence of scrutinizing tools and lack of motivation to use them as well as the problems of complexity of adaptation and obtrusiveness of scrutability. These objectives are to be achieved by accomplishing the following goals:

- **Goal 1: Develop tools enabling scrutinization of personalization models.** These tools must provide easy-to-use and efficient mechanisms and interfaces allowing users to view and edit the user model, domain knowledge, content metadata, and adaptation rules.
- **Goal 2: Develop a visual method for explaining personalization effects.** This method must display personalization effects in a way that portal users can comprehend. It must ensure that users are aware of personalization in the portal, know what information the portal uses for personalizing content, and know how to access and adapt this information in order to adjust personalization to their needs and preferences.
- **Goal 3: Validate the proposed approach.** We aim to proof whether our solution allows users to scrutinize adaptivity in web portals in an easy-to-use and efficient way. Also, we aim to determine whether our approach solves the above-mentioned problems of personalized systems. Additionally, we aim to evaluate the impacts our approach makes on the user's perception of such systems.

1.3. Organization of the Document

The rest of this document is organized as follows. Chapter 2 provides **background** information and basic concepts of the research areas related to this thesis. The chapter provides an introduction to adaptivity on the web, describes examples of user-adaptive systems, and elaborates on user modeling, i.e., explains what information about users is gathered by adaptive systems, how, and for what purpose. Then the chapter introduces web portals, which are the target applications of this thesis, and provides the necessary details about the portal technology and personalization in web portals. Finally, the chapter provides background on the Social Semantic Web. It elaborates on such concepts as user-generated content and metadata as well as explains the main ideas and technologies of knowledge representation on the Web.

In Chapter 3, we elaborate on the **functional and qualitative requirements for scrutable adaptivity in community-enabled web portals**. We ground these requirements on the foundations of adaptive systems, web portal technology, and social semantic web that we have described in Chapter 2. Moreover, we explain the importance of these requirements using the challenges reported in Section 1.1, namely the usability and trust problems of adaptive systems, the issues related to correctness and completeness of personalization models, and the obstacles of scrutable adaptivity.

Chapter 4 presents a review of the **state of the art in scrutable adaptivity**. It describes the approaches that have been proposed for scrutinizing personalization models, namely, approaches to scrutable user modeling, systems that allow the user to customize adaptation rules, methods to visualizing the domain knowledge models, and the approaches to user-generated metadata provision in adaptive systems. Also, the chapter describes the systems that provide users with overview and explanation of adaptation effects as well as the systems that allow users controlling personalization. Finally, the chapter provides a comprehensive comparison of the related systems with respect to the requirements identified in Chapter 3.

In Chapter 5, we lay out a **framework for scrutable adaptivity in community-enabled web portals**. We provide an overview of the conceptual architecture of the framework and its four units. Additionally, we describe two web portals that the framework was integrated into, namely a news aggregating portal and a portal for biochemical literature. We use these portals to describe the system components of the framework and give examples of user interaction with the tools for scrutable adaptivity. Also in this chapter, we describe two units of the framework that provide auxiliary functions. We present the **Domain Modeling Unit** that stores the domain knowledge model and provides management and access operations on this model. We also describe the **Resource Management Unit** that generates, stores, and provides access to the semantic metadata of portal content.

The **User Modeling Unit** is presented in Chapter 6. This unit is responsible for storing, updating, and managing a user model that provides machine-processable information about features of individual users. Also, we present IntrospectiveViews, an interactive visualization of user models. This visualization allows users to view and alter their user models in an efficient and easy-to-use way. We describe three iterations this visualization and report on results of two studies evaluating its usability and hedonic qualities.

Chapter 7 presents the **Personalization Unit**, the last part of the framework. This unit provides rules and mechanisms for tailoring portal content to needs and preferences of individual users. It also provides graphical user interfaces for delivering personalized content and allowing users to scrutinize and control the personalization behavior.

In Chapter 8, we present a **validation** of our approach. First, we report on a user study of scrutable adaptivity. This study assesses the impacts of our approach on the usefulness, usability, user satisfaction, transparency, and trustworthiness of personalized portals. Second, we examine the compliance of our framework with nineteen functional and qualitative requirements that we identified in Chapter 3. For each requirement, we describe how it is fulfilled by our approach.

Finally, Chapter 9 summarizes and concludes this thesis. In this chapter, we revisit the main challenges that motivated this research and discuss how our approach addresses them. Afterwards, we summarize the main contributions of our

research and generalize about them. We also report on the research limitations of this work and outline directions for possible enhancements of our approach and further research.

CHAPTER 2

Background

This chapter provides background information and basic concepts of the research areas related to this thesis. Section 2.1 provides introduction to *adaptivity on the web*, describes examples of user-adaptive systems, and elaborates on user modeling, i.e., explains what information about users is gathered by adaptive systems, how, and for what purpose. Section 2.2 introduces *web portals*, which are the target applications of this thesis, and provides the necessary details about the portal technology and personalization in web portals. Finally, Section 2.3 provides background on the *Social Semantic Web* and elaborates on such concepts as user-generated content and metadata as well as explains the main ideas and technologies of knowledge representation on the Web.

2.1. Adaptivity on the Web

Adaptive Hypermedia (AH) emerged as a new research field to address the challenges of the "lost in hyperspace" phenomenon. According to Brusilovsky [29], the goal of this research area is to enhance the functionality of hypermedia through personalization. This area embraces the systems that build a model of the user's goals, knowledge, and preferences and automatically adapt their presentation and navigation structures based on this model.

2.1.1. Types of Adaptation

Adaptivity can be especially helpful in systems containing a large number of resources and being used by users with different information needs. The adaptivity in such systems can serve various goals. Jameson [70] identified five different types of functions supporting information acquisition that can be provided by user-adaptive

systems: helping users to find information, tailoring information presentation, recommending products, supporting collaboration, and supporting learning.

Helping users to find information. The functions of this type aim at assisting the user with finding relevant documents in large information repositories, for instance, in news archives or on the World Wide Web. The support can be provided in three forms: support for browsing, support for query-based search or filtering, and spontaneous provision of information. An adaptive system can provide *support for browsing* by pointing the user to the relevant documents. An example of such support is demonstrated by WebWatcher [71], a system that accompanies the user as she browses on the Web and highlights the hyperlinks where to go next based on the user's browsing interests. An adaptive system can *support query-based search or filtering* by assisting the user in formulating the queries or making it automatically. Tvarozek et al. [125] describe a faceted browser that automatically hides or disables irrelevant facets based on the in-session user behavior as well as her long-term user model. Finally, an adaptive system can *spontaneously provide information* while the user is working on a certain task. An example of such assistance is demonstrated by Watson [34], a system that displays links to relevant information while the user is typing some text in a word processor.

Tailoring information presentation. This type of adaptation aims at presenting content in a way that best suits the user's individual needs. This is when the adaptive system identifies the content fragments of a document, e.g., a Web page, that are most relevant to the user and her context and reorganizes the document's presentation in some way to make the relevant fragments appear more prominent. Bunt et al. [40] describe five "priority on context" techniques for adapting the content presentation, namely, stretch text, dimming, coloring, sorting, and scaling. *Stretch text* and *sorting* are the most "aggressive" techniques of adaptive presentation: the former completely hides the surrounding context from the user and the latter changes the structural information of the original document. Scaling is the least "aggressive" technique as it preserves the distinctive structural elements of the deemphasized information. Tsandials and Schraefel [124] describe an approach to scaling hypertext documents using the fish eye technique. In their approach, certain fragments of a document can grow as the user's interests become relevant to the content of the fragment (Figure 2.1).

Recommending products. This type of adaptation received the highest interest in the industry. Product recommendations can be of mutual interest for the vendor and the customer: they can increase the sales and help the customer to find relevant products. A number of e-commerce websites have included support for this feature.

Figure 2.1.: Stretchtext by Tsandilas and Schraefel [124]

For instance, Amazon[1], based on the customer's purchasing and browsing history, recommends products that the customer might be interested to purchase. Also Amazon offers a feature called "customers who bought this item also bought...": on the page about a certain product the system displays a list of other products that were bought by customers who also bought this product, for which Amazon uses so called *collaborative filtering*. In a similar way, Last.fm[2] recommends music albums and tracks for purchase based on the user's listening history and the listening history of the users with similar preferences. The research literature reports on a number of other product recommenders, including such systems as FindMe [41], MovieLens [92], and SETA [7].

Supporting collaboration. Such support can be provided to the user in the form of recommendations to people or groups of people whom the user might want to get in contact with. For instance, PHelpS [46] recommends to inexperienced workers the peer helpers whom the worker can interact with in order to obtain help on handling a certain task. McDonald and Ackerman [91] describe an expertise recommender system that supports collaborative work within organizations. One of the most well-known social systems that provide this type of support is Facebook[3]. The

[1] http://www.amazon.com
[2] http://www.last.fm
[3] http://www.facebook.com

system recommends the user other Facebook members and communities based on the list of existing friends and the groups that the user is part of.

Supporting learning. This type of support can be provided in intelligent tutoring systems and intelligent environments in such forms as automatic selection and restructuring of instructional materials, provision of problems to solve, tests, and timely feedback. For example, Java Hyperbook [66] provides intelligent tutoring support to students learning Java programming language. The system guides the student through the entire course by pointing her to the next reasonable learning steps and recommending reading lists based on the student's learning goals. A number of adaptive educational systems provide social navigation support. For instance, KnowledgeSea II [31] (Figure 2.2) provides a map-based visual representation of learning materials. The system generates visual cues to indicate the resources that have received more attention from the users within the same group and to show the positive character of annotations made by the group. The darkest cells represent the most useful topics. In addition to the color cues, the system uses four other cues. The keywords represent the content topics, which allows the students easily finding cells by content. Some cells also represent the lecture numbers (shown in red). The stack of papers icon denotes the number of documents in that cell. The yellow icon indicates the user-written notes on the content of the cell. Finally, the human icon indicates the number of the user's own visits of the content of that cell – the darker the icon is, the more frequently the student has accessed the content.

2.1.2. Adapt to What?

A must-have component for any user-adaptive system is the user model. Brusilovsky and Millàn [33] define the user model as *"a representation of information about an individual user that is essential for an adaptive system to provide the adaptation effect, i.e., to behave differently for different users"*. There are a number of the user's aspects that an adaptive system can leverage for adaptation. Brusilovsky and Millàn [33] identify the five most popular and useful features to be modeled about an individual: the user's knowledge, interests, goals, background, and individual traits. In addition to the user's individual features, Brusilovsky identifies a number of attributes describing the user's context of work, which can also be exploited by an adaptive system.

Knowledge. User knowledge models usually define the concepts that users have already background in as well as the concepts that they have to become familiar with. User's knowledge is one of the most important features for adaptive educational systems. For instance, AHA! [53] exploits student knowledge models to

Figure 2.2.: Knowledge Sea II [31]

determine the parts of a course that are most suitable for the student. Brusilovsky and Millàn [33] differentiate user knowledge models by their level of complexity in two groups: scalar models and structural models. The *scalar models*, also known as stereotype models, are the simplest ones: They estimate the user's knowledge of the entire domain on some scale, e.g., good, average, poor. For instance, the MetaDoc [26] system classifies readers by their knowledge of UNIX/AIX systems into four categories: novices, beginners, intermediates, and experts. Based on this model, the system modifies presentation of hypertext documents to better suit the user's level of knowledge. The *structural models* provide more advanced form for representing the user's knowledge: They split the knowledge domain into independent fragments and estimate the user's knowledge for each fragment. The most popular type of structural models is overlay models. An overlay knowledge model defines the user's knowledge as a subset of the domain model representing the expert-level knowledge of the subject [33].

Interests. User interest models define the topics, products, or services that the user is likely to be interested in. Brusilovsky and Millàn [33] differentiate user interest models into two types: keyword-level models and concept-level models. The *keyword-level models*, represent the user's interest as a weighted vector of keywords,

where the keywords represent the topics of interest and the weights assigned to the keywords represent the degree of interest. For example, in Amalthaea [99], user interests are represented as a vector of weighted keywords extracted from the visited Web pages, where the weights are calculated with the TF*IDF weighting scheme [113]. *The concept-level models*, also known as overlay models, represent the user's interest as a subset of concepts defined in the underlying domain model. These models are more powerful than the keyword-level ones. They ensure more accurate representation of user interests and allow for interest propagation. For example, in the PVA system [44], a user interest profile, called personal view, is represented as a subset of hierarchy of categories, called world view. Due to such modular structure, the system achieved high accuracy in personalizing Web content.

Goals and tasks. The models of this type represent information about the purpose or the plan the user might want to accomplish in the system. Modeling user goals has received a high popularity in the area of adaptive interfaces and intelligent help systems. For example, the ADAPTS system [32], in addition to the domain and user models, exploits a task model to provide support to maintenance technicians. The system dynamically adjusts the sequence of setups, tests, and repair/replace procedures based on the information about the technician and the task he is trying to accomplish.

Background. This category covers a broad range of features defining the user's previous experience outside the system's core domain [33]. The most typical background features being used in adaptive systems include the user's education, profession, job responsibilities, and language abilities. Due to relatively small scope of background features, the background models are usually represented as stereotype models. For example, in the medical domain, users can be grouped by their background into three stereotypes: students, nurses, and doctors. Since background information does not change as dynamicly as the user interests or knowledge, it is typically provided explicitly: either by the user or by the administrative staff, manager, or supervisor.

Individual traits. The individual traits describe the user's personality and may include such information as the user's personality traits (e.g., introvert/extravert), cognitive styles (e.g., holist/serialist), cognitive factors (e.g., working memory capacity), and learning styles [33]. Similar to the user's background, the information about the user's individual traits is usually represented in stereotype models and entered explicitly based on results of some psychological test.

Context. The context-based adaptation has gained a high popularity in ubiquitous and mobile adaptive systems. These systems leverage a broad range of factors describing the environment and the situation the user is currently working in. The user's context may include such information as the user's platform, device, time, location, personal context (e.g., the user's blood pressure, pulse, mood), physical environment (e.g., light, temperature), social situation (e.g., meeting, lecture), and many others.

2.2. Web Portals

Web portals emerged in the late 1990s primarily as the gateways to different information resources available on the Internet. Companies like AOL, Altavista, and Yahoo used them to guide their user communities through the World Wide Web. The first portals provided users with Web directories, fulltext search capabilities, and certain communication services like email and chat. Later, portals gained special attention among enterprises as a platform to integrate not only the corporate information resources, but also the company's legacy systems. Nowadays, a large number of organizations use portals extensively as a single point of access to information, applications, and people.

2.2.1. What is a Web Portal?

A web portal is a Web application that provides users with a unified access to various information resources and services. As an example, let us have a look at My Yahoo![4] (Figure 2.3). The portal provides users with a wide spectrum of personal services, such as email, maps, latest news, financial information, weather forecast, entertainment and communication, and many more. Yahoo demonstrates all important properties of a web portal: single sign-on, consistent look-and-feel, and personalization. The *single sign-on* property enables users to log in just once and get access to all the different services and content without being prompted to provide the user name and password again and again. The *consistent look-and-feel* ensures that the users get a unified view of and access to all the applications available in the portal. Finally, the portal provides a *personalized service* that enables users to aggregate and arrange their favorite content and applications into a single page.

Yahoo Portal described above provides an example of a personal portal. However, there are a number of other types of portals. Saha [112] classifies portals according to the usage scenarios into five categories:

[4] http://my.yahoo.com

Figure 2.3.: My Yahoo! personal portal

- *Corporate portals* are used by companies to provide their employees with a single point of access to the corporate information and applications. Many corporate portals provide such services as content and knowledge management, business process management, business intelligence, and customer-relationship management.

- *Inter-enterprise portals* are used primarily as platforms allowing business partners to collaboratively coordinate such processes as supply-chain management, inventory, cash-flow, and logistics.

- *e-marketplace portals* are used as virtual markets supporting business-to-business (B2B) and business-to-customer (B2C) transactions. Online shops are one of the best known examples of e-marketplace portals. They support the entire process of purchasing products online, including product catalogs, shopping carts, payment, and product delivery. This category also covers

government portals, the platforms supporting the communication between the government and the citizens.

- *Personal portals* serve as personal gateways to the content and services on the World Wide Web. The best known examples of personal portals are Yahoo and Google. They provide in a highly personalized manner such services as Web directory, search, email, maps, weather forecast, news, collaboration, and entertainment.

- *ASP portals* are the online platforms supporting the business model of Application Service Providers (ASP). Such portals can be used, for instance, for renting a hosting service or an application.

2.2.2. Portal Technology

The Portlet Specification JSR 286 [104] has become one of the most widely used industry standards for the portal technology. The standard defines an Application Programming Interface (API) for programming portal applications in the Java programming language. According to the standard, the portal infrastructure consists of three major components: portlets, portlet container, and portal server.

A **portlet** is a pluggable user interface component that provides a specific piece of content or an application. Some of the examples of portlets are: a city map portlet, a portlet for flight booking, a car rental portlet, or a weather portlet. Portlets can be aggregated into a portal page (Figure 2.4) and can work together to support user's complex tasks, e.g., travel planning. Portlets are displayed in portlet windows. Depending on the portal configuration, a portlet window may also contain title and control buttons to configure, minimize, maximize, and close the portlet. Within a portal page, portlets can communicate with each other using events and public render parameters, which are handled by the portlet container. For instance, after the user has changed the destination city in the flight booking portlet, the portlet can send an event to other portlets on that page, e.g., to the weather portlet and to the city map portlet, notifying them that the destination city is changed. The recipient portlets can update their content respectively, e.g., the weather portlet will show weather forecast for the destination city, whereas a map portlet will display the map of that city.

A **portlet container** is the component that provides the execution environment for portlets and is responsible for managing the life cycle of portlet instances, i.e., their instantiation, initialization, use, and end of service. It is also responsible for handling the inter-portlet communication and storing the portlet preferences.

A **portal server** is a mediate component operating between the client and the portlet container. It is responsible for aggregating portlets into a portal page and submitting user requests from the portal page to the portlet container. The process

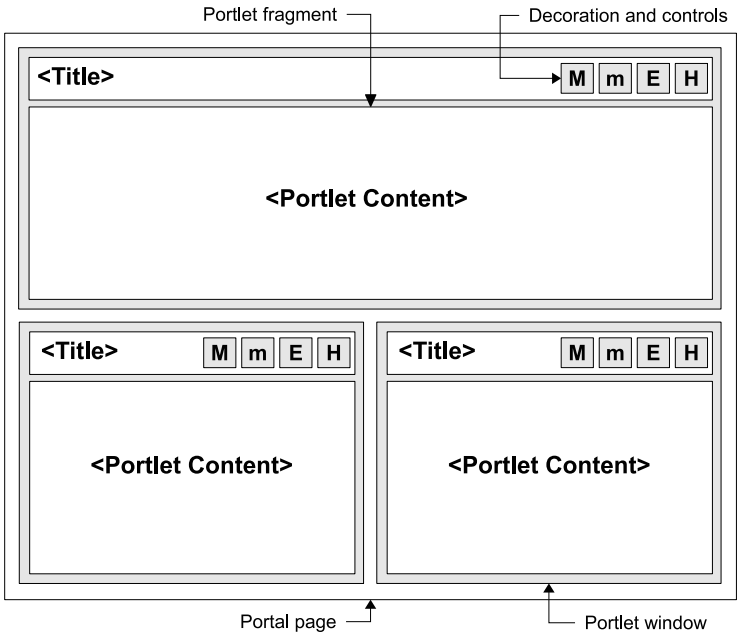


Figure 2.4.: Portal page (adapted from JSR 286 [104])

of creation of a portal page (shown in Figure 2.5) runs as follows. The portlet container generates individual portlet instances and hands them to the portal server. The portal server aggregates the portlets into a portal page and sends it to the client device, which displays the page in a browser.

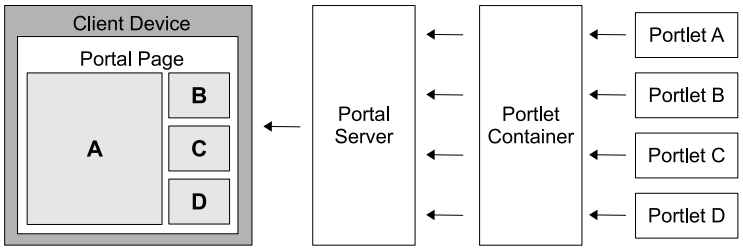


Figure 2.5.: Portal Page Creation (adapted from JSR 286 [104])

2.2.3. Personalization in Web Portals

In the late 1990s, with the advent of web portals, personalization received interest in the industry. Portals pioneered as one of the first commercial adopters of the personalization technology. Personalization features in the early web portals were

primarily aiming to support *adaptability*: The portals allowed the user-initiated customization of look-and-feel and arrangement of content. However, as the field of Adaptive Hypermedia was maturing, a number of proposals were made to introduce *adaptivity* in web portals to achieve the system-generated adaptation effects. This section describes both types of personalization in web portals.

Adaptability in Web Portals

In adaptable portals users are enabled to customize several parameters by choosing the preferred values [61] (p.489). Adaptability is a relatively simple type of personalization and is supported by most portals. E.g., the two leading personal portals, MyYahoo[5] and iGoogle[6], support adaptability. They allow users to modify the color scheme of portal pages and add, delete, rearrange, and configure portlets on a page. However, a web portal can support some more sophisticated adaptability features. For instance, Nauerz et al. [103] describe two adaptability methods in portals: user-adapted structures and user-created page flows. The first method allows the user to alter the portal's navigation topology. Through a special portlet the user can turn off/on visibility of portal pages, change the page order, and alter the hierarchy of pages. The second method enables the user to generate a so called pageflow, a sequence of pages that the user traverses frequently. The recorded pageflows can be called later and traversed by clicking "previous" and "next" buttons. Moreover, the user can exchange the recorded sequences with other portal users. According to the authors, such mechanisms can ease completion of tasks the user performs frequently.

Adaptivity in Web Portals

In an adaptive portal, the adaptation effects are initiated autonomously by the system without direct user intervention [61] (p.489). This type of adaptation requires the portal to have a user model representing, depending on the desired adaptation effects, such features as the user's interests, expertise, goals, background, or context. There are a number of adaptation effects that could be initiated by a web portal, namely dynamically adapted navigation topology, displaying links to relevant pages, personalizing search results, augmenting content with background information, and automatically rearranging portlets based on the user's task.

Nauerz et al. [103] describe two types of adaptation effects that can be initiated by a portal: system-adapted structures and system-created recommendations. The first adaptivity method alters the portal navigation topology based on the

[5] http://my.yahoo.com/
[6] http://www.google.com/ig

user's navigation history. The pages that the user visits frequently are being promoted to better navigation positions. A similar type of adaptation is provided by mPERSONA [106], a solution for personalized mobile portals. The second method described by Nauerz et al. is a less aggressive type of adaptivity: for a page requested by the user, the method determines other pages that the user is likely to navigate to and blends in the links to those pages in the portal's theme.

Another type of adaptivity in web portals was proposed in [102]. The authors describe an approach to automatically augmenting portal content with relevant documents and background information based on the user's current context and long-term interests. The proposed system extracts named entities (e.g., companies, locations, people, etc.) from the content requested by the user and compares the extracted entities with the user's long-term interests represented in her user model. Based on the entities that the user is interested in, the system compiles a list of the portal's internal and external resources that the user might be interested in. The identified resources are displayed by the portal either as side portlets or blended in the source text of the requested document as hyperlinks.

2.3. Social Semantic Web

The current generation of the World Wide Web (WWW) has been characterized by many as a Social Semantic Web [28, 62, 64]. This is where the two paradigms, the Social Web and the Semantic Web, complement each other in the way they approach content generation and organization [62]. The *Social Web*, also called Web 2.0, regards the WWW as a participation platform enabling users to communicate and collaborate with each other as well as to contribute content. The concept of *Semantic Web* was envisioned by Berners-Lee [23], who described the web where not only humans can cooperate, but also machines. In order to achieve this cooperation, he wrote, the information and services on the web should be given a well-defined meaning that can be understood by computers.

In the Social Semantic Web the principles of the two paradigms work together to achieve a better quality and richer functionality of the WWW. It exploits the willingness of users to generate content and metadata [95] (p.12) as well as the technology to represent the semantics of web entities in a well-defined way that can be understood by computers. The well-defined semantics of data on the web can increase the quality of search, enable better and richer personalization, achieve interoperability of different web applications, and enable automation of routine tasks.

2.3.1. User-Generated Content and Metadata

Unlike in the so called Web 1.0, the read only medium where only a limited number of users could contribute content, in the Social Web the users are provided with a wide range of collaboration and authoring possibilities. Users can express and share their thoughts and opinions on blogging and micro-blogging websites (e.g., LiveJournal[7] and Twitter[8]). They can also collaborate on generating large knowledge bases using the wiki technology (e.g., Wikipedia[9]). The Social Web provides a wide range of means for creating and sharing multimedia content, including photos, audio, and video (e.g., Flickr[10] and YouTube[11]). The set of technologies boosted by Web 2.0 enables users to even aggregate various types of content from multiple sources into new applications, called mashups (e.g., Yahoo Pipes[12]).

However, with the exponentially growing amount of content on the Web, the need for well-defined semantics of the produced content becomes even stronger. In such a dynamically evolving information space, not only machines need well-defined semantics of data, but also humans. Web users need some means to quickly locate content of their interest and to differentiate the good quality resources from the ones of bad quality. Fortunately, the paradigm of Social Web provides a solution for obtaining this semantics. It exploits the willing of users to generate metadata. The metadata on the Social Web can be generated by users in forms of tags, ratings, and comments.

Tags

In the Social Web, a tag is a keyword assigned (usually by a user) to a web resource or its part to describe its meaning or to indicate the category it belongs to. Usually tags are organized into collections, so called folksonomies [128]. Folksonomies have become a popular means for categorizing content and improving search and navigation. For instance, a social bookmarking system Delicious[13] uses a folksonomy for organizing users' bookmarks to resources on the Web. Users can describe the meaning of bookmarks with freely chosen tags. A single resource can be bookmarked by multiple users, which makes possible to determine most popular tags for a given resource based on the frequency of the tags applied to the resource. Also, it makes possible to determine the most popular resources for a given tag. Users can use tags not only to search within own bookmarks, but also to search bookmarks

[7] http://www.livejournal.com/
[8] http://www.twitter.com/
[9] http://www.wikipedia.org/
[10] http://www.flickr.com/
[11] http://www.youtube.com
[12] http://pipes.yahoo.com/
[13] http://www.delicious.com/

created by others. This can be done by means of so called tag clouds that display the most popular tags as hyperlinks of different font size indicating their popularity. A similar approach to tag-based search is used by many other community-enabled web systems, e.g., Flickr[14] and Technorati[15].

The phenomenal popularity of collaborative tagging in online communities can be partially explained by the ease and efficiency of the tagging process as well as by its utility function for the community. Unlike the taxonomy-based categorization, tagging does not require any training and understanding of some controlled vocabularies [90]. Users can quickly label resources in various systems using the keywords they like. With respect to the utility function of tagging, there are plenty of usage patterns for the community. Golder and Huberman [59] identified seven functions that tags perform for bookmarking systems, which they claim also apply to other tagging systems.

1. *Identifying what (or who) it is about.* Tags provide a very concise and understandable description of the content.

2. *Identifying what it is.* Tags can tell the type of medium, e.g., whether the medium is a news article, a photo, or a blog entry.

3. *Identifying who owns it.* Tags can identify the owner of content or the person who created it.

4. *Refining categories.* Some tags provide fine granularity or refinement of existing categories.

5. *Identifying qualities or characteristics.* Tags can carry some emotional characteristics of the tagged content or the persons assigned them. Such tags as scary, funny, stupid, etc. can reflect the users' opinions regarding the content.

6. *Self reference.* Tags can indicate ownership. Some users tag content with such tags as "mystuff" or "mycomments" to denote the relation of tags to the tagger.

7. *Task organizing.* Users can tag content with respect to their tasks, e.g., use "jobsearch" to tag the resources they think to be useful for the job search process.

In addition to the above-mentioned functions, tags can be used for enabling personalization and recommendation on websites. Nakamoto et al. [101] describe a recommender system that leverages a tag-based contextual collaborative filtering

[14] http://www.flickr.com/
[15] http://www.technorati.com/

approach to provide the user with in-context recommendations to relevant websites. The system, implemented as an extension for the Firefox browser, monitors the user's current browsing activity and displays a list of recommended websites in the browser's sidebar. The recommended websites are selected based on the user's tags she has previously used for bookmarking as well as based on the tags describing the currently displayed website that were given by other users in the bookmarking system. A number of other approaches to tag-based recommendations are described by Parra and Brusilovsky in [108].

Ratings and Comments.

Even though tags can carry some information about the user's personal feeling about the tagged content, their primary utility is to convey the content's semantics. To express the user's feeling and opinions, the Social Web applications provide two other mechanisms, namely rating and commenting.

Ratings. RateMyProfessors.com is one of the first sites providing rating mechanism [109]. The website allows anonymous users, typically students, to rate professors in American, Canadian and British schools with respect to a number of characteristics, such as easiness, helpfulness, clarity, appearance, and so on. Usually rating is done on some scale, e.g., on a scale of 1-5, yes and no, and so on. User ratings are used to rank resources with respect to their popularity or quality. For instance, YouTube exploits user ratings to identify the most popular videos, which the system promotes to better navigation positions.

Also ratings can be leveraged in collaborative filtering systems for recommendations [114]. For instance, in MovieLens[16] the user can rate movies on a 1-5 scale, where 1 - "awful" and 5 - "must see". Using the given ratings, the system identifies other users that gave similar ratings and based on the interests of the users sharing similar ratings the system recommends other movies. User ratings can be used in collaborative filtering (CF) systems in either of two methods: the classic CF method and item-to-item CF method [86]. The *classic CF* method represents a user as an N-dimensional vector, where N is the number of rated items and components of the vectors are the distinct ratings assigned by the user. The method then identifies other users whose vectors look similar and recommends the user the items that were rated positively by the users sharing the most similar rating vectors. The *item-to-item CF* method was first introduced by Amazon [85] and is known as "customers who bought this item also bought..." type of recommendation. Instead of matching the user's ratings to the ratings of other users, the item-to-item CF for a given item (a product that the user is viewing or hast just ordered) identifies the

[16] http://www.movielens.org/

users that rated the item positively and uses the other items that were positively rated by those users to issue item-to-item recommendations.

Comments. In addition to ratings, many websites also provide commenting features. Users can post their comments regarding a resource or person in free-text form. For instance, YouTube allows its users to leave short comments regarding the viewed videos. The system also allows the users to rate comments as poor or good and replying to comments, which sometime leads to discussions and debates involving many users. Amazon supports commenting features too. The customers are provided with the possibility to post reviews on the products available in the catalog and share their personal experience and feeling about products. Moreover, Amazon allows users to rate the product reviews as helpful or not helpful. The system exploits the ratings to generate lists of the most helpful reviews, which are displayed along with the product description as an additional source of information to help users making their decisions.

Content of comments and their character can vary depending on the type of content being commented on. Park et al. [107] identified four categories of comments and investigated the correlation between the nature of content being commented on and the type of comments. The four types of comments are: summary, additional information, opinion, and noise. The authors used this categorization to analyze the content of comments on four social services: bookmarking service del.icio.us, community-enabled online shopping website Amazon.com, video sharing service YouTube.com, and content sharing website Digg.com. They describe the types of comments and explain the found correlations as follows:

- *Summary* is a concise description of the source content, a short statement revealing the content's gist. The authors found that the comments of this type dominate on del.icio.us, which indicates that the users of the bookmarking service have tendency to summarize the source content, rather than discuss it.

- *Additional information* can be supplied to the source content to help its understanding, e.g., through a detailed description, similar or related content. The study found that the amount of the comments of this type is considerably higher on Amazon.com that indicates the preference of the Amazon's customers for rather detailed and comprehensive product reviews.

- *Opinion* is a statement reflecting the user's personal and subjective attitude towards a document, which could be an opinion, evaluation, speculation, or assessment. The study revealed a relatively high proportion of the comments of this type on Digg.com and Amazon.com.

- *Noise* comments include the low-quality or irrelevant statements, such as jokes, spam, and sarcasm. The authors found considerably high proportion of the comments of this type on YouTube (about 64% of all comments). According to the authors, majority of the comments made on the video sharing service are jokes, very short statements of emotion or thoughts.

Though the original application of commenting was to provide users with additional information on the source content and let them share their opinions, over time it has received a number of other applications. User comments can be exploited for extracting opinions and generating ontologies [52], identifying attitude towards companies or distinct products [98].

2.3.2. Knowledge Representation

One of the main ideas of the Semantic Web is to achieve semantic representation of the resources on the web that could be understood by computers [23]. This requires that the entities on the web (resources, users, relationships, etc.) must be supplemented with additional description (metadata) represented in a machine-processable way.

Metadata

Resource Description Framework (RDF)[17] has become the most widely used language for representing machine-processable metadata of resources on the web. The main idea of RDF is that (1) resources are uniquely identified by *Unified Resource Identifiers* (URIs) and (2) described in terms of properties and property values. The example shown in Figure 2.6 represents an RDF-based description of a document and its creator.

There are two most common serialization formats to represent RDF data: RDF/XML and Notation 3. RDF/XML[18] represents RDF data using the syntax of eXtensible Markup Language (XML)[19]. Notation3[20], also called N3, was designed with human-readability in mind and provides a more compact representation of RDF data. Listing 2.1 and Listing 2.2 illustrate two examples of serialization of the RDF data shown in Figure 2.6 in RDF/XML and N3 formats respectively.

RDF provides a simple and efficient mechanism to describe resources on the Web. However, it can also be used for defining shared vocabularies as well as for representing knowledge in a certain domain by defining classes hierarchies and

[17] http://www.w3.org/RDF/
[18] http://www.w3.org/TR/REC-rdf-syntax/
[19] http://www.w3.org/XML/
[20] http://www.w3.org/DesignIssues/Notation3

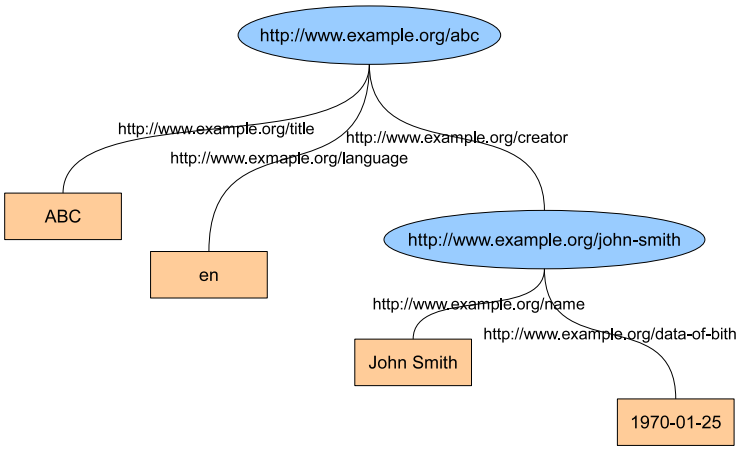


Figure 2.6.: Sample RDF data

relations among classes. This knowledge is usually represented in ontologies and used to enable computers to reason about the domain.

```
<?xml version="1.0"?>

<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:ex="http://www.example.org/#">

  <rdf:Description rdf:about="http://www.example.org/abc">
    <ex:title>ABC</ex:title>
    <ex:language>en</ex:language
    <ex:creator rdf:resource="http://www.example.org/john-smith"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://www.example.org/john-smith">
    <ex:name>John Smith</ex:name>
    <ex:date-of-bith>1970-01-25</ex:date-of-bith>
  </rdf:Description>

</rdf:RDF>
```

Listing 2.1: Resource description in RDF/XML

```
@prefix dc: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix dc: <http://www.example.org/#>.

<http://www.example.org/abc>
  ex:title "ABC";
  ex:language "en";
  ex:creator <http://www.example.org/john-smith>.
```

```
<http://www.example.org/john-smith>
  ex:name "John Smith";
  ex:date-of-bith "1970-01-25".
```

Listing 2.2: Resource description in N3

Ontologies

In computer science, the concept of ontologies is used to describe a mechanism for defining relationships between terms in a machine-processable way. One of the most cited definitions of ontologies is the one coined by Gruber [63], who defined ontology as: *"an explicit specification of conceptualization"*. This definition was later modified by Borst [25] and Studer et. al [121] as follows: *"An ontology is a formal, explicit specification of a shared conceptualization. Conceptualization refers to an abstract model of some phenomenon in the world by having identified the relevant concepts of that phenomenon. Explicit means that the type of concepts used, and the constraints on their use are explicitly defined. Formal refers to the fact that the ontology should be machine-readable. Shared reflects the notion that an ontology captures consensual knowledge, that is, it is not private of some individual, but accepted by a group."*

Ontologies describe a domain using individuals (instances), classes (concepts), attributes, and relationships. Individuals are the basic components of an ontology. These are concrete instances of classes. For example, John Smith is an instance of class person. *Classes* are the sets of objects that share the same characteristics. *Attributes* are the characteristics of classes. And finally, *relationships* are the associations between classes. There are three types of relationships: *subsumption* defines objects as members of a class, *hierarchy* uses child and parent-of relationships to build hierarchy of classes, and *meronymy* relationship describes how objects are combined together to build other objects.

Types of ontologies. Ontologies can be classified in two ways: according to the amount and type of structure of the conceptualization and the subject of conceptualization [60]. Under the first categorization type, the following types of ontologies are distinguished: controlled vocabularies, glossaries, thesauri, informal is-a hierarchies, formal is-a hierarchies, frames, ontologies that express value restrictions, and ontologies that express general logical constraints. In the second category, which distinguishes ontologies by the subject of conceptualization, the following types of ontologies are present:

- *Knowledge representation ontologies* define the representation constructs that can be used to formalize knowledge according to a particular knowledge representation paradigm [127].

- *General ontologies* define primitives that can be used to represent common sense knowledge, which can be reused across different domains [127].
- *Upper-level ontologies* define very general concepts, to which all root elements in existing ontologies can be grounded [60].
- *Domain ontologies* define concepts and relationships that can be used as vocabularies in a particular domain of knowledge, for example medicine, engineering, law, etc [97].
- *Task ontologies* define vocabularies that are specific to a generic task or activity, for example, diagnosing, scheduling, selling, etc [97].
- *Application ontologies* define concepts and relations that are required to model knowledge in a particular application [127].

Formalization languages. Several languages and notations have been proposed for formalizing ontologies, namely Ontology Inference Layer (OIL) [57], DARPA Agent Markup Language (DAML)[21], Web Service Modeling Language (WSML)[22], RDF Schema[23], and Web Ontology Language (OWL)[24]. However, the most dominating languages for representing knowledge on the Web are RDF Schema and OWL.

- *RDF Schema*, also called RDFS, is a W3C recommendation and specification for defining vocabularies and simple ontologies. RDFS provide means to define classes, class properties, and organize classes into class hierarchies.
- *Web Ontology Language* is a more expressive language for representing ontologies. Similar to RDFS, OWL provides means to define, classes, properties, and relations. However, it has a number of additional capabilities: It allows defining instances of classes, relations of different types, restrictions, and axioms.

[21] http://www.daml.org/
[22] http://www.wsmo.org/wsml/
[23] http://www.w3.org/TR/rdf-schema/
[24] http://www.w3.org/TR/owl-features/

CHAPTER 3

Requirements Analysis

In this chapter, we elaborate on functional and qualitative requirements for scrutable adaptivity in community-enabled web portals. We ground these requirements on the foundations of adaptive systems, web portal technology, and social semantic web that we described in Chapter 2. Moreover, we explain the importance of these requirements using the challenges reported in Section 1.1, namely the usability and trust problems of adaptive systems, issues related to the correctness and completeness of personalization models, and obstacles of scrutable adaptivity.

3.1. Functional Requirements

As it was mentioned in Section 2.1, there are four essential components that an adaptive system must have to personalize the content, presentation, and navigation to needs of individual users. According to Brusilovsky and Millàn [33], an adaptive system must have a *user model*. This model provides essential information about users, e.g., their interests, knowledge, goals, etc. It is also necessary that the content of the system is complemented with *metadata* [94] providing the semantics of content (see Section 2.3.1). To achieve automatic selection of documents that match user interests and goals, both the user model and the metadata must use the same vocabulary. This vocabulary must define essential concepts of the *domain knowledge*. As it was mentioned in Section 2.3.2, the vocabulary must be formalized in a language capable of representing rich semantic information about the domain. Finally, an adaptive system must have *personalization rules* that define what *personalization effects* must be made for the user on a given resource.

In this section, we identify functional requirements for scrutable adaptivity in community-enabled web portals (further *framework*) with respect to the four key components of adaptive systems, namely user model, domain knowledge, metadata,

and personalization rules. Also, we elaborate on the requirements that apply to the methods of displaying personalization effects to portal users. For each functional requirement, we provide a list of challenges from Section 1.1 that the requirement addresses.

3.1.1. Requirements to the User Model

It has been widely argued in the literature [2, 33, 51, 56, 76] that for representation of user features, such as interests or knowledge, semantic user models provide a number of significant advantages. Unlike vector user models, the concepts in semantic user models can be interconnected through various relations. These relations can be leveraged to compensate information scarcity in user models [33]. For instance, they can be used for propagating interest from one item to another. Hence,

Requirement F1: Semantic representation of user model. The user model of the framework must be semantically enhanced. It must provide information about semantic classs of entities and be able to represent semantic relations among entities. The user model must be represented in a formalism that can be interpreted by the framework and used for reasoning. It must be able to propagate user interest and knowledge using the semantic relations among items.

Challenges addressed:

- Challenge 6: Complete and accurate user model

One of the most frequent critiques of adaptive systems is that users have no control over their user models. This negatively affects the usability of adaptive systems. Hiding user models from the users decreases the system's transparency and predictability [70]. Users do not know what information the system uses for personalization. It also hinders them from anticipating the system's behavior and from bringing it in the state they want. Apart from that, preventing users from accessing and editing the user model can cause that the model becomes incomplete or contains wrong assumptions. This can lead to wrong personalization effects and inappropriate adaptive behavior. Finally, users' right to access the personal information that the system collects about them is protected by law in many countries [80, 117]. For instance, according to the German Act on the Protection of Personal Data Used in Teleservices (Gesetz über den Datenschutz bei Telediensten),

> "§ 7 User's right to information
> The user shall be entitled at any time to inspect, free of charge, stored

> data concerning his person or his pseudonym at the provider's. The information shall be given electronically if so requested by the user." [1]

Hence,

Requirement F2: User access to user model. The framework must provide the user with full control over the user model. It should enable the user to get an overview of the entire model and view detailed information about the model attributes. Also, it should empower the user to add new information and alter existing items.

Challenges addressed:
- Challenge 1: Predictability and transparency
- Challenge 2: Controllability
- Challenge 4: Privacy
- Challenge 6: Complete and accurate user model

The completeness and correctness of information represented in the user model is an essential requirement for generating appropriate personalization effects. However, user interests, knowledge, and goals may change over time. In the context of large information portals with frequently updated content it is burdensome and time-consuming for users to keep the entire model up-to-date and as complete as possible. Hence,

Requirement F3: Automatic update of user model. The framework must be able to generate user models automatically in an unobtrusive manner. For this purpose, the framework must be able to leverage the users' interaction in the portal, such as the documents they read, bookmark, rate, or comment on.

Challenges addressed:
- Challenge 6: Complete and accurate user model

According to Cook and Kay [48], it is important that the user model contains a justification part. In this part the system should explain to the user the system's reasoning about the user model. It should explain what information the system has used for making the assumptions it stores in the user model. For instance, it should explain the user why it believes the user is interested in or knowledgeable about the items the user model stores. This is important for making the system's behavior more transparent, comprehensible, and predictable for the user. Hence,

Requirement F4: Explain sources of information in user model. The framework must explain users the provenance of information in the user model. They must be able to view all the sources based on which the system deduced about user features.

Challenges addressed:
- Challenge 1: Predictability and transparency
- Challenge 4: Privacy

3.1.2. Requirements to the Domain Knowledge

Domain knowledge models have become an essential component for a broad range of adaptive systems. They represent entities of the domain of interest in a machine understandable formalism, oftentimes in form of an ontology. A number of adaptive systems use domain knowledge models as a semantic vocabulary to model user interests [2] and knowledge [6, 105]. As it was mentioned in Section 3.1.1, the semantic relations captured in the domain knowledge model can be used by the user model to propagate user interest or knowledge from one entity to another. Also, these models can provide such information as the semantic class an entity belongs to [2]. This information can be leveraged for disambiguating different entities having the same name. For example, it allows disambiguating the Apple (company) from an apple (fruit). Hence,

Requirement F5: Semantic representation of domain knowledge. The framework must represent domain knowledge in a formal and machine processable format. It must enable other components, namely, the user model and metadata management component, to perform reasoning over the domain knowledge. The domain knowledge model must represent semantic relations among entities. It must also represent the semantic class of entities. Additionally, it must be able to disambiguate entities having the same name.

Challenges addressed:
- Challenge 6: Complete and accurate user model
- Challenge 8: Timely and accurate annotation of resources

Complete and accurate information in domain knowledge models is an essential requirement for accurate reasoning over the domain knowledge. Oftentimes, domain knowledge models are built by a group of experts and knowledge engineers [60]. However, the process of building ontologies by experts is time-consuming and

costly. In the context of large information portals with constantly changing content, maintaining domain knowledge model up-to-date and complete is especially difficult. To address this challenge a number of approaches have been proposed for automatic domain knowledge generation. Lonsdaleet et al. [87] proposed a method for generating domain ontologies from text documents. Agirre et al. [3] describe an approach to enriching large ontologies using freely available documents on the web. Kietz et al. report in [78] on a method for generating domain ontologies based on documents from corporate intranets. Hence,

Requirement F6: Automatic generation of domain knowledge. The framework must provide a method for automatic update of the domain knowledge model. It must leverage the content of portal documents to extract relevant entities and insert them into the domain model. It must be able to extract the semantic class of entities and where applicable semantic relations between entities.

Challenges addressed:

- Challenge 7: Complete and accurate domain knowledge

Also, a number of approaches have been proposed for enabling the user community to access and contribute to the domain knowledge. Braun et al. [27] report on a collaborative approach to ontology engineering. This method integrates user-driven tagging of web content and folksonomies with heavy-weight formal ontologies. It shows that given necessary tools it is possible to motivate the user community to contribute to the domain knowledge. Hence,

Requirement F7: User community access to domain knowledge model. The framework must enable the user community to access the domain knowledge and contribute to it. Users must be enabled to create new entities and define relations between existing entities. Since different users might have different world views, the framework must allow users to create their own subset of domain knowledge. The system must be able to leverage the individual knowledge fragments to update the shared knowledge model. It must also protect the knowledge from vandalism and unintentional damage by users.

Challenges addressed:

- Challenge 1: Predictability and transparency
- Challenge 2: Controllability
- Challenge 7: Complete and accurate domain knowledge

3.1.3. Requirements to Metadata

Metadata plays an important role in adaptive systems. It conveys the machine-processable meaning of the system's content. Adaptive systems use metadata to retrieve content that match user needs. Nowadays, the most common approach to providing metadata is to annotate content with a flat list of keywords, also called tags. However, this approach has a number of drawbacks [83]. First, synonyms cause low recall problem because documents can be annotated with different spellings of the same concept. For example, documents about the United States of America, can be annotated with different keywords having the same meaning, such as *US*, *USA*, and *United States*. Second, polysems, words having multiple meanings, can cause the problem of low precision. For instance, for a search query *apple* the system might retrieve documents about apples (fruit) and documents about the Apple (company). Third, flat lists of keywords do not allow recommending related content, which is especially important in adaptive systems. For instance, for someone interested in pasta, the system should be able to retrieve content not only about pasta, but also about spaghetti and farfalle, because they are related to pasta. Hence,

Requirement F8: Semantic representation of metadata. The framework must provide semantically-enriched metadata of content. The semantic representation of metadata must allow the framework to deal with the problems of synonyms and polysems. It should also allow retrieving relevant content using semantic relations between concepts used for annotation of content.

Challenges addressed:

- Challenge 8: Timely and accurate annotation of resources

In large community-enabled portals, it is almost impossible to foresee the dynamics of content update. New content can be contributed by the user community on hourly basis. However, it is essential that newly added content is annotated with metadata as soon as possible, so that the personalization and search engines can leverage it for adaptation and retrieval. A large number of approaches and tools have been proposed for automatic generation of semantically rich metadata. Open Calais[1] is a web service that allows extracting semantic entities from text documents. It extracts entities of such types as company, person, technology, product, etc. Also, it is able to identify relations between entities, such as between a company and an employee, a product, or technology. Hence,

[1] http://www.opencalais.com/

Requirement F9: Automatic generation of metadata. The framework must provide mechanisms for automatic annotation of portal content. At the time when a new document is added to the portal, the framework must generate semantic metadata for the document and store it in the system.

Challenges addressed:
- Challenge 8: Timely and accurate annotation of resources

However, the automatically generated metadata do not always represent the entire meaning of the document or sometimes may even convey incorrect meaning. Hence,

Requirement F10: User community access to metadata. The framework must allow the user community to access and edit metadata of portal content. It must allow supplementing the automatic annotations with user-generated metadata. Graphical user interfaces for annotating portal resources must be intuitive and easy to use.

Challenges addressed:
- Challenge 1: Predictability and transparency
- Challenge 2: Controllability
- Challenge 8: Timely and accurate annotation of resources

3.1.4. Requirements to Personalization Rules and Effects

As it was described in Section 2.2, portal pages may consist of one or several portlets. A portlet may provide a static document or dynamically generated content of an application. Personalization effects occurring in portlets may vary from portlet to portlet. For instance, a portlet displaying search results may sort the hits by the relevance to the user interest profile. A portlet that displays news stories may highlight the most interesting for the user stories by color. Moreover, a portlet may support multiple personalization effects. Hence,

Requirement F11: Portlet-specific personalization rules. The framework must support defining personalization rules for individual portlets. It must allow portlet application developers to describe the personalization rules supported by the portlet in a formal way. This description should enable portlet instances to generate personalized markup according to the defined personalization rules.

Challenges addressed:
- Challenge 14: Portlet-level personalization

One of the most common critiques of adaptive systems is the violation of the transparency principle [70]. Transparency refers to the extent to which system's status and actions are visible and understandable. In many adaptive systems, it is not always clear for the user what content is personalized and how it is personalized. This, however, may negatively impact the user experience. Hence,

Requirement F12: Explanation of personalization. The framework must provide users a comprehensive overview of personalization taking place on portal pages. It must explain users what elements are adapted and based on what information the adaptation is done. The user must be able to know at all time whether any page element is personalized.

Challenges addressed:
- Challenge 1: Predictability and transparency
- Challenge 10: Lack of awareness of scrutability

Another serious problem that adaptive systems have been criticized for is the violation of the controllability principle [70]. Controllability is the property that allows users to bring the system in the state they want. In many adaptive systems, adaptation takes place fully automatically and users have no control over it. This may cause that users do not get relevant content or are provided with unwanted information. Hence,

Requirement F13: Turning personalization on and off. The framework must provide users full control over personalization in the portal. It should enable users to turn personalization on and off. Also, it is important to take into account the distinctive features of web portals, i.e., the user should be able to switch personalization on and off at the level of individual portlets.

Challenges addressed:
- Challenge 2: Controllability
- Challenge 3: Obtrusiveness of adaptive behavior
- Challenge 5: Breadth of experience

Requirement F14: User access to personalization rules. In order to ensure that adaptation effects are suitable to the user and provided in appropriate way, the system should enable users to access personalization

rules. Since personalization effects vary from portlet to portlet, users should be able to customize personalization rules at the level of individual portlets.

Challenges addressed:
- Challenge 1: Predictability and transparency
- Challenge 2: Controllability

Personalization effects are generated based on information from the user model, metadata, semantic description of domain knowledge, and personalization rules. Since users must be provided with edit access to all of these four components, it is essential that they also understand how changes in one model affect the personalization on the portal page. It is an important requirement to ensure that the user can predict the system's actions [70]. Hence,

Requirement F15: Explain implications of altering models. To ensure predictability of the adaptive behavior, the framework must provide users with an understandable explanation of how a change in one of the adaptation models influences the personalization effects on a portal page. For instance, it should explain how a change in the user model will alter the portal content or the way it is displayed.

Challenges addressed:
- Challenge 1: Predictability and transparency

3.2. Qualitative Requirements

We believe that making adaptivity scrutable in compliance to the functional requirements defined in Section 3.1 can improve the system's transparency, predictability, and controllability. We also believe that scrutable adaptive systems can provide users a better control over their privacy. Additionally, we claim that by allowing users to contribute to user model, domain knowledge, metadata, and personalization rules can improve the quality of personalization. However, introducing scrutability to an adaptive system has a number of challenges, which impose a number of qualitative requirements. In this section, we elaborate on four qualitative requirements. Similarly to the functional requirements, for each qualitative requirement we provide a list of challenges from Section 1.1 that the requirement addresses.

Nowadays, an overwhelming majority of adaptive systems still keep adaptation models hidden from users. Scrutability has not yet become an integral function of

all adaptive systems. Therefore, average users have not developed a mental model of affordance of controlling adaptation [50]. Hence,

Requirement Q1: Findability of scrutinizing tools. The user should be aware of the existence of the tools for scrutinizing adaptivity and should be able to easily and quickly find them. The framework must place the entry points to the scrutinizing interfaces in the most possible proximity to the adapted fragments, so that the user could easily enter the parts of adaptation models that were used for the personalization and alter them.

Challenges addressed:

- Challenge 10: Lack of awareness of scrutability

Different users might wish different degrees of scrutability. Some users might scrutinize often, some less often, some might not be willing to scrutinize at all. Hence,

Requirement Q2: Unobtrusiveness of scrutinizing tools. The system must ensure that those users who are willing to contribute to adaptation models and control personalization can always find the corresponding scrutinizing tools. But it should also ensure unobtrusiveness of scrutinizing tools: those users who do not want to have explanation of adaptation effects and/or access the personalization models should not be distracted by the scrutinizing tools.

Challenges addressed:

- Challenge 13: Obtrusiveness of scrutability

Depending on the complexity of mechanisms used for adaptation and the structure of information stored in adaptation models, e.g., in the user model, it might be difficult for average users to understand how the adaptation works and to customize it. Since users might not use the tools for controlling adaptation often enough, they might not be willing to spend time on reading manuals or looking up into the help system. Hence,

Requirement Q3: Usability of scrutinizing tools. The framework must ensure good usability of tools for controlling personalization. It should ensure that average users are able to use the scrutinizing tools easily and efficiently without preliminary training.

Challenges addressed:

- Challenge 10: Lack of awareness of scrutability
- Challenge 12: Complexity of adaptation

Altering adaptation models and adjusting personalization effects is not the main task users want to accomplish when they use an adaptive system. Therefore, users might not be motivated enough to scrutinize. Hence,

Requirement Q4: Foster curiosity and engage users. In order to motivate users to use the scrutinizing tools and contribute to the adaptation models, the system should foster curiosity by enabling users to explore their user models, the domain knowledge model, personalization rules and seeing the effects of altering information in these models. The system should also engage users by making the interaction pleasurable and fun.

Challenges addressed:

- Challenge 11: Lack of motivation to scrutinize

3.3. Summary

In this chapter, we elaborated on fifteen functional and four qualitative requirements for scrutable adaptivity in community-enabled web portals. These requirements were identified based on the foundations of adaptive systems (Section 2.1), portal technology (Section 2.2), social semantic web (Section 2.3), and the challenges reported in Section 1.1, namely usability and privacy challenges of adaptive systems, challenges related to completeness and correctness of adaptation models, and challenges of scrutable adaptivity. Figure 3.1 provides an overview of the identified requirements with respect to the challenges they address.

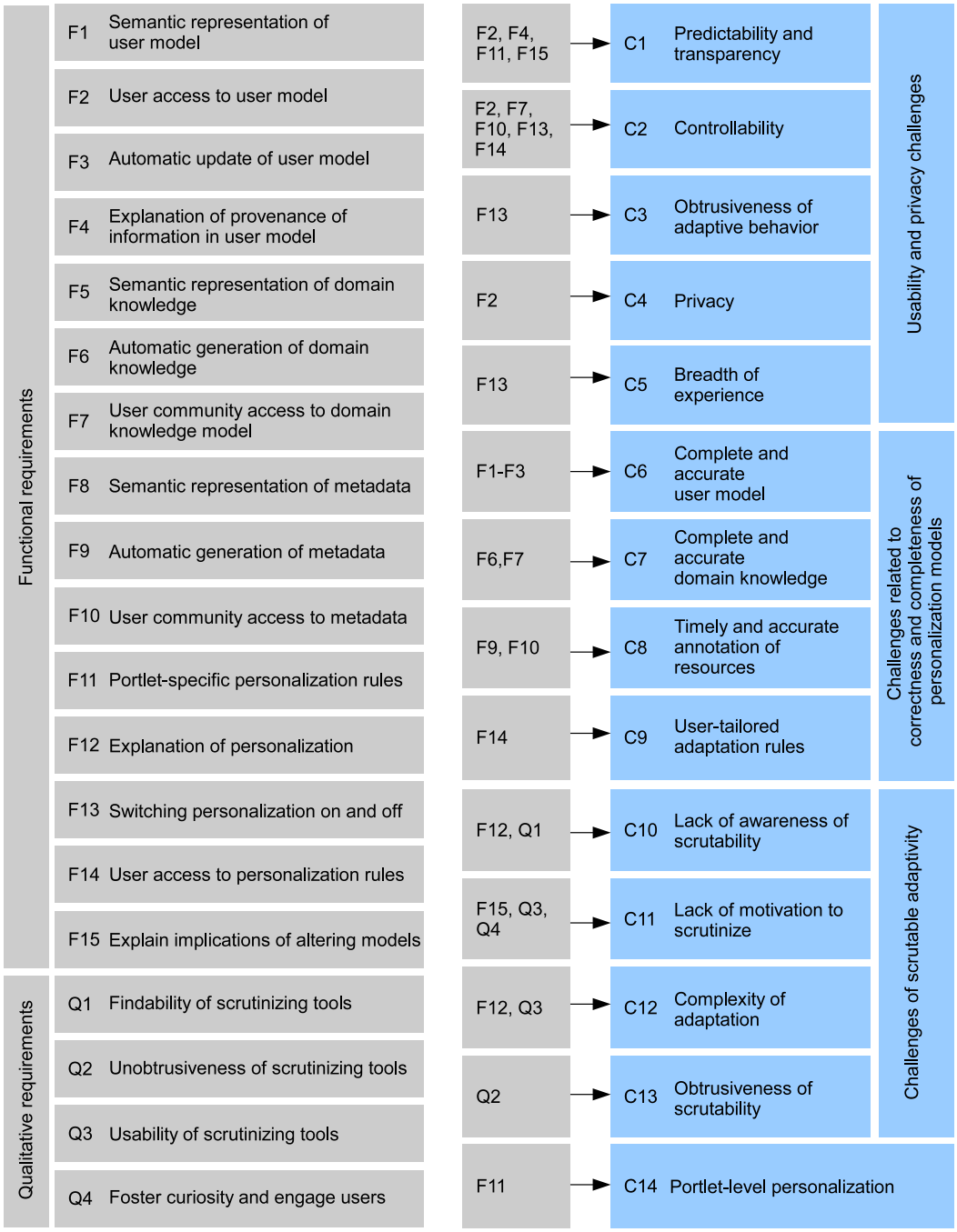


Figure 3.1.: Requirements for scrutable adaptivity in community-enabled web portals with respect to the challenges they address

CHAPTER 4

State of the Art in Scrutable Adaptivity

In the Introduction chapter, we elaborated on the need for scrutability of adaptation mechanisms and models in adaptive systems in general. In the Requirements Analysis chapter, we described the key components for adaptation in community-enabled web portals. Also, we identified requirements for making the adaptivity in community-enabled portals scrutable. In particular, we claim that the portal must provide users full control over the **user model** representing users' personal information that the system collects and uses for adaptation. It must allow users to provide *metadata* to portal documents and allow the user community contribute to the portal's *domain knowledge model*, representing the semantic description of the domain entities used in the user model and metadata. Also, the system must provide users access to *personalization rules* and allow adjusting these rules to their needs. Finally, the system must provide users some sort of an **overview and explanation of adaptation effects** and comprehensible and effective means to **control the adaptation effects** and if desired turn off the adaptation features.

This chapter provides an overview of the state of the art in scrutable adaptivity. We structured this overview according to the four components required for adaptation, namely user model, domain knowledge, metadata, and adaptation rules. Section 4.1 describes approaches to scrutable user modeling. In Section 4.2 we elaborate on research proposals for revealing domain knowledge models to the user community. In Section 4.3 we report on adaptive systems that provide users with access to metadata representing the semantics of documents available in the system. Section 4.4 provides an overview of two categories of adaptive systems: systems with explicitly defined adaptation rules and systems where adaptation rules are inferred by automatically based on the user's interaction with the system. Additionally, in

Section 4.5 we report on approaches to summarizing and explaining adaptation effects to users and methods for providing user control over personalization. Finally, in Section 4.6, we provide a comprehensive comparison of the described approaches with respect to the requirements identified in Chapter 3.

4.1. Scrutable User Models

A number of approaches have been proposed to scrutable user modeling. They vary by the type of information being modeled (e.g., interests or knowledge), sources from which the user features are inferred (e.g., visited web pages or the user's interaction behavior), the form of presenting the modeled information, level of access and control, etc. Bull and Kay [36] proposed a framework for describing open learner models. We generalize this framework to describe the approaches to scrutable user modeling across different domains. The framework defines eleven dimensions of opening learner models and groups them into three categories, namely, *what is available*, *how is the model presented*, and *who controls access*.

What is available?

1. ***Extent of model accessible*** defines the level of the model's openness to the user. More specifically, it defines what attributes and to what extent the user can control. Depending on the purpose, the user might be provided either with a complete or partial access to the model. In the area of adaptive tutoring systems, a common approach to revealing user models is to present the user with a summary view, which only *partially* reflects the content of the entire model. For instance, the APTList Tutor [49] provides the learner with a skill meter showing the list of learning goals and the progress the learner has already made with respect to the goals. Partial access to the user model is also provided by the NetCoach adaptive tutoring system [130]. However, a few systems attempt to reveal the *complete* model. Ahn et al. [5] describe an open user interest model for the YourNews adaptive news system. Unlike the two previously-mentioned approaches, the user in YourNews can view the entire collection of interests identified by the system. Similar extent of access to user models is provided in SASY [50], um toolkit [73], and PeerGlass architecture [79].

2. ***Match underlying presentation*** specifies how close the user model is to the underlying representation and structure of the domain knowledge. For instance, Dimitrova [55] and Zapata-Rivera et al. [133] describe approaches that represent the user model in a way that entirely reflects the structure of

the underlying knowledge model. The former represents the learner model in a form of visual graphs and the latter displays it with Bayesian Networks.

3. ***Access to uncertainty*** determines whether the system represents the user's information with uncertainty and whether the degree of uncertainty can be viewed by the user. For instance, Kliger [79] describes a number of visual approaches to represent uncertainty of user interests identified by the system based on the news stories previously read by the user. User reading interests are represented as keywords on a two-dimensional surface (Figure 4.1), where one of the axes defines the system's confidence. A similar approach to representing uncertainty is proposed by Uther and Kay [126]. They describe a system for visualization of large user models, VlUM, which also displays user models on a two-dimensional space, where the offset on x-axis denotes the system's certainty.

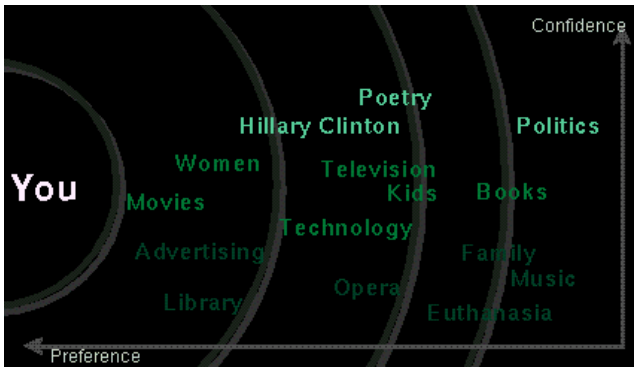


Figure 4.1.: A visual approach to representing the system's confidence by Kliger [79]

4. ***Role of time*** determines whether the user can access the current state of the model, obtain a retrospective view, or can be provided with a set of predicted states. To our knowledge, none of the proposed scrutable user models provides a view on the state of the entire model in the past or predicts the model's state in the future.

5. ***Access to sources of input*** defines whether users can view the sources from which the system made its assumptions about them. This feature has become a common property of many scrutable models. For examples, the um_view interface [48] has a so called 'justify' view that displays the source from which the system concluded about the user's knowledge on the selected item. Similarly, the Mneme architecture [74] allows users to view the evidence

Figure 4.2.: An approach to representing the model's effect on personalization described by Tsandilas and Schraefel [124]

of information stored in the user model. Moreover, it enables users to add new sources from which the system can infer certain facts about users.

6. ***Access to model effect on personalization*** determines whether the user can see the effects of changing the attributes in her user model on the personalization. Such capability is provided by SASY [50]. In the system, users are enabled to change preferences in the user profile and see how it influences the personalization effects, i.e., adapted content on the page. A similar interaction style with user models is provided in the adaptive hypermedia system described by Tsandilas and Schraefel [124], which uses the stretchtext technique to adapt hypermedia content based on the user interests. Upon a user change of interest, the system stretches or shrinks the page fragments that contain information relevant to the interest being altered (Figure 4.2).

How is the model presented?

7. ***Presentation*** specifies the form of presenting the modeled information to the user, i.e., whether this information is presented in a textual form or visualized in some way. For example, in SASY [50], the user model is represented in a tabular form containing a list of rather verbose questions, their descriptions, and answer options. Though this is a simple way to represent the user information, it is not suitable for displaying large and complex models. For presentation of large and complex user models, graphical formats are more appropriate. A number of approaches have been proposed to visualizing user models. PeerGlass [79] demonstrates a visual method to exploring user models through a Rolodex of model planes. The um_view interface [48] allows

traversing through user models by expanding the tree of leaves and viewing detailed information about the items in the model. VlUM [126] is capable to visualize large user models and enables users to get an overview of the whole model as well as to view a subset of related beliefs in the model. The approaches described in [55] and [133] visualize learner models using concept graphs and Bayesian Networks respectively.

8. ***Access method*** specifies the degree of user's control on the modeled information. VlUM [126] (Figure 4.3) and its successor SIV [75] provide an example of *inspectable* user models. Both systems allow the user to view the model, navigate through it, and obtain some additional information about the items present. Neither system allows explicitly making any change in the model. A similar access method is provided in PeerGlass [79]. In *editable* user models, the user can view and alter attribute values. In SASY [50] users can view the attributes that the system has inferred about them and edit values of those attributes. Another class of models allow users *adding* new items. For instance, the user model editor of YourNews [5] allows users adding new interests into their models by typing the corresponding keywords in a string field. Contrarywise, Amazon[1] supports primarily *removal* of items from the user model. Amazon's user model is basically represented as a collection of the user's purchases, viewed products, ratings, and searches, which altogether are used for issuing recommendations. Users can view the collected information and delete items that they do not want the system to store and use for recommendations. Finally, access to user models can be provided in a form of *negotiation*. Users can negotiate with the system about the values of attributes in the user model [39, 77, 55].

9. ***Flexibility of access*** determines whether the modeled information can be presented in multiple formats and whether users can control the level of details. For instance, Bull and McKay [38] describe an approach to representing the learner model in multiple formats. More specifically, the authors elaborate on a learning environment for children, in which the learner model can be presented in two forms: children's view and teacher's view. The children's view is adapted in a way that young users can better understand. E.g., the view does not show misconceptions.

Who controls access?

10. ***Access initiative*** determines who initiates access to the model. In most scrutable user models, the *user* is able to initiate access to the user model.

[1] http://www.amazon.com

Figure 4.3.: VlUM by Uther and Kay [126]

However, in some systems, other users can initiate the access, too. For instance, in a learning environment students can initiate access to the learner models of their *peers* [35] as well as the *instructor* [38] can initiate access to the student models.

11. ***Control over accessibility*** determines whether the user can share the model with other users or systems. UMPTEEN [37] provides such control. Students can make their learner models available for inspection by peers and instructors.

4.2. Open Domain Knowledge Models

According to Brusilovsky [30], the domain model is the "heart" of the knowledge-based approach to developing adaptive hypermedia systems. A domain model repre-

sents a set of small knowledge elements, called (depending on the system) concepts, knowledge items, topics, knowledge elements, or learning objects. These knowledge elements are used in adaptive systems for representing different features of users, e.g., their knowledge, interests, or tasks, etc. They are also used for defining the semantics of the system's content (web pages, learning objects, products, etc.) and personalization rules in a way that can be understood not only by humans, but also by computers. Domain models can be represented in two forms: vector models and network models [31]. *Vector models*, the simplest types of domain models, represent the domain knowledge as a vector of concepts (keywords) without internal structure. In contrast, *network models* define the domain knowledge as a network of concepts representing the semantic relationships among them ("is-a", "part-of", or more semantically rich relationships). Such models are usually formalized in ontologies. This type is the most common in adaptive systems since such models empower reasoning and inference capabilities to fight the sparsity of knowledge and enable interest and knowledge propagation [33].

In the early stage of the field of adaptive hypermedia, authoring of domain models was the privilege of domain experts, i.e., teachers, course developers, or knowledge engineers. The tool support for this type of authoring was very poor. For instance, in the first versions of AHA! [54], experts had to formalize the domain knowledge in an XML-based markup language. The GUI support for designing domain models came first with AHA! 2.0 [53]. The system provided a GUI editor that allows defining concepts, relationships among them, and the propagation weights on the relationships.

Nowadays, several adaptive systems provide GUI support for authoring and representing domain knowledge. However, due to the focus of this thesis on scrutability, we describe only the systems revealing the content and structure of domain models to the end users. An interesting example of revealing the domain knowledge to users is provided by Advanced Learning Environment (ALE) [81], an adaptive e-learning system containing more than 20 online courses on design and architecture. The system leverages the domain knowledge, represented as a concept graph, to support concept-based navigation (Figure 4.4). By clicking an interesting concept in the text of a lecture, the user can start exploring the concept map, from which the user can learn about relationships among relevant concepts and find the paragraphs related to the selected concept.

Another example of giving access to the domain knowledge to the end user of an adaptive system is described by Mabbott and Bull [89]. The authors elaborate on four views of representing learner models to students, where each of the four views reflects the structure of the domain knowledge. The *related concepts* (Figure 4.5.a) view shows the hierarchical structure of topics. The *lecture view* (Figure 4.5.b) represents topics of the course organized according to the lecture structure. The *concept map* view (Figure 4.5.c) represents semantic relationships among topics.

Figure 4.4.: Concept-based navigation in ALE [81]

Finally, the *pre-requisites* (Figure 4.5.d) view represents a recommended sequence of learning topics. It is important to mention that each of these views uses color coding to denote the status of the student's knowledge in the topics.

The research literature reports on a few other adaptive systems revealing the domain knowledge to the user. These include but not limited to um Toolkit [73], NetCoach [130], STyLE-OLM [55], and DynMap+ [111]. However, all these systems allow users only to view the domain model. None of them allows the user to alter it. Though in the educational domain preventing the end users from write access to domain models can be justified by the fact that the end users are students not having sufficient domain knowledge and expertise, for adaptive systems in other domains letting the end users contribute domain knowledge might be of benefit for both the system and the user community. To our knowledge, no adaptive system has been proposed with a feature of allowing to collaboratively edit domain models. However, in the areas of Semantic Web and Knowledge Management, a number approaches have been proposed to collaborative domain knowledge engineering, including such systems as OntoWiki [9], OntoEdit [122], and Semantic MediaWiki [129].

myOntology [119] is one of the most mature and fully fledged systems for collaborative knowledge engineering. The system provides a web-based platform for building lightweight ontologies in a community-driven manner leveraging the wiki philosophy. Through an easy-to-use graphical interface, the software empowers the user community to create domain vocabularies by adding and editing concepts, connecting concepts with each other, and providing description of concepts in form of text or multimedia resources. myOntology keeps the history of all changes to

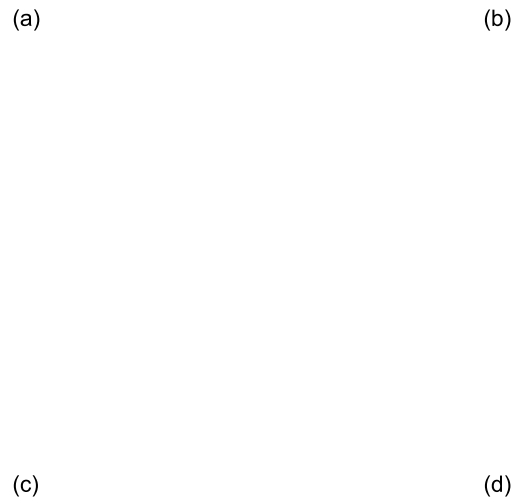


Figure 4.5.: Alternative views on domain knowledge [89]: (a) related concepts view, (b) lectures view, (c) concept map view, (d) pre-requisites view

all ontological elements and supports the undo mechanism. Finally, the software empowers importing and exporting ontologies in OWL format and supports the creation of freeze points in order of ensure stability of certain versions of ontologies.

An important aspect of domain models is the form of model representation to the user. Domain models can be extremely large and complex, hence it is important to provide the user with a comprehensible representation of the domain knowledge as well as with flexible and powerful management features. A large number of approaches have been proposed to visualizing knowledge [72], which mainly deal with the visualization of knowledge formalized in ontologies. The most popular techniques for visualizing ontologies are *indented trees* (e.g. the tree of concepts in the Protégé[2] ontology editor, Figure 4.6.d), *graphs* (e.g. graph visualization

[2]http://protege.stanford.edu/

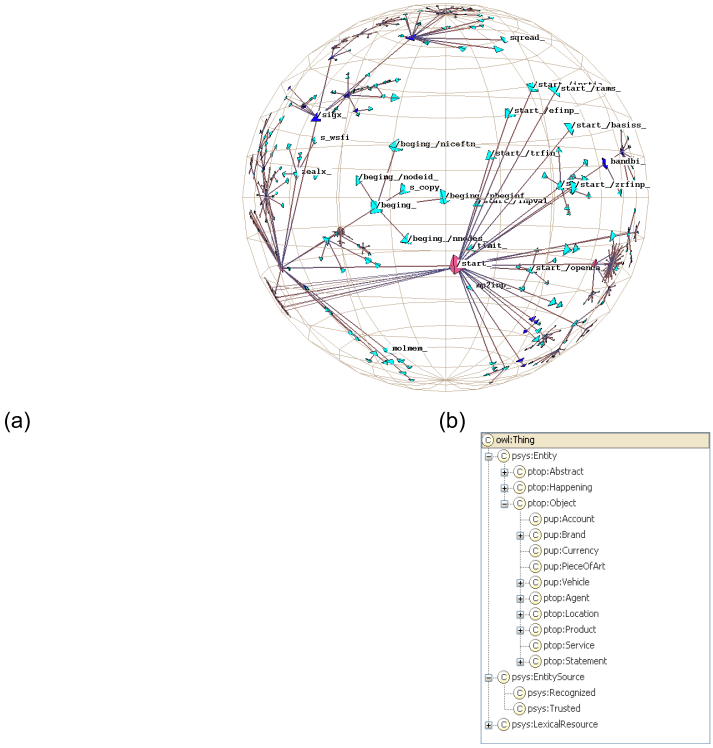


Figure 4.6.: Visualization of ontologies: (a) treemap visualization by Baehrecke et al. [10], (b) hyperbolic tree visualization by Munzner [100], (c) graph visualization by Gansner and North [58], (d) tree of concepts in the Protégé ontology editor (http://protege.stanford.edu/), (e) Aduna cluster maps (http://www.aduna-software.com/)

by Gansner and North [58], Figure 4.6.c), *tree maps* (e.g. visualization of gene ontologies by Baehrecke et al. [10], Figure 4.6.a), *cluster maps* (e.g. Aduna[3] cluster maps, Figure 4.6.e), and *hyperbolic trees* (e.g. the visualization by Munzner [100], Figure 4.6.b).

4.3. User-Generated Metadata

In order to achieve adaptation effects on a repository of information resources leveraging the content-based filtering approach, in addition to the definition of user features (e.g. user interests), it is important to have a metadata defining well-formed meaning of the resources in the system's repository. We will call this type *semantic metadata*. In case of the systems exploiting collaborative filtering for recommendations, it is important to have a metadata revealing the qualitative or emotional characteristics of resources (e.g. ratings and comments). We will call this type *qualitative metadata*.

Assigning the qualitative metadata has been from its origin a privilege of the end users. A large number of adaptive systems provide their users with this privilege (e.g. Knowledge Sea II [31], CHIP [8], and ALE [81]). On the contrary, defining the semantic metadata was originally a privilege of experts and system administrators. For example, in such adaptive educational systems as AHA! [54, 53], NetCoarch [130], and StyLE-OLM [55], defining the semantic metadata is an exclusive privilege of the teacher or course designer. End users, i.e., the learners, do not have permission to annotate the learning resources with semantic metadata. However, recently, there has been a shift towards achieving more openness. A few systems have been proposed to empower the end user with the possibility to semantically annotate resources. This section describes two of such systems: Metasaur [75] and iCITY [42].

Metasaur [75] is an interface for assigning semantic ontology-based metadata to learning objects in educational adaptive systems. Metasaur exploits the SIV (Scrutable Inference Viewer) tool for providing an interactive ontology visualization and assisting the user in assigning metadata. The interface (Figure 4.7) consists of four parts. The upper-left part contains a drop-down menu listing *all lectures* available for annotation. The upper-right section displays the content of the *selected lecture*, i.e., lecture slides. Users can navigate through the slides using the *previous* and *next* buttons. The bottom-right section displays a list of already assigned *metadata* and a menu through which the user can assign new metadata to the selected object. On the left side, the interface displays the SIV tool visualizing content of the underlying *domain ontology*. The ontology concepts are displayed as string labels on a two-dimensional surface. Upon selection of a label, the interface

[3] http://www.aduna-software.com/

rearranges the labels in a way that the selected label becomes more prominent by the increased gaps between the selected label and the other labels (heuristic evaluation is the currently selected term in Figure 4.7). Also, Metasaur enables users to perform keyword search in the ontology. Users can either select any string in the slide and press the *Search Selected* button or can type a string in the search box and click the *Search* button. The concepts matching to the given search criteria will be highlighted by the interface. Having found appropriate terms for annotation, users can add them to the list of metadata describing the currently selected slide.

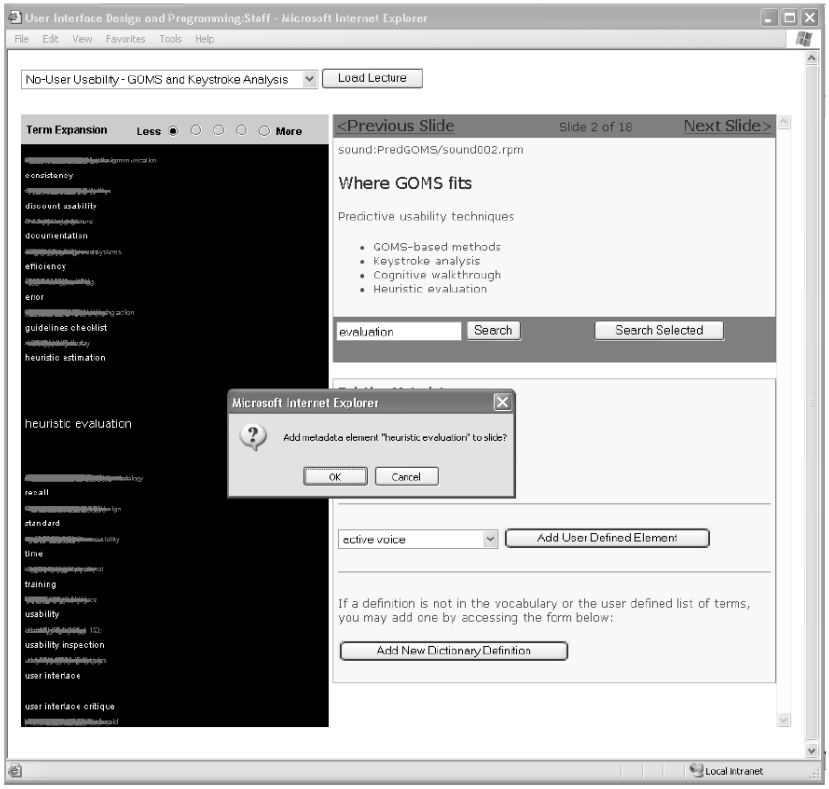


Figure 4.7.: Metasaur [75], an interface for ontology-based annotation of learning objects

iCITY [42] (Figure 4.8) is a community-enabled adaptive system capable of providing personalized recommendations to cultural events in Torino, Italy. The system provides a diverse spectrum of social features. Users can post new events, rate content, assign tags, and provide text comments. User-generated metadata is used by the system for providing supplemental navigation means as well as for refining and personalizing content. Tags are used to allow the user to navigate to

the events labeled with a certain keyword. Event lists are personalized according to the user model containing user interests and contextual information (e.g. location and event date). The events are displayed in an order such that the more relevant events are promoted to the top positions. Also, a number of visual cues are used to emphasize the recommended items. The recommended events are annotated with *thumb-ups* denoting the potential interest to the user, *ratings* denoting the community's opinion, and *number of views* denoting the popularity. Finally, the *font size of tags* in the tag cloud is used to indicate the most popular tags.

Figure 4.8.: iCITY's [42] recommendations of events

4.4. Scrutable Adaptation Rules

Most adaptive systems contain a set of rules defining what can be adapted in the system, how the adaptation should be performed, and when (under what conditions) it can be done. These adaptation rules can be defined in two ways: *explicitly* by the administrator or *implicitly* by the system based on the user's interaction with the system. Rules of both types have a potential to generate adaptation effects that surprise or frustrate user. Therefore, it is important that in addition to the ability to scrutinize the user model, the user should be enabled to inspect and possiblly alter adaptation rules.

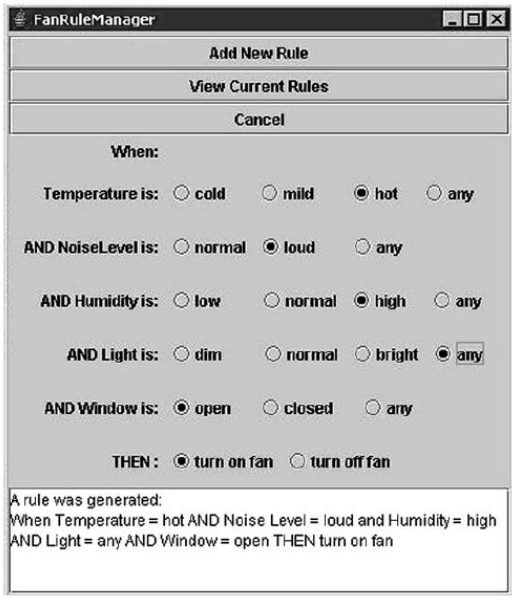


Figure 4.9.: IOS [45] interface for editing adaptation rules

Explicitly defined adaptation rules have been a common form of defining adaptation rules for authoring adaptive educational hypermedia (AEH). In AEH systems, usually the instructor or course designer defines adaptation rules for the learning materials. For example, in T-MAESTRO [110], an authoring tool for personalized learning, teachers are enabled through a GUI to create adaptive courses by defining the course structure, its metadata, and setting the adaptation rules. For instance, through the editor, the teacher can create two different structures of a course teaching Italian language. The teacher define what structure is recommended to students preferring to learn Italian and what structure better suits need of those who plan to visit Italy as tourists. A number of other AHS systems provide a GUI environment to defining adaptation rules for different learning materials; these systems include SIGUE [43], NetCoach [131], and ALE [120]. Though many of AEH systems have some authoring tool allowing viewing and defining personalization rules, this function is still a privilege of the instructor. None of the mentioned system enables the learner to view and modify the adaptation rules created by the instructor.

Implicitly created adaptation rules are inferred by the system autonomously based on the observation of the user's interaction with the system. For instance, CAP (Calendar APprentice) [96] analyses the user's use of calendar in order to identify the user's scheduling preferences. The system runs a decision tree learning

algorithm on the user's calendar log and produces a set of learnt adaptation rules represented in IF-THEN format. These rules are then used for achieving the system's proactive behavior, e.g., the system making an advice regarding the event's duration and location. In case of proactively behaving adaptive systems, it is even more important that the user understands the system's behavior and is able to correct misconceptions. Intelligent Office System (IOS) [45], for instance, provides a comprehensive support for scrutability of implicitly created adaptation rules. The system can learn the user's behavior in an office environment and infer rules for achieving adaptive behavior of office automation systems, e.g., turning on/off the ventilation, heating, and light. The system allows users viewing and (if necessary) overriding these rules as well as creating their own rules as shown in Figure 4.9.

4.5. Comprehensible and Controllable Personalization

According to Jameson [70], in order to ensure transparency of adaptive systems, users should understand the system's actions. With respect to the adaptivity it requires that the user is aware of what is adapted and why it is adapted. Czarkowski [50] argues that an adaptive system should (1) provide an overview of personalization that has been performed on a page and (2) provide explanation of the reasons why the personalization was performed, e.g., what attributes from the user model influenced it. Both requirements are fulfilled by SASY [50]. The system provides a Highlight tool that summarizes the adaptation effects that took place and the reasons for that. The Highlight tool (Figure 4.10) is implemented as a side panel that lists the adaptation effects performed on the current page, i.e., it shows how many items have been added or removed. When the user clicks on the links in the summary, the system reloads the current page and highlights the fragments that have been removed or added as a result of personalization. Additionally, the tool provides explanations of the adaptation effects. It adds an explanation message to the highlighted fragments (e.g. "Added because your profile has: You have a low budget") and shows all explanations in the side bar.

Unlike SASY providing a textual explanation of adaptation, PeerGlass [79] leverages a visual approach. The system's adaptation effect is the automatic selection of news articles matching the user interest model in a personalized newspaper system. For each selected article, the system explains why the article was selected by displaying so called Totem poles (Figure 4.11) representing the article's features that strongly match the user's interests. The Totem poles are represented in a three-dimensional space and can be freely rotated by the user in order to see all four sides of poles, which represent four categories of article features: type (e.g. interview or press release), topic (e.g. basketball or Clinton), nature (e.g. liberal or depressing), and source (e.g. Detroit News^{TM}or New York Times^{TM}). The Totem

Figure 4.10.: SASY's [50] Highlight tool providing an overview and explanation of personalization

poles are aimed to help the user to understand why a certain article was selected. It also helps the user to see whether the system's believes about the user's interests are correct. In case when the system has certain misconceptions about the user's interests, the user can access her user model by clicking the *Examine Profile* button located at the bottom of the page and make necessary corrections.

Similarly to PeerGlass, visualization techniques are also used in TaskSieve [4], an adaptive information retrieval system for intelligence analysis. TaskSieve leverages the Adaptive VIBE visualization framework to display relations between the retrieved documents, search terms, and user model. Figure 4.12 displays a set of documents retrieved for a given user query. The retrieved documents are represented by squares. The yellow circles denote the user search terms, whereas the blue circles represent terms from the user model that are present in the retrieved documents. The proximity of a square to a circle denotes the relevance of the corresponding document to the term. To further explore the relevance between documents and terms, users can move terms on the canvas, which will affect the positioning of documents. Additionally, the system highlights the search terms and terms from the user model in the text of retrieved documents. These techniques

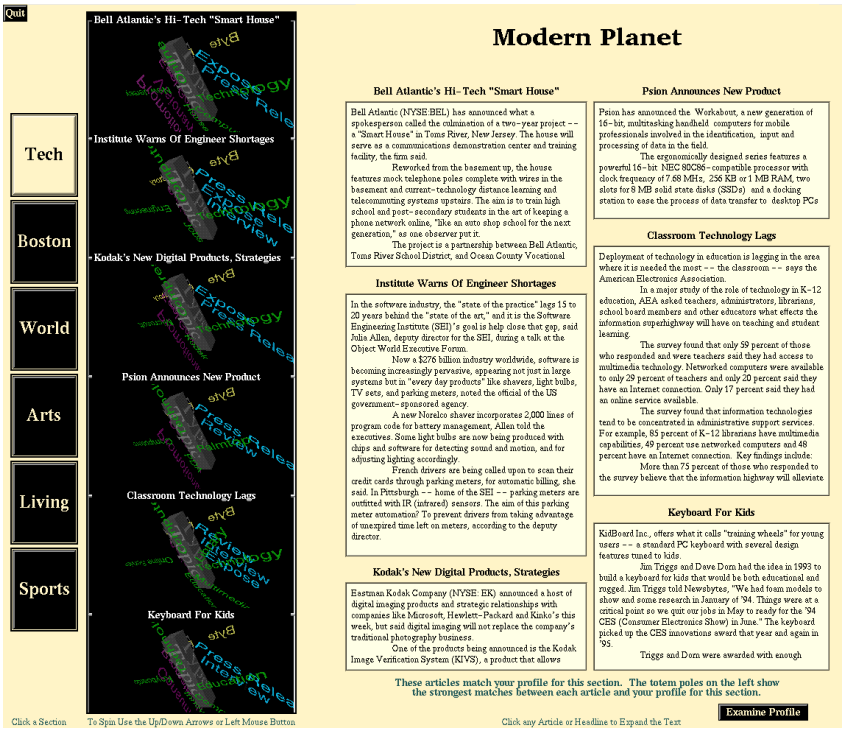


Figure 4.11.: Totem poles [79] showing why the articles were selected

help users to comprehend the influence of the user model on the end adaptation effects.

In case of recommender systems, the system can inform the user about the applied personalization effects by displaying the system-generated recommendations in separate blocks and making it explicit that the content of those blocks is personalized. For instance, Amazon displays a number of sections with personalized content on its front page and titles these sections in a way letting the user understand that their content is tailored to the user, e.g.: "Our Recommendations for You", "Related to Items You've Viewed", "Inspired by Your Browsing History", "Customers with Similar Searches Purchased", and so on. Also, within those sections, Amazon explains why the products have been recommended. For example, in case of the section titled "Related to Items You've Viewed" (Figure 4.13), the system displays the product that the user has viewed and the list of other products that were viewed by the customers who also viewed this product. Moreover, this section contains a link to the user's browsing history, where the user can see all the viewed items and delete the items that she does not like to be stored and used for recommendations. In case of Amazon's recommendations that are based on the

Figure 4.12.: Adaptive VIBE [4] visualizing the relevance between retrieved documents, search terms, and the user model

user's purchasing or rating history, for every recommended product, the system displays an explanation, which can be either the user's purchase of some relevant product ("We recommended you this item because you bought...") or a high rating given by the user on a relevant product ("We recommended you this item because you rated..."). In a way similar to the search-based recommendations, the user is enabled to influence the recommendations based on the purchasing and rating history: The user can (1) rate a recommended item as uninteresting or (2) mark the previously bought/rated items to be not used for recommendations. A similar approach to explaining personalization is used in the music recommender system proposed by Bogdanov et al. [24]. For each recommended track, the system injects a text message explaining why it is recommended. More specifically, it displays a related track that was explicitly marked by the user as favorite.

In addition to explicitly showing what has been personalized and allowing the user to influence it by changing some attributes in the user model, some systems allow the user to define whether the adaptation should take place at all, define the extent to which the system can personalize, and enable or disable the usage of user's interaction history for adaptation. For example, Intelligent Office System (IOS) [45] allows the user to specify whether the system can operate (turning on/off the office light, heating, ventilation) either fully autonomously or upon the user's permission. In the system's main control GUI (Figure 4.14), the user can define whether the system should prompt the user before turning on/off the device or can

Figure 4.13.: Amazon's recommendations: Related to Items You've Viewed (screenshot made from http://www.amazon.com on February 23, 2010)

do it without prompt. Also, the system allows setting a threshold for proactive behavior by clicking the appropriate buttons in the *Proactive Threshold* section. E.g., if the user wants that the system performs only those actions for which it has high confidence, she can click *High* button. Finally, the user is enabled to deactivate the proactive behavior completely by clicking *Proactive off* button.

Figure 4.14.: Control on the system's adaptive behavior in IOS [45]

Another type of control is demonstrated by Amazon. The system allows the user to activate/deactivate the recording of the user browsing history. By deactivating this feature, the system stops recording the user's searches and visited pages. It, therefore, prevents the system from issuing recommendations based on the user's browsing history.

4.6. Conclusions

In this chapter, we presented an overview of approaches to scrutinizing various components of adaptive systems. We described methods and visual approaches to inspecting user models. Also, we described several adaptive systems that reveal domain knowledge models to the user community and provide users with access to metadata representing the semantics of documents. Finally, we presented several methods for explaining adaptation effects and for controlling personalization. In this last section, we discuss some of the systems described above with respect to the requirements identified in Chapter 3. Table 4.1 provides an overview of the compliance of the discussed systems to the identified requirements.

User Models

All approaches to scrutable user modeling presented in this chapter include methods for automatic generation of user models based the user interaction with the adaptive system (Requirement F3: Automatic update of user model). For example, interest user models of the personalized news systems YourNews [5] and Totem Poles [79] are generated based on the set of keywords extracted from the news that users have read. Knowledge user models in the NetCoach [130] adaptive learning system are generated based on the evidence of the concepts describing the learning objects accessed by students, the tests they have attempted, and their explicit marking of concepts.

A few scrutable user models also fulfill Requirement F1: Semantic representation of user model. Among the systems described in this chapter, scrutable user models are represented using a semantic formalism only in intelligent tutoring systems. NetCoach [130], um Toolkit [48, 73], and SIV [75] model learner knowledge using semantic concepts defined in ontologies. DynMap+ [111] and ViSMod [133] represent learner knowledge models using concept maps and concepts graphs respectively. STyLE-OLM [55] combines ontologies with probabilistic Bayesian Networks to model student knowledge. User models of those intelligent tutoring systems represent mostly knowledge of students. Very little research has been conducted on scrutable user models that leverage a semantic formalism to represent other user information, for instance, user interests. Also, very little research has been conducted on semantic scrutable user models for domains other than e-learning.

Another aspect of scrutable user models that has been very little investigated is the edit access to user models (Requirement F2: User access to user model) represented semantically (Requirement F1: Semantic representation of user model). Most systems allow users only to view information stored in semantic user models, but not to edit it. Among the systems described in this chapter, only um Toolkit [48, 73] fulfills both requirements. It models user knowledge using an ontology and

allows users to edit it, i.e., it allows users to override system's believes. To our knowledge, there are no other approaches to scrutable user modeling that represent user information using ontologies and allow users directly edit this information.

Also, relatively little attention has been paid on explaining users the provenance of information in the model (Requirement F4: Explain sources of information in user model). Mostly scrutable user models show users the resulting believes, but not the sources based on which these believes were made. To our knowledge, only um Toolkit [48, 73] provides this information explicitly. It provides so called *justify* view over the model that displays the sources of evidence that the system's conclusions are based on. We believe, however, that this function must be provided in every scrutable user model since it ensures the predictability of the system's behavior [70]. Users should know and understand how their interaction with the system affects the information in their user model.

Our discussion on scrutable user modeling can be summarized as follows: Mostly research on scrutable user models represented semantically is focused on student knowledge models in the e-learning domain. Very little research has been conducted on semantic representation of scrutable user models in other domains and for other types of user models, e.g., user interest models. Also, very few approaches have been proposed to providing users with the edit access to semantic user models. Finally, very little attention has been paid to explaining users the provenance of information in their models.

Domain Knowledge

In this chapter, we reported on several adaptive systems that represent the domain knowledge separately from the user model and use a machine-processable notation for describing its semantics (Requirement F5: Semantic representation of domain knowledge). For example, the adaptive system for accessing multimedia learning objects Metasaur [75] represents the domain knowledge in an OWL ontology. Similarly, the adaptive learning system NetCoach [130] models the semantics of courses in a knowledge base, which defines domain concepts and relations. These systems allow users to view domain knowledge models to some extent. However, none of these systems allow users to alter the domain knowledge (Requirement F7: User community access to domain knowledge model).

Also, very few adaptive systems have mechanisms for updating this knowledge automatically (Requirement F6: Automatic generation of domain knowledge). Most systems require a knowledge engineer or domain expert to create and extend these models. Among the described systems, only the adaptive learning system by Kay and Lum [75] includes a component called Mecureo to construct domain ontologies automatically based on existing dictionaries and glossary sources.

In summary, our analysis of domain knowledge models shows that very little research has been conducted to investigate the possibilities of updating domain models in adaptive systems by the user community. Also, in the research area of scrutable adaptive systems, very little attention has been paid to methods for automatic extension of domain knowledge. We believe, however, that by providing hybrid methods for updating domain knowledge models that combine user community contributions with system generated updates, it is possible to achieve a better coverage of the domain knowledge and improve its accuracy. This can improve the quality of adaptation effects and search in the system. We also believe that it can decrease the costs of maintaining the domain knowledge drastically.

Metadata

We presented several adaptive systems which content is annotated using machine-processable semantic notations (Requirement F8: Semantic representation of metadata). For example, content of such adaptive learning systems as AHA! [54, 53], NetCoach [130], STyLE-OLM [55], and Metasaur [75] is annotated with semantic metadata. However, only one of these systems, Metasaur, allows users to provide such metadata (Requirement F10: User community access to metadata). Semantic metadata in other systems can be provided only by teachers or course designers.

With respect to Requirement F9: Automatic generation of metadata, two of the systems described in this chapter include methods for generating content metadata. Content in adaptive news systems YouNews [5] and Totem Poles [79] is annotated with automatically extracted metadata. However, in both systems the metadata is represented as a vector of unconnected keywords. We found no scrutable adaptive system that provides methods for automatic generation of semantic metadata.

To sum up, very little research has been conducted to investigate possibilities for providing semantic metadata by user communities in adaptive systems. Also, very little attention has been paid to automatic generation of semantic metadata in scrutable adaptive systems.

Personalization Rules and Effects

We reported on several adaptive systems that provide users with an explanation of personalization effects (Requirement F12: Explanation of personalization). The adaptive hypertext system SASY [50] highlights page fragments that were personalized and displays the attributes of user model that influenced this personalization, e.g., it displays such text messages as “Added because your profile has: You have a low budget”). However, assertion of lengthy textual messages into the original content might negatively affect the usability of the system. Users, especially on web systems, tend to scan through content to grab the key points instead of read-

ing every sentence. Assertion of textual messages explaining personalization effects might hinder them in the scanning. Explanation messages are not the main interest of users. Hence, they might clutter the main content and decrease its scannability. As a result, users might need more time to interpret the content than they would need on a page without the explanation of personalization effects.

Unlike SASY, the PeerGlass architecture [79], TaskSieve [4], and the system proposed by Tsandilas and Schraefel [124] leverage visualization techniques to explain personalization effects. For instance, Kliger's PeerGlass architecture, next to personalized content, displays 3D visualizations of the parts of user model that were used for personalizing the content. Additionally, it allows users to open the user model directly from the personalized page, make necessary changes in it, and see the effect on personalization (Requirement F15: Explain implications of altering models). However, the user models in PeerGlass architecture and in the other two systems have a rather simple structure. Hence, it is not clear how suitable this approach is for explaining personalization effects generated based on complex user models, e.g., ontology-based user models. It is also not clear how well these approaches can be applied for explaining the implication of changes in complex user models.

With respect to Requirement F14: User access to personalization rules, we found only one system that allows users to view and edit rules that it uses for personalization. Intelligent Office System IOS [45] can generate rules for controlling office systems fully automatically based on operation patterns of the office workers, such as the time for turning on and off the ventilation, heating, and lights. But it also allows office workers to view these rules and override them if necessary. Additionally, this system allows users to turn the proactive behavior off (Requirement F13: Turning personalization on and off).

To conclude, very little research has been conducted on explanation of personalization effects that are generated based on complex user models, in particular, ontology-based user models. Also, relatively little attention has been paid to methods for explaining the implications of altering ontology-based user models. Finally, with respect to Requirement F11: Portlet-specific personalization rules, we found no portal solutions that allow users to define personalization effects for individual portlets.

	AHA! [53]	ALE [81]	APT [49]	DynMap+ [111]	iCITY [42]	IOS [45]	Metasaur, SIV [75]	NetCoach [130]	PeerGlass [79]	SASY [50]	STyLE-OLM [55]	TaskSieve [4]	Tsand. & Schraef. [124]	um Toolkit [73]	ViSMod [133]	VlUM [126]	YourNews [5]
F1: Semant. represent. of user model	+	+	+	+	-	-	+	+	-	-	+	-	?	+	+	+	-
F2: User access to user model	-	V	V	V	-	E	V	V	E	E	V	E	E	E	V	V	E
F3: Automatic update of user model	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
F4: Explain src. of inf. in user model	-	-	-	-	-	-	-	-	-	-	-	-	-	+	-	-	-
F5: Semant. represent. of dom. knowl.	+	+	+	?	-	-	+	+	-	-	+	-	?	+	+	+	-
F6: Automatic generation of dom. knowl.	?	-	?	?	-	-	+	-	-	-	-	?	?	?	?	?	?
F7: User commun. access to dom. knowl.	-	V	?	?	-	-	V	V	-	-	-	-	?	V	V	V	-
F8: Semant. represent. of metadata	+	?	?	?	-	-	+	+	-	-	+	-	?	?	?	+	-
F9: Automatic generation of metadata	?	?	?	?	?	-	?	-	+	-	-	+	?	?	?	?	+
F10: User community access to metadata	-	?	?	?	E	-	E	-	-	-	-	-	?	?	?	?	-
F11: Portlet-specific personalization rules	NA	NA	NA	NA	-	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
F12: Explanation of personalization	-	-	-	-	-	-	-	-	+	+	-	+	+	-	-	-	-
F13: Turning personalization on and off	-	-	-	-	-	+	-	-	-	-	-	-	-	-	-	-	-
F14: User access to personalization rules	-	-	-	-	-	E	-	-	-	-	-	-	-	-	-	-	-
F15: Explain implicat. of altering models	-	-	-	-	-	-	-	-	+	-	-	-	+	-	-	-	-

Table 4.1.: Comparison of systems for scrutable adaptivity: (V) - view, (E) - edit, (+) - fulfilled, (-) - not fulfilled, (NA) - not applicable, (?) - no information

CHAPTER 5

Framework for Scrutable Adaptivity

This chapter presents a framework for scrutable adaptivity in community-enabled web portals. Section 5.1 provides an overview of the conceptual architecture of the framework. It describes the four units constituting the framework and depicts the layered architecture for communication among the system components. Section 5.2 provides information about two portals in which we prototypically integrated this framework. These portals are used in this thesis as examples showing functions of various components of the framework. Also, this chapter provides a detailed description of the first two units of this framework: Section 5.3 presents the unit for domain modeling and Section 5.4 presents the resource management unit.

Parts of research presented this chapter were published in our earlier papers. The research related to the portals that we use as application scenarios in this thesis is published in [21, 22]. Our work on personalization modeling is published in [14]. Our approach to automatic generation of metadata for portal content is published in [115, 116].

5.1. Conceptual Architecture

The main goal of this thesis is to develop a framework for enabling users to scrutinize the adaptation models and process in web portals. According to the functional requirements elaborated in Chapter 3, for adaptation effects in a web portal, several components are required. The portal needs a user model providing information about individual users, a metadata repository describing meaning of portal content, and personalization rules defining the logic of adapting portal content to user needs. In order to achieve automatic selection of portal resources that match user needs,

both the user model and metadata repository must be defined using the same vocabulary. This requires a domain model that provides a vocabulary defining domain knowledge in a machine-processable language.

According to Requirements F1, F5, and F8, it is essential that the user model, domain model, and metadata are defined in a formal and highly expressive language capable of representing rich semantic information. To fulfill Requirements F3, F6, and F9, the system must provide mechanisms for automatic update of these three components. To comply with Requirements F2, F7, and F10, the system must provide mechanisms and interfaces allowing users to view and edit the domain model, metadata repository, and user model. Also, to fulfill Requirement F12, it is important that the system provides users with a comprehensible explanation of personalization effects that take place on portal pages. Requirements F14 and F13 demand that the system gives users full control over personalization by allowing users to adjust personalization effects to their preferences or turn personalization off.

To fulfill the above mentioned requirements, we propose a framework consisting of four units, namely, units for domain modeling, resource management, user modeling, and personalization. Figure 5.1 displays the system architecture of the proposed framework.

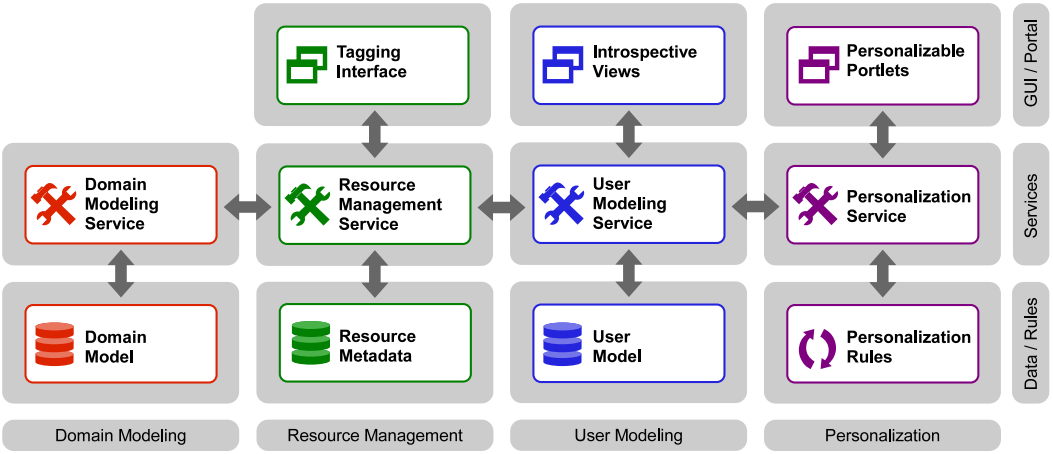


Figure 5.1.: System architecture

Domain Modeling Unit (DMU) is responsible for storing, accessing, and managing the domain model. The domain model is defined in Web Ontology Language (OWL)[1] capable of representing rich semantic information (fulfills Requirement F5: Semantic representation of domain knowledge). Access and management operations

[1] `http://www.w3.org/TR/owl-features/`

over the domain model are implemented by the domain modeling service. Among others, this service implements methods for automatic update of the domain model (fulfills Requirement F6: Automatic generation of domain knowledge). Also, the domain model can be indirectly updated by users, which fulfills Requirement F7: User community access to domain knowledge model. This is achieved by allowing users to semantically organize tags that they use for annotating portal resources and items in their user models.

Resource Management Unit (RMU) consists of a repository providing metadata of portal content and a service and an interface for managing this metadata. The metadata repository is represented using the semantic vocabulary of the domain ontology, which fulfills Requirement F8: Semantic representation of metadata. The resource management service implements methods for accessing and editing this metadata. Also it implements methods for generating metadata fully automatically by processing portal content with tools for natural language processing (fulfills Requirement F9: Automatic generation of metadata). In addition, metadata can be provided by users through a graphical user interface (fulfills Requirement F10: User community access to metadata).

User Modeling Unit (UMU) contains a user model providing information about individual users. The user model is designed as an overlay model that defines user features as a subset of semantic concepts from the domain model (fulfills Requirement F1: Semantic representation of user model). Access and management methods on the user model are implemented by the user modeling service. This service also implements methods for automatic update of the model based on the user browsing history as well as by inferring new features using semantic relations between concepts defined in the domain model (fulfills Requirement F3: Automatic update of user model). Additionally, the unit provides a graphical user interface that allows users to view and edit their user models (fulfills Requirement F2: User access to user model).

Finally, *Personalization Unit (PU)* provides rules for personalizing content, interfaces for delivering personalized content to users, and methods allowing users to control personalization. Personalization rules are defined as a sequence of transformation procedures that applied to the original content to transmute it into a personalized state wanted by the user. The framework allows setting personalization effects at the level of individual portlets (fulfills Requirement F11: Portlet-specific personalization rules). The unit provides a *PersonalizablePortlet* class. This class is designed for portlet application developers who need programming portlets that deliver personalized content to users and give users full control over personalization. This class implements a method that embeds a GUI element showing users what personalization effects have been made in the portlet (fulfills Requirement F12: Explanation of personalization). In this element, users can also adjust personalization effects to their preferences (fulfills Requirement F14: User access to personaliza-

tion rules). Moreover, the class implements a method allowing users to activate and deactivate personalization in portlets, which fulfills Requirement F13: Turning personalization on and off. Finally, personalizable portlets place an entry point to the user's interests profile next to the personalized content (fulfills Requirement Q1: Findability of scrutinizing tools), so that, in case of incorrect personalization behavior caused by a wrong assumption about users' interests, users can easily access their profile and override the system's belief. A change in the interest profile is immediately projected on the personalized content. The availability of this function fulfills Requirement F15: Explain implications of altering models.

The system components are distributed across four layers according to their function and the execution environment. *Data / Rules Layer* represents the system's structured data and models. *Service Layer* represents the system's services and tools implementing the logic for managing and accessing the data models. The services access the data models either through the ODBC[2] and JDBC[3] protocols or via application programming interfaces (API) provided by the repository that stores the model. The communication among the services is enabled through HTTP protocol and JavaScript Object Notation (JSON) is used as the serialization format. *GUI / Portal Layer* includes the graphical user interfaces through which the user community can access the adaptation models as well as it includes the interfaces that provide personalized content. Similar to the inter-service communication, the GUIs communicate with services through HTTP protocol and use JSON for serialization.

5.2. Application Scenarios

For the proof-of-concept implementation, the framework was integrated into two web portals: a news aggregation portal and a portal for biochemical literature. In the rest of the thesis, these portals will be used as example application scenarios for the framework. Both portals were deployed on IBM WebSphere Portal[4] provided by IBM for research and prototypical development in the context of Minerva Portals project[5], a collaborative research initiative of Friedrich-Schiller University of Jena and IBM Deutschland Research & Development GmbH. Also, both portals are integrated with the Semantic Assistants framework [132]. This integration allows users to leverage a variety of tools for Natural Language Processing and text mining within the portal. A more detailed description on our work on the integration of portal technology with NLP can be found in [21].

[2] Open Database Connectivity
[3] Java Database Connectivity
[4] `http://www.ibm.com/software/websphere/portal/`
[5] `http://www.minerva-portals.de`

The adaptive behavior of both portals, the news aggregating portal and the portal for biochemical literature, can be scrutinized and controlled by users. Some portlets in these portals are implemented using the *PersonalizablePortlet* class (Section 7.3) provided by the framework for scrutable adaptivity. They enable users to turn personalization on and off and fine-tune personalization effects at the level of individual portlets. Section 7.4 describes in detail the process of user-controlled personalization.

5.2.1. News Aggregating Portal

News portals are one of the most widespread applications of portal technology. They provide a single-point of access to a broad spectrum of news content from one or multiple media agencies. There are several aspects that strengthen the need for adaptive behavior in news portals. First, the majority of news portals, especially news aggregating portals, provide an enormous amount of content on various topics. Second, the set of available news stories is updated very frequently, sometimes on an hourly basis. Both aspects may impede users in navigating through the portal and finding interesting news stories. Also, due to the temporariness of user interest in happenings or situations described by news stories, users tend to scan through stories, instead of thoroughly reading them. They want to grab key points of stories in the shortest possible amount of time. Hence, users need an intelligent support that assists them in finding the most interesting and relevant news stories and helps in their further analysis and interpretation.

The news aggregating portal developed for this thesis provides a personalized access to news stories of such media agencies as BBC™, CNN™, and Wikinews[6]. News content is harvested by the resource management service (Section 5.4.2) using RSS feeds provided by the agencies. In the portal, news stories are organized into news sections, such as World & Politics, Business, Science & Technology, etc. Figure 5.2 displays a portal page with a list of news stories from the World & Politics section (in the middle). In addition to news on politics, the page displays a list of recommended news stories from other categories (on the right).

By clicking on the title of a news story, the portal will display its content on a separate page (Figure 5.3). Users can process content of news stories with a number of tools for Natural Language Processing provided by the Semantic Assistants framework. This assistance can be requested through the *Semantic Assistants* menu (Figure 5.4) that can be opened by clicking the *Semantic Assistants* button, shown as a battler icon and located on the title bar of the portlet displaying the news story. Using this menu, the user can connect to a Semantic Assistants server and view a list of offered assistants. This list is presented as a collapsible menu. In

[6] http://www.wikinews.org

Figure 5.2.: News aggregating portal: List of news stories

the default state, it displays the name and status of assistants. A click on an assistant name will open a submenu providing the assistant description and a number of control elements for invoking the assistant and viewing the results.

In order to invoke an assistant, the user needs to set the run parameters (if applicable) and click the *Run Assistant* button. At this moment, the assistant status will be set to *requested*. As soon as the result is available in the portal, either found in the portal cache or newly received from the Semantic Assistants server, the assistant status will be changed to *result available* and its submenu will display options for viewing the result. More specifically, it will display a list of view portlets.

The list of view portlets is determined based on the result type. For example, the *Information Extractor* assistant, selected in Figure 5.4, extracts named entities from the source text and returns them as an annotation set. Results of this type can be rendered in four ways: on a map, as images, as an index, or the entities can be highlighted in the source text. Users can select view options they like and click the *Display Results* button to view the assistant outcomes. If users want to display results of several assistants in one portlet, the results will be aggregated.

Figure 5.3.: News aggregating portal: Individual news story

5.2.2. Portal for Biochemical Literature

This application scenario was developed within the Genozymes project at Concordia's Centre for Structural and Functional Genomics[7]. We created a web portal [BIBM paper] for biologists, biochemists and geneticists that work on lignocellulose research. The goal of this research is to find novel ways of creating bioproducts and biofuels from green waste. Part of this work is the curation of characterized glycoside hydrolases[8] of fungal origin from the domain literature.

To support these researches in the biocuration task, the portal provides a personalized single-point of access to abstracts of scientific publications harvested from multiple databases, e.g., from PubMed[9], and supports a further analysis of these abstracts. More specifically, it allows users to process the harvested content with

[7] CSFG, `http://genomics.concordia.ca/`
[8] family of enzymes used to break down plant cell walls
[9] `http://www.ncbi.nlm.nih.gov/pubmed`

Figure 5.4.: Semantic Assistants menu

a number of semantic assistants. For instance, it provides assistants for extracting named entities, such as organisms, enzymes, genes, substrates, etc. Also, it provides assistants for generating summaries and indexing of literature.

Additionally, the platform allows users to obtain a personalized view on the literature and results of semantic assistants. The platform can sort abstracts either chronologically or according to the relevance to the user's interest profile, which the platform builds unobtrusively based on the user browsing history. It can foreground the most relevant abstracts or fragments of abstracts that match items in the user profile.

Figure 5.5 displays a personalized page that users see after they have logged in into the portal. This page consists of a number of portlets providing different types of content and functions. The *Query* portlet on the left displays a list of user search queries, which are used by the portal to retrieve publications from scientific databases. This portlet allows users to add, edit, and delete queries and organize them hierarchically. Upon a mouse click on a query, the portal will display a list of matching publications in the *Listing* portlet.

Figure 5.5.: Portal for biochemical literature

Similarly as the *News* portlet on the news aggregating portal (Section 5.2.1), the *Listing* portlet allows users to request various types of semantic assistance. Users can view a list of named entities extracted from the publications, their summaries, or an index. All types of assistance supported by the portlet can be seen in the *Semantic Assistants* menu (Figure 5.4). In this menu, users can choose an assistant they want and set desired view options for the assistant results.

Depending on the type of assistant, its results can be displayed in the source text, as an index, a map, or a text in a side portlet. For instance, Figure 5.5 displays results of mycoMINE assistant [93], which extracts entities and facts related to fungal enzymes involved in lignocellulose degradation, such as enzymes, organisms,

genes, substrates, pH, temperature or activity assay conditions, etc. The entities extracted by the assistant are underlined in the text of publications listed in the origin portlet. They are also displayed as an index in a side portlet. The index portlet lists all entities grouped by entity type. By hovering the mouse pointer over an entity in the index, all mentions of this entity will be highlighted in the publications. Users can also obtain some additional information about the extracted entities. By clicking an entity in the index, they will see all features of the entity, e.g., alias, abbreviations, links to scientific databases, such as BRENDA[10] and UniProtKB[11].

5.3. Domain Modeling Unit

The domain modeling unit is responsible for keeping the *domain knowledge model* (Section 5.3.1) up-to-date and as complete as possible. The domain model is updated automatically based on the semantic analysis of portal content. It can also be altered by the user community. All operations on the model are performed by the *domain modeling service* (Section 5.3.2). The user community access to the model is enabled through the domain modeling functions of the *tagging interface* (Section 5.4.3) and user modeling interface *IntrospectiveViews* (Section 6.3.5).

5.3.1. Domain Knowledge Model

The domain knowledge model represents the semantic vocabulary describing the portal's domain knowledge defined in a machine-processable format. This vocabulary is used by the other units for semantically annotating the system's resources in the metadata repository (Section 5.4.1) and defining user features in the user model (Section 6.1). To mitigate the problem of scarcity of the available information about user features and achieve a higher recall and precision in selecting the resources to recommend, the domain knowledge model is represented as an ontology formalized in the Web Ontology Language (OWL)[12], which fulfills Requirement F5: Semantic representation of domain knowledge.

The domain knowledge model consists of two parts: a terminological component (TBox) and an assertion component (ABox). TBox contains definition of ontology classes and their object and data properties, whereas ABox provides instantiation of the classes and properties defined in TBox. Physically TBox is represented as an OWL file located on the portal server. ABox is represented as a triple store managed by Sesame Framework[13].

[10] `http://www.brenda-enzymes.info`
[11] `http://www.uniprot.org/help/uniprotkb`
[12] `http://www.w3.org/TR/owl-features`
[13] `http://www.openrdf.org`

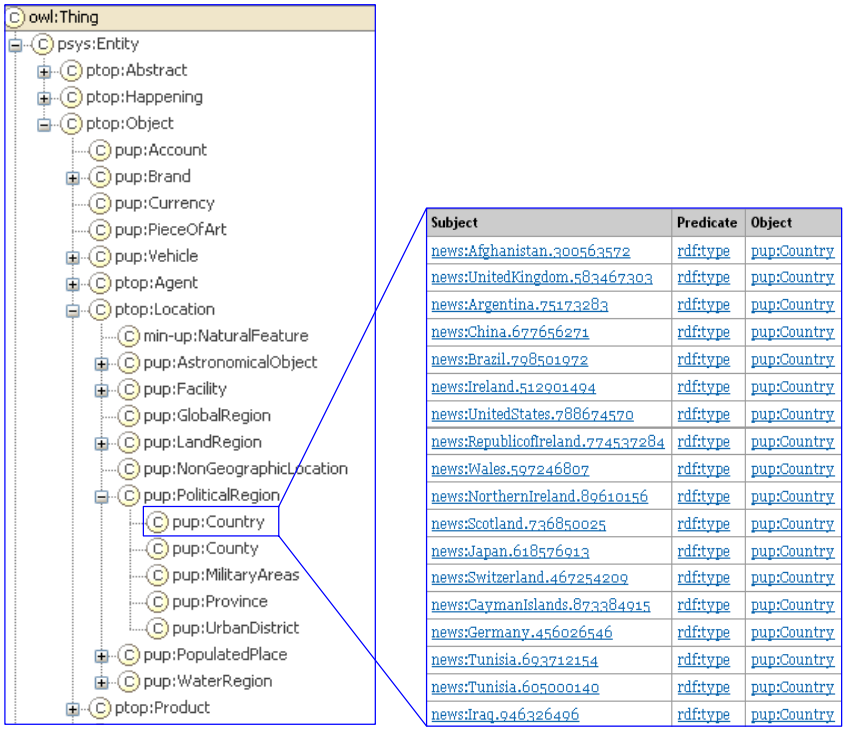


Figure 5.6.: News domain ontology: (a) - terminological component TBox; (b) - assertion component ABox

Figure 5.6 displays an excerpt of the domain ontology that was developed as a proof-of-concept for the news aggregating portal (Section 5.2.1). This ontology is grounded on the Proton[14] upper-level ontology, which defines general concepts such as organizations, geographic locations, products, etc. (Figure 5.6(a)). The instantiations of these concepts are defined in ABox. E.g., for the concept *Country*, ABox defines actual countries (Figure 5.6(b)). Instances can be interconnected through object properties defined in TBox. Figure 5.7 shows relations of the instance *United States* (country) to other instances. Among others, it shows relations to the federal states and cities located in the country. Also, as seen from Figure 5.7, the model stores information about morphological variations of instance names. It represents this information through *hasAlias* assertions.

Since the domain model is accessible for editing by the entire user community, where different users might have different views on the world, it allows users to

[14] `http://proton.semanticweb.org`

Subject	Predicate	Object
news:UnitedStates.788674570	rdf:type	pup:Country
news:UnitedStates.788674570	psys:mainLabel	"United States"
news:UnitedStates.788674570	psys:hasAlias	news:UnitedStates.219581662
news:UnitedStates.788674570	psys:hasAlias	news:US.280845373
news:NewYork.576734177	ptop:locatedIn	news:UnitedStates.788674570
news:Phoenix.728966655	ptop:locatedIn	news:UnitedStates.788674570
news:Washington.224447097	ptop:locatedIn	news:UnitedStates.788674570
news:Californian.37046109	ptop:locatedIn	news:UnitedStates.788674570
news:Texas.573316880	ptop:locatedIn	news:UnitedStates.788674570
news:California.77993381	ptop:locatedIn	news:UnitedStates.788674570
news:Utah.71316202	ptop:locatedIn	news:UnitedStates.788674570
news:Hawaii.974892056	ptop:locatedIn	news:UnitedStates.788674570
news:Chicago.147673957	ptop:locatedIn	news:UnitedStates.788674570
news:Arizona.902796844	ptop:locatedIn	news:UnitedStates.788674570
news:Tucson.15022205	ptop:locatedIn	news:UnitedStates.788674570
news:RhodeIsland.427788819	ptop:locatedIn	news:UnitedStates.788674570
news:NewYorkCity.603711547	ptop:locatedIn	news:UnitedStates.788674570

Figure 5.7.: News domain ontology: semantic relations among instances

have personal versions of the model. As it was proposed in our earlier work [83], the model allows users to define own entities and entity relations. For user-defined versions, the model leverages the notion of *ConceptScheme* defined in the SKOS[15] (Simple Knowledge Organization System) standard. The model contains a *skos:ConceptScheme* for each user. All concepts contributed by users are included into their concept schemes through an assertion displayed in Listing 5.1.

```
<UserConcept> skos:inScheme <UserScheme> .
```

Listing 5.1: User-defined concepts

User-defined relations are also defined through *skos:inScheme* construct displayed in Listing 5.2.

```
<UserScheme> skos:hasTopConcept <UserConcept> .
<UserConcept> skos:broader <AnotherUserConcept> .
<AnotherUserConcept> skos:inScheme <UserScheme> .
```

Listing 5.2: User-defined relations

This structure enables defining user own hierarchies and inter-connecting concepts in a way that corresponds users own representation of the world.

5.3.2. Domain Modeling Service

The domain modeling service (DMS) provides operations for managing the domain knowledge and enabling access to it for other subsystems and graphical user inter-

[15] `http://www.w3.org/TR/skos-reference`

faces. The service supports three types of operations: query operations, inference operations, and modify operations.

- *Query operations* are performed to retrieve content of the domain knowledge model. The service includes query operations for getting instances by URI, by full name, or by first letters of the name (used in GUIs for auto-complete function). It also includes operations for retrieving all properties and values of an instance and retrieving all instances of a concept. Additionally, it allows retrieving personal versions of domain ontology created by individual users.

- *Reasoning operations* leverage semantic constructs of the domain ontology to retrieve its content. More specifically, they use semantic relations among instances and classes to retrieve relevant content. The service includes reasoning operations for getting parent concepts of a concept, getting sub-concepts of a concept, and getting related concepts. Also, it includes operations for getting related instances.

- *Modify operations* are performed to add, edit, and delete content in the domain ontology. In the current implementation of the domain modeling service, the modify operations are performed only at the instance level. The service includes operations for adding new instances, adding and modifying relations of existing instances, and removing instances from the model. To support automatic generation of domain knowledge, the service supports adding instances and instance relations from the results of semantic annotation of portal content (fulfills Requirement F6: Automatic generation of domain knowledge). As described in Section 5.4.2, the resource management service automatically annotates portal documents using external tools for natural language processing, which extract named entities and relations from text documents. Using a semantic types mapping (Appendix B), the service converts the extracted entities and entity relations into instances and instance relations of the domain ontology and sends them as a bulk to the domain modeling service. For every instance and relation in the bulk, the domain modeling service checks if it exists in the domain ontology. If it does not exist, it makes a corresponding assertion in the model.

Implementation of DMS. Same as other services of the framework, the domain modeling service is implemented as a Java Application and packaged in a Java Archive file (JAR). The operations on the terminological component of domain model (TBox) and are performed using Jena API[16], whereas the operations on the

[16] `http://incubator.apache.org/jena`

assertion component ABox are performed using Sesame API[17]. In order to provide access to the domain modeling operations for components located on servers other than the one hosting the domain model and to enable access from JavaScript-based applications, the framework provides a **restful web service for domain modeling**. The web service is implemented as a Java Web Application and packaged in a Web Archive file (WAR). The WAR package can be deployed on any Java Servlet Container, e.g., Apache Tomcat[18] or IBM WebSphere Application Server[19]. JavaScript Object Notation (JSON)[20] is used for serialization of data generated by the service.

5.3.3. Domain Modeling GUIs

In addition to the automatic updates based on the annotation of portal resources using NLP tools, the framework also enables the user community to update the domain knowledge model (fulfills Requirement F7: User community access to domain knowledge model). Users can contribute to the domain model through two graphical user interfaces. First, users can contribute to the domain knowledge implicitly by providing semantic tags to portal resources through the *tagging interface* proposed in our earlier work [83, 82]. Second, users can indirectly update the domain ontology by adding and semantically connecting items in their user model using the *IntrospectiveViews interface* provided by the User Modeling Unit (Section 6.3). The domain modeling functionalities of the tagging interface and IntrospectiveViews are described in detail in Section 5.4.3 and Section 6.3.5 respectively.

5.4. Resource Management Unit

The resource management unit is responsible for maintaining semantic description of portal content. The description is stored in the *metadata repository* (Section 5.4.1) and is provided either automatically by semantically analyzing portal content or by the user community. The machine-generated description is obtained by processing portal documents with NLP tools and transforming the analysis results to the domain ontology by means of a semantic types mapping (Appendix B). Whereas the user community annotations are contributed through the *tagging interface* (Section 5.4.3), which can be embedded in the portal theme and accompany almost any portal resource. The access to the resource metadata, including the query and edit operations, is provided through the *resource management service* (Section 5.4.2).

[17] `http://www.openrdf.org`
[18] `http://tomcat.apache.org`
[19] `http://www-01.ibm.com/software/webservers/appserv/was`
[20] `http://www.json.org`

5.4.1. Metadata Repository

Metadata repository represents machine-processable semantics of the meaning of portal documents. This semantics is used by other subsystems of the framework for various purposes. For example, the *personalization service* (Section 7.2) uses it to find documents that match interests of individual users. The repository represents metadata as an overlay over the *domain ontology* (Section 5.3.1), i.e., portal documents are described with the vocabulary of the domain model (fulfills Requirement F8: Semantic representation of metadata). More precisely, metadata of a portal document is defined as a set of tuples *(D, P, V, A, R, T)*, where:

- D - document ID
- P - document property (e.g., author, subject, document type, source, publication place, etc.)
- V - actual value the document has on the given property, which can be of a primitive type or URI of an ontology instance
- A - agent that made the annotation (user ID or name of NLP tool)
- R - relevance of the value V on the property P to the document D (defined as a real number in range from 0 to 1)
- T - timestamp of annotation

This representation allows defining not only semantic metadata, e.g., subject or author of the document, but also qualitative metadata, such as user comments and ratings. The representation of semantic metadata using the vocabulary of domain ontology resolves the problems of synonyms, different spellings, polysemy, and scarcity of annotations discussed in Section 3.1.3 (Requirement F8: Semantic representation of metadata). The domain ontology provides necessary constructs to define synonyms and different spellings of the same concept, which can improve the precision and recall of search. Also, it allows inter-connecting related concepts, which can be used for expanding search and retrieving related content.

5.4.2. Resource Management Service

The resource management service (RMS) provides methods for accessing and editing metadata of portal content. The service supports two types of operations: information harvesting operations, query operations, and modify operations.

- *Information harvesting operations* are used for populating the portal with new content. These operations are intended for information aggregation portals

that provide content gathered from external sources. An example of such portals is the news aggregating portal described in Section 5.2.1. RMS provides operations for fetching documents using RSS[21] feeds. RMS reads the content of provided RSS feeds and, for every feed entry, it fetches the document content using the URL provided by the feed entry and saves the content in the portal database. Every newly added document is automatically annotated using the modify operations of RMS. The method for information harvesting is described below.

- *Query operations* are performed to retrieve content and metadata of an individual document or a set of documents. It includes a method for retrieving content of a given document. Also, it provides methods for getting all metadata for a given document ID and methods for getting a set of documents matching certain criteria, e.g., documents created by a certain author or documents on a certain subject.

- *Modify operations* are performed to add, edit, or delete information in the metadata repository. RMS includes operations for inserting metadata of a document and methods for updating and deleting existing insertions. Also, it includes methods for automatic annotation of portal content (fulfills Requirement F9: Automatic generation of metadata). The method for automatic annotation of content is described below.

Automatic Content Harvesting and Annotation

The method for gathering and annotating content in the framework follows our approach to semantic enrichment of social media resources for adaptation proposed in our earlier work [116]. Below we describe this method on the example of a news aggregating portal, which we described in Section 5.2.1. This portal provides a single-point of access to news stories from different media agencies and blogs. The flow of harvesting and annotation of content is illustrated in Figure 5.8.

At the first step, RMS fetches news stories into the content database of the portal. For this, it uses a set of channels for RSS feeds supplied by a portal administrator. Every channel contains feeds on a specific topic, e.g., business news from BBC. In every provided channel, it reads feeds and *fetches* the content of HTML pages located at the URLs provided by feeds. The content of fetched document is then *saved in the content database*.

Afterwards, RMS *annotates* the newly retrieved documents. For automatic annotation, RMS uses external tools for Natural Language Processing capable to extract named entities and relations. In the proof-of-concept implementation, OpenCalais[22]

[21] Really Simple Syndication)
[22] `http://www.opencalais.com`

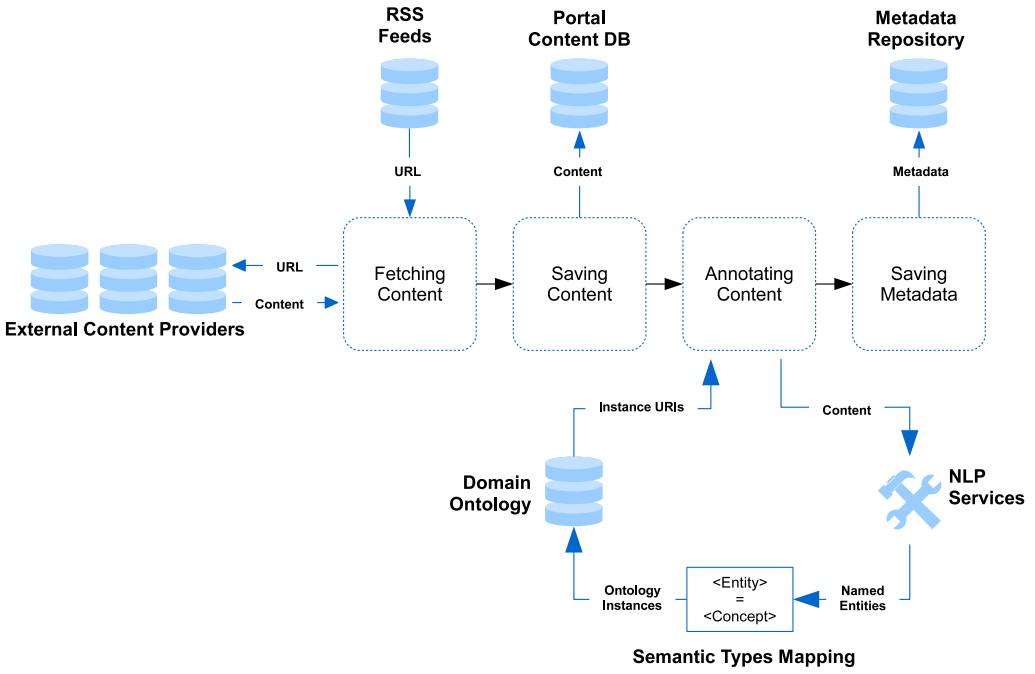


Figure 5.8.: Content harvesting and annotation

is used for named entity extraction. OpenCalais is a web service for semantic annotation of text documents. It extracts entities, events, facts, and relations from textual documents in plain text, HTML, and XML formats. Among others, it extracts such entities as company, technology, product, person, country, city, and many others.

For every entity or relation extracted by OpenCalais, RMS identifies an appropriate concept or object property from the domain model based on the semantic types mapping (Appendix B), which maps the ontology scheme of the NLP tools to the scheme of the system's domain ontology. Once the domain concept is known, the service invokes the domain modeling service to check the domain ontology for existence of a matching instance. To mitigate the problem of ambiguity, the service leverages the entity name, its type, and properties. If a corresponding instance is found, the service checks whether it can update it using the results of semantic processing (e.g. update one of the properties) and then uses URI of that instance to include it into the document metadata. If no instance is found, RMS calls DMS to insert a new instance using the entity name, ontology concept, and any other attribute value or relation returned by the NLP tools. After the instance is inserted, its URI is used to include it into the document metadata.

For each document, the output of the annotation step is a set of URIs referring to the instances from the domain ontology. These instances represent the semantic entities identified in the document. Also, depending on the NLP tool that is used for entity extraction, the output may include the relevance of identified entities to the given document. For each identified instance, RMS generates a tuple according to the format of metadata repository (Section 5.4.1). At the final step, RMS *saves the generated metadata* in the metadata repository.

Implementation

Same as other services of the framework, the resource management service is implemented as a Java Application. In order to provide access to the resource management operations for components located on servers other than the one hosting the content database and metadata repository and to enable access from JavaScript-based applications, the framework provides a **restful web service for resource management**. The web service is implemented as a Java Web Application. JSON format is used for serialization of data generated by the service.

5.4.3. Tagging Interface

Since automatically generated metadata does not always represent the entire meaning of content or does not represent it precisely, it is essential that users can also annotate portal content (Requirement F10: User community access to metadata). The framework enables the user community to contribute metadata for portal content through the tagging interface developed in the context of the master thesis project of Alexander Kreiser [82, 83].

Tagging interface (Figure 5.9) is developed in a Rich Internet Application (RIA) fashion using AJAX technology. The interface can be integrated in a portal through a portlet. This portlet can be displayed at any portal page containing a document for which user-generated annotation is required. The interface consists of three areas: Tag Resource, Suggested Tags, and Browse Tag Space. In the *Browse Tag Space* area users can select the tags they want to assign to a document. They can either select an appropriate tag from the hierarchy representing their entire tag tree or type the tag name in the lookup field. When typing, the interface displays a list of tags matching the string the user has entered, so that the user could select the appropriate one from the list. Once the user has selected a tag from the list, it will be visualized in the tree below the lookup field. In such a way users can explore their tag hierarchies and select either broader or narrower concepts.

The tags selected by the user for annotation are inserted into the *Tag Resource* container. If necessary, users can remove tags from the container or add related tags through the tag context menu displayed in Figure 5.10. Also, in the con-

Figure 5.9.: Tagging interface [82]

tainer they can view additional information about the tags. By hovering the mouse pointer over a tag, the interface will display all spellings of the tag and its definition (Figure 5.11). Once users have inserted all tags they want, they can finish the annotation by clicking the *Tag It!* button. The selected tags will be assigned to the document as metadata.

Figure 5.10.: Tag context menu [82]

Figure 5.11.: Tag information [82]

Support for Domain Modeling

By providing semantic annotations through the tagging interface, users implicitly contribute to the domain knowledge model. The tagging interface allows users not only to assign tags to documents, but also to connect tags semantically. Users can connect tags using the tag context menu shown in Figure 5.10. From the menu, they can chose tags that have a broader or narrower meaning than the selected tag or related in some other way. Also, they can connect two tags in the *Tag Resource* container by dragging one tag onto another. Figure 5.12 shows an example of connecting two tags in a drag-and-drop fashion.

Figure 5.12.: Connecting tags in drag-and-drop fashion [82]

By allowing users to organize their tag spaces into hierarchies and semantic networks, the framework can update the domain knowledge model with valuable semantics. All entities and relations provided by the user community as annotations of content are stored in the domain ontology as instances and instance relations. By performing statistical analysis of the frequency of relations it is possible to induce the most relevant and important relations among instances. This information can be leveraged by the framework to improve the adaptation effects in the portal. It can be used for search query expansion, recommendation of related content, and propagating user interest and knowledge.

5.5. Summary

This chapter introduced a conceptual architecture of the framework for scrutable adaptivity in community-enabled web portals. This architecture was designed based on the requirements identified in Chapter 3. The framework consists of four units, namely, units for domain modeling, resource management, user modeling, and personalization. The units for domain modeling and resource management were described in detail in this chapter. The units for user modeling and personalization are presented in Chapter 6 and Chapter 7 respectively.

Also, this chapter described two application scenarios for the framework: a news aggregating portal and a portal for biochemical literature. Both portals are used across the thesis to describe various components of the framework. They are also used as platforms for evaluation of the framework.

CHAPTER 6

User Modeling Unit

In Chapter 5, we presented a conceptual architecture of the framework for scrutable adaptivity consisting of four units. Also in that chapter, we described its two auxiliary units: the domain modeling unit and the resource management unit. In this chapter, we present the third unit of the framework, namely, the user modeling unit. This unit is responsible for storing, managing, and updating information about individual users. The unit consists of three main components: user model, user modeling service, and IntrospectiveViews interface. The *user model* semantically represents information about individual users. The *user modeling service* provides access and management operations on the user model. Finally, *IntrospectiveViews* provides users an interactive visualization for viewing and editing their user models.

Parts of research presented this chapter were published in our earlier papers. Our approach to user modeling is published in [15, 16]. The first version of the interface for scrutinizing user models IntrospectiveViews and results of the first evaluation of this interface are published in [18]. Also, a patent application [17] has been made for the visualization concept and interaction patterns of IntrospectiveViews. Subsequent versions of this interface and results of further studies are published in [13, 19, 65].

6.1. User Model

The user model provides key information for adaptation, i.e., it stores information about certain features of individual users. The user model proposed in this thesis can be used for modeling different user features, such as user interests, goals, or knowledge. However, due to the time constraints of this research, in the proof-

of-concept implementation, we focus only on user interests. From now on, the framework will be described on the example of a user interest model.

We define user interest as a fact indicating that a given user is interested in a certain concept to a certain degree. The *concept* can denote either a real world object, e.g., company, geographic location, person, etc., or an abstract notion, e.g., area of science, discipline, technology, etc. To fulfill Requirement F1: Semantic representation of user model, the user model is designed as an overlay model, i.e., user interests are represented as an overlay of domain concepts which semantic is defined the ontology-based domain knowledge model (Section 5.3.1).

For each concept, the model stores information about the exact *degree* to which the user is interested in it. The degree of interest is defined as *interest weight* represented by a real number in the range from 0 to 1. Additionally, the model stores an approximated semantic interpretation of the interest weight as *interest status*. The model supports four statuses of interest: interesting, partially interesting, uninteresting, and blocked. A blocked concept is a concept that the user explicitly blocked from being monitored and used for personalization. It means that the system ignores any evidence of the user activity on this concept and does not recommend any content related to this concept.

Also, the user model contains information about the *evidence* of user interest, i.e., the source based on which the system concluded that the user is interested or uninterested in the concept. In community-enabled web portals user interests can be identified from multiple sources. User interests can be derived from the documents that users read, share, bookmark, and annotate. The system can infer new items of interest by propagating interest from existing items to new ones using the semantic relations defined in the domain ontology. Finally, users can define their interests explicitly.

The proof-of-concept implementation of the user interest model supports three sources of interests: log-based updates, inference-based updates, and user explicit updates. Log-based updates are performed based on the evidence of documents that the user has accessed in the portal. Inference-based updates are performed based on semantic relations among instances in the domain ontology. Log-based and inference-based updates are performed automatically by the user modeling service (Section 6.2). User explicit updates are made by the user explicitly through the *IntrospectiveViews* interface (Section 6.3).

Logically, the user model is represented as a set of tuples *(U, C, W, S, E)*, where

- *U* - user identifier.
- *C* - URI of the instance denoting the domain concept the user is interested in. Instances are semantically defined in the domain ontology.

- W - interest weight, i.e., exact degree of interest, represented by a real number in the range from 0 to 1 (1 - interesting, 0 - uninteresting).
- S - interest status represented by a linguistic variable (i - interesting, p - partially interesting, u - uninteresting, b - blocked).
- E - evidence of interest represented by the identifier of the last update entry from the updates log. The update log entry contains evidence of interest for at least one of the three sources (log-based updates, inference-based updates, user explicit updates). The methods for eliciting user interests are described below.

In the proof-of-concept implementation of the framework, the user interest model is implemented as a relational database displayed in Figure 6.1. The *user-model* table contains the current state of the user interest model. For each concept of the given user, it stores the interest level (e.g. interesting) and exact weight of interest (e.g. 0.9). Also, it stores the ID of the last update log entry as a foreign key. This key points to the *user-model-updates* table, which provides detailed information about all updates made in the user model. In particular, it stores the time when the update was made and the evidence of interest. The evidence of interest is the information whether the interest was calculated based on the browsing log, inferred from other items, or was set by the user explicitly. This evidence is represented by the ID of corresponding update log entry from one of the three tables: log-based-updates, inference-based-updates, and explicit-updates.

Each user model update is based on an evidence from at least one source. Therefore, every record in the *user-model-updates* table contains at least one foreign key. However, for certain concepts, the system may have evidence of interest from multiple sources. For example, it may have an evidence of interest calculated based on the browsing log and an evidence of interest that was propagated from related items. In such a case, the record will contain two foreign keys: references to an update record in the *log-based-updates* table and a record in the *inference-based-updates* table. The resulting interest-weight of the record in the *user-model-updates* table is a cumulative weight, which is calculated based on the interest-weight of the log-based update and the interest-weight of the inference-based update. Section 6.2 presents a detailed description of the algorithm for calculating interest weights.

Browsing Log. To achieve automatic updates of the user model based on the reading history of users, the model stores information about the documents users access in the portal. The *resource-log* table stores the IDs of resources that users access along with the timestamp and ID of the portal page on which the resource was accessed. The *concept-log* table contains the concepts that these resources were annotated with at the time of user access. It also stores the information about the

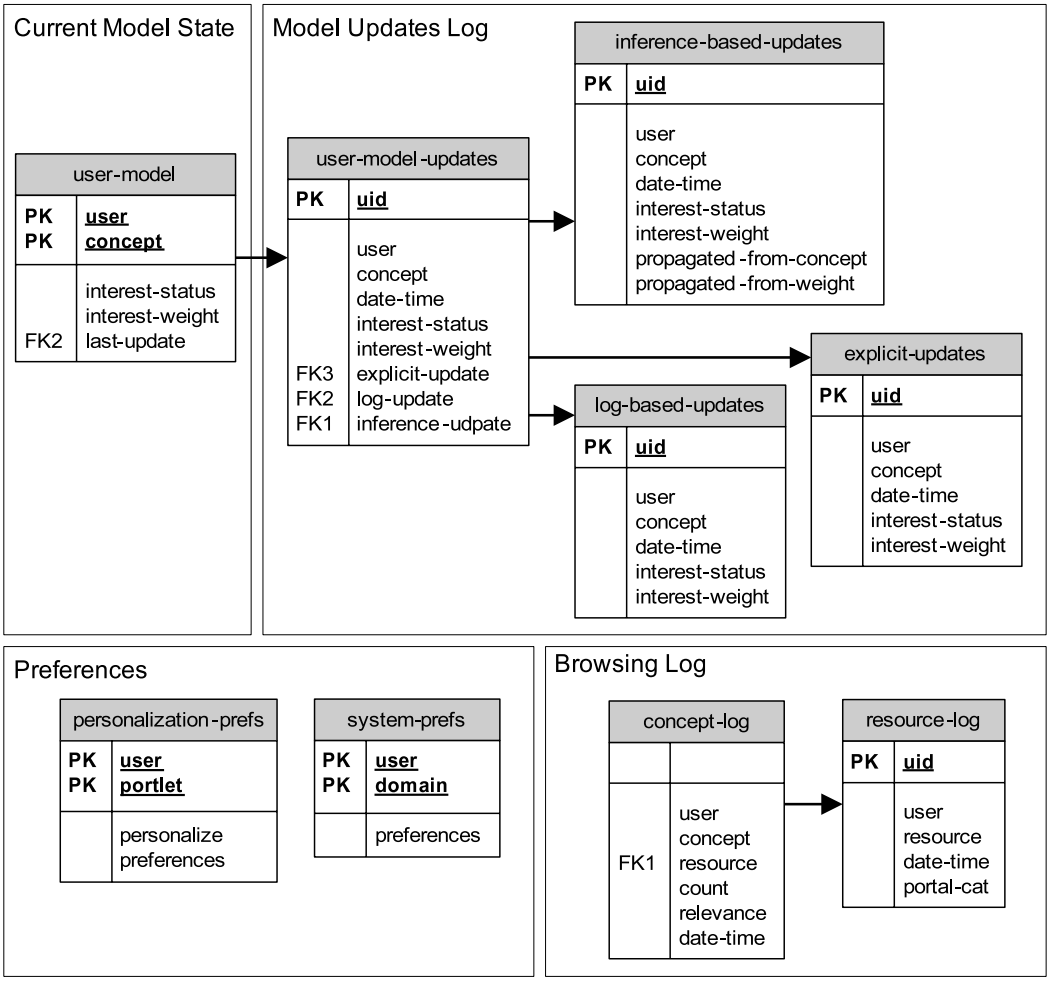


Figure 6.1.: Implementation of the user interest model

frequency of the concept in the given document and its relevance to the document. The information about the concept frequency and relevance is used by the user modeling service (Section 6.2) to calculate the interest weight of the concept.

Personalization and System Preferences. Additionally, the user model stores user preferences on personalization effects in portlets and the system's behavior. As described in Section 7.3, the framework provides a *PersonalizablePortlet* class that can be used for programming portlets with controllable personalization. A personalizable portlet can be in two states: personalized or standard. In personalized state, the portlet can support more than one personalization effect. For example, in a news portlet (Section 5.2.1), news stories can be sorted according to the user's interests, or the most interesting for the user stories can be highlighted, or both. Users may prefer different personalization effects for different portlets. These preferences are stored in the *personalization-prefs* table. For the given user and portlet, it stores whether the portlet must be displayed in personalized or standard view and what personalization effects must be displayed if the portlet is viewed in personalized state. The *system-prefs* table stores user preferences with respect to various aspects of the system behavior. In particular, it stores user preferences with respect to the visualization of user model in the IntrospectiveViews interface (Section 6.3), e.g., what parts of the user model should be displayed, at what zoom level, what color scheme the user prefers, etc.

6.2. User Modeling Service

The user modeling service implements the logic of user modeling and provides access to the user model for the other components of the framework. The service supports three types of methods: query, modify, and modeling methods.

- *Query methods* are invoked to retrieve content of the user model. The service provides a method for retrieving features, e.g., interests, of the given user. It also allows retrieving features matching certain criteria, e.g., all items the user is strongly interested in or all blocked items. Finally, the service provides methods for retrieving user preferences on the personalization and system behavior.

- *Modify methods* are called to alter information in the user model. The service provides methods for adding and removing user interests and changing weight and status of interests. If modify methods are invoked by users through the IntrospectiveViews interface (Section 6.3), they are stored in the user model as user explicit updates. Also, the service provides methods for setting preferences on the personalization and system behavior.

- *Modeling methods* are used for performing automatic updates of the user model. The service implements methods for generating log-based updates and inference-based updates. It also includes a method for merging multiple updates to generate a cumulative interest weight. A detailed description of the user modeling algorithm is provided below.

User Modeling Algorithm

To fulfill Requirement F3: Automatic update of user model, the user modeling service implements operations that allow building and modifying user interest models unobtrusively for users. The service implements two types of modeling operations: log-based updates and inference-based updates. Log-based updates modify the user model based on the user browsing history, i.e., the frequency of semantic concepts describing the portal documents users read. Inference-based updates propagate user interest from one item to another using semantic relations defined in the domain knowledge model. Finally, the service implements methods for merging updates and calculating cumulative interest weight. These methods are invoked if the user model contains an evidence of user interest from multiple sources, e.g., it contains an evidence of interest from the browsing log and an evidence of interest propagated from related concepts. The invocation of automatic update methods is triggered when a user accesses a document in the portal (Listing 6.1).

```
FUNCTION doUpdateTriggeredByDocumentView(user, document)
  doLogBasedUpdate(user, document)
  mergeUpdates(user)
```

Listing 6.1: Automatic updates triggered by *documentView* event

Log-based updates. The method implementing the logic of log-based updates is triggered by an event when a user accesses a document in the portal. Listing 6.2 displays the flow of actions performed by the method. First, the method makes a log entry in the *resource-log* table (Figure 6.1). Then it calls the resource management service (Section 5.4.2) to retrieve the semantic concepts that the document is annotated with and records the returned concepts into the *concepts-log* table. Afterwards, it retrieves all entries for the user from the *concepts-log* table and for each concept computes the term frequency. Based on the term frequency values of the concepts from the log, the method clusters the concepts into three clusters using k-means clustering algorithm. Each cluster corresponds to an interest group denoting one of the three interest statuses supported by the user model, namely interesting, partially interesting, and uninteresting. For each concept in each cluster, the method sets interest status based on the interest group the concept belongs to. Also, it computes the interest weight of the concept based on its relative term

frequency in the cluster and the minimum and maximum interest weights in the interest group. Finally, the method writes the concept into the *log-based-updates* table along with the interest status and interest weight. After the log-based update is completed, the method for inference-based update is called for every concept that has been affected by the log-based update.

```
FUNCTION doLogBasedUpdate(user, document)
  // write document ID into the resource log
  updateResourceLog(user, document)

  // retrieve concepts annotating the document
  documentConcepts = RMS.getDocumentConcepts(document)

  // write document concepts into the concepts log
  updateConceptLog(user, document, documentConcepts)

  // retrieve all concepts from the concept log
  allConceptsInLog = getLogConcepts(user)

  // computer term frequency for each concept
  FOR EACH concept IN allConceptsInLog
    concept.termFrequency = conept.occurrence / allConceptsInLog.size

  // cluster concepts based on the term frequency
  clusters = clusterConcepts(allConceptsInLog, 3)

  // compute interest weights and write updates into the log
  FOR EACH cluster in clusters
    FOR EACH concept IN cluster.concepts
      interestGroup = getInterestGroup(cluster)
      concept.interestStatus = interestGroup.linguisticVariable
      relativeTermFrequency = (concept.termFrequency - cluster.minTermFrequency) /
        cluster.maxTermFrequency - cluster.minTermFrequency
      concept.interestWeight = interestGroup.minInterestWeight +
        (interestGroup.maxInterestWeight - interestGroup.minInterestWeight) *
        relativeTermFrequency
      recordInLogBasedUpdatesLog(user, concept)

  // do inference-based updates for affected concepts
  updatedConcepts = getConceptsFromLastLogBasedUpdate(user)
  FOR EACH concept IN updatedConcepts
    doInferenceBasedUpdate(user, concept)
```

Listing 6.2: Algorithm of log-based updates

Inference-based updates. Inference-based updates are triggered by log-based updates. Listing 6.3 displays the flow of actions performed by the method. First, the method calls the domain modeling service (Section 5.3.2) to retrieve related con-

cepts. For each related concept, it checks if the concept is present in the user model of the given user. If true, the method writes its interest weight into an array. After interest weights of all related concepts are recorded into the array, the method computes a propagated interest weight of the concept. It multiplies the highest interest weight from the array by an interest propagation coefficient (a value that can be configured by the administrator). Using the computed interest weight, the method determines the interest status of the concept. Finally, it writes the concept in the *inference-based-updates* table (Figure 6.1) along with the interest status and weight and the information about the concept that the interest was propagated from.

```
FUNCTION doInferenceBasedUpdate(user, concept)
  // get a list of related concepts from the domain model
  relatedConcepts = DMS.getRelatedConcepts(concept)

  // get interest weights for related concepts
  FOR EACH relatedConcept IN relatedConcepts
    IF UMS.isInUserModel(relatedConcept)
      weights.put(relatedConcept.interestWeight)

  // compute propagated interest weight
  concept.interestWeight = weights.getMax() * interestPropagationCoefficient

  // determine interest status
  concept.interestStatus = getInterestStatusByWeight(concept.interestWeight)

  // write updates into the log
  recordInInferenceBasedUpdatesLog(concept)
```

Listing 6.3: Algorithm of inference-based updates

Merging updates. The method for merging updates is called every time after the completion of either log-based or inference based updates. This method implements the logic of computing a cumulative interest weight combining evidence of interest from multiple sources. Listing 6.4 displays the flow of actions performed by the method. First, the method generates a set of user interests by a union of items from the user model, log-based updates, and inference-based updates. For each item in the set, the method retrieves evidence of interest from the three sources, namely, from log-based updates, inference-based updates, and user explicit updates. If a user explicit update is available, the method assigns the interest weight and status of this update to the item since the user explicit changes have the highest priority. If no explicit update is found, the method takes the evidence from the log-based and inference based updates. If both updates are available, the method takes the average interest weight. Else, it takes the interest weight of either available update. Based on the interest weight, the method determines the interest status

and assigns it to the item. Finally, it stores the changes in the user model and makes a log entry in the *user-model-udpates* table.

```
FUNCTION mergeUpdates(user)
  // get interests from the actual user model
  interests = UMS.getInterests(user)

  // add new interests identified from the browsing log
  logBasedInterests = UMS.getLogBasedInterests(user)
  FOR EACH logBasedInterest IN logBasedInterests
    IF logBasedInterest IS NO IN interests
      interests.add(logBasedInterest)

  // add new interests propagated from other items
  inferenceBasedInterests = UMS.getInferenceBasedInterests(user)
  FOR EACH inferenceBasedInterest IN inferenceBasedInterests
    IF inferenceBasedInterest IS NOT IN interests
      interests.add(inferenceBasedInterest)

  // get a set of explicitly added interests
  explicitlySetInterests = UMS.getExplicitlySetInterests(user)

  // merge sources of interests
  FOR EACH interest IN interests
    IF interest IS IN logBasedInterests
      interest.logBasedUpdate = logBasedInterests(interest).update
    IF interest IS IN inferenceBasedInterests
      interest.inferenceBasedUpdate = inferenceBasedInterests(interest).update
    IF interest IS IN explicitlySetInterests
      interest.explicitUpdate = explicitlySetInterests(interest).update
    IF interest.explicitUpdate IS NOT NULL
      interest.interestWeight = interest.explicitUpdate.interestWeight
      interest.interestStatus = interest.explicitUpdate.interestStatus
    ELSE
      IF interest.logBasedUpdate IS NOT NULL
          AND interest.inferenceBasedUpdate IS NOT NULL
        interest.interestWeight = AVG(interest.logBasedUpdate.interestWeight,
          interest.inferenceBasedUpdate.interestWeight)
      ELSE IF interest.logBasedUpdate IS NOT NULL
        interest.interestWeight = interest.logBasedUpdate.interestWeight
      ELSE
        interest.interestWeight = interest.inferenceBasedUpdate.interestWeight
      interest.interestStatus = getInterestStatusByWeight(interest.interestWeight)

  // update the actual state of user model and make log entries
  updateUserModel(interests)
  recordInModelUpdatesLog(interests)
```

Listing 6.4: Algorithm of merging updates

Implementation

The user modeling service is implemented as a Java Application. For clustering of user interests, the application uses Weka API[1], an open source toolkit for data mining and machine learning. The other services of the framework located on the same server communicate with the UMS by making Java calls to its public methods.

For access from JavaScript- or Applet-based GUIs, e.g., the IntrospectiveViews interface, the framework provides a **restful web service for user modeling**. The web service is implemented as a Java Web Application and deployed on the portal server. JSON format is used for serialization of data generated by the service. The web service allows calling any public method of UMS by making an HTTP request to the portal server. In case of query methods providing user features enriched with machine-processable semantics, requests are made according to the following syntax:

```
http://<server>/ums/<domain-model>/<method>/?[param=value]
```

Along with the method name and parameters (if applicable), requests must contain a domain model name. This tells what domain knowledge model should be queried to retrieve the semantics of user features.

6.3. IntrospectiveViews

To fulfill Requirement F2: User access to user model, the framework provides a graphical user modeling interface IntrospectiveViews. IntrospectiveViews is an interactive visualization of semantic feature-based user models. It displays user models in a comprehensible way and allows altering them. To fulfill Requirement Q3: Usability of scrutinizing tools, the interface was designed following Shneiderman's Visual Information Seeking Mantra [118]. This interaction pattern was chosen because it allows visualizing large and complex information sets in an understandable and easy-to-use way. Shneiderman's principle of interaction with large information sets can be summarized as: "overview first, zoom and filter, then details-on-demand". This principle is reflected in the interaction pattern of IntrospectiveViews as follows. The interface offers users an overview over all items present in the model. It allows zooming into different parts of the model to get a better view and allows filtering items according to different criteria to minimize the amount of items displayed. Also, it provides detailed information of an item upon user request and allows revealing relationships among items. Finally, the interface allows users to add, edit, and delete items in their user models.

IntrospectiveViews is implemented as a Java Applet. Hence, it can be easily embedded into a portal page and run in any web browser. For the communication with

[1] `http://www.cs.waikato.ac.nz/ml/weka/`

other components of the framework, the interface uses restful web services provided by the framework. It uses the web service for user modeling (Section 6.2) to query and edit the user model. For retrieving and editing content in the domain knowledge model, the interface uses the web service for domain modeling (Section 5.3.2). Also, it uses the web service for resource management (Section 5.4.2) to retrieve related portal documents when the user requests detailed information of a certain item in the user model.

In order to ensure that IntrospectiveViews provides all necessary functions of a scrutable user modeling interface and allows users to interact with it in an easy-to-use way, it was designed iteratively following user-centric design practices. In the first iteration (Section 6.3.1), the interface provided support for all seven categories of information seeking tasks defined by Shneiderman in [118], namely, overview, zoom, filter, details-on-demand, relate, history, and extract. We conducted a user study (Section 6.3.2) to evaluate the usefulness of these tasks for scrutable user modeling and the usability of implementation of these tasks in IntrospectiveViews.

Based on the results of the usefulness and usability evaluation, we redesigned the interface. In the second version of IntrospectiveViews (Section 6.3.3), we improved the functions that were rated as most important, removed the less important functions, and redesigned the functions on which the usability was rated low. The second iteration of IntrospectiveViews was also evaluated in a user study (Section 6.3.4). The goal of that study was to evaluate not only the inherent usability (pragmatic quality), but also the hedonic qualities and perceived appearance of the interface.

The findings of the second study helped us to improve the perceived usability and appearance of the interface. The improved design was implemented in the third version of IntrospectiveViews (Section 6.3.5). Also in the third version, we introduced functions for domain modeling (Requirement F7: User community access to domain knowledge model). This version allows users to implicitly contribute to the domain knowledge by editing their user models. It allows adding new instances into the domain model, creating user own semantic classes, and defining relations between items.

In the rest of this section, we present the three iterations of IntrospectiveViews. We describe them using user interest models for three different domains, namely a news aggregating portal, a social network, and a portal for biochemical literature. In this section, we also report on the findings of the two user studies that we conducted.

6.3.1. IntrospectiveViews v.1

We implemented the first version of IntrospectiveViews using the example of a user interest model for a news aggregating portal (Section 5.2.1). Figure 6.2 displays the visualization of interests of a user in the news aggregation portal. The interface

displays interests as labels on a circular surface consisting of three color zones. The positioning of items on the surface, i.e., the distance from the center, is determined by the interest weight of the item, i.e., the closer an item appears to the center, the higher interest it represents. The font size of items denotes their frequency in the user browsing log: the items that users encounter often appear bigger in relation to the less frequent ones.

Figure 6.2.: IntrospectiveViews v.1

The three color zones of the circle represent the three interest statuses supported by the user interest model, namely interesting, partially interesting, and uninteresting. The color scheme of the zones is chosen according to the hot-and-cold metaphor. The hot zone, shown red, contains items that the user is strongly interested in. The cold zone, shown blue, contains the items that the user is not interested in. The purple zone contains the items that the user is somewhat interested in. The border areas of the zones are painted in a gradient color to denote the fuzziness between them, i.e., uncertainty of interest degree.

User interests are grouped into circular sectors by the semantic class they belong to, i.e., the ontology class of the respective instance in the domain knowledge model.

The sectors displayed outside of the circle. They are distinguished by different shades of gray and identified by the class name. The model displayed in Figure 6.2 contains items of six classes: *Company, Province, Country, Organization, Person, City*. We introduced this grouping into the interface to help users to see the semantic structure of their user models. More precisely, it allows seeing items of the same class placed close to each other.

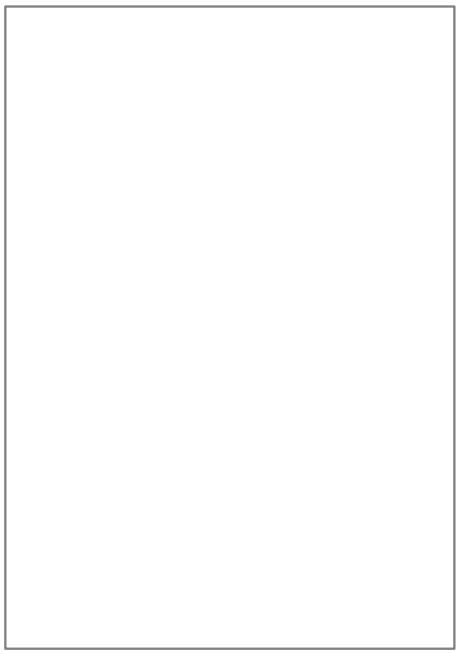
As it was mentioned above, IntrospectiveViews implements the information seeking mantra of Ben Shneiderman [118]. It supports all seven information visualization tasks, namely overview, zoom, filter, details-on-demand, relate, history, extract. Users can get an **overview** of the entire user model by setting the zoom level to 100% or lower. They can **zoom** into a certain part of the model to get a better view of items in that part. E.g., a user can zoom in the Company sector to get a detailed view of all companies in the user model. In an enlarged view, the user can navigate through the collection of items by dragging the surface in the respective direction.

Also, the interface allows users to **relate** items in the user model in a number of ways. It allows relating semantic classes by the number of items they contain. These numbers are encoded in the span of the circular sectors representing semantic classes. Items can also be related by the size of the interest group they belong to. This size is encoded into the radius of colored rings containing items of specific interest groups.

The interface supports a number of **filtering** options. Items can be filtered by semantic class and by interest group. For instance, a user can display the most interesting companies, organizations, and people by selecting the corresponding checkboxes in the *Types* window and the *Interested* checkbox in the *Interest Groups* window. In this manner, users can control the amount of items on the screen, which is especially important in very large user models.

With respect to the **details-on-demand**, IntrospectiveViews can provide different types of additional information on items in the user model. For instance, by hovering over an item, the interface will visualize the semantic relations of this item to other items in the user model (Figure 6.3.a). Upon a click on a label, an infobox will be displayed (Figure 6.3.b). The infobox contains a description and a picture of the item. To fulfill Requirement F4: Explain sources of information in user model, the infobox displays also the evidence of user interest, i.e., it shows whether the interest was calculated based on the browsing log, propagated from related items, or was set by the user explicitly. Also, the infobox displays a list of related documents that the user has not seen yet and a list of related documents that the user has already viewed. By clicking a document hyperlink, its content will be displayed on a portal page.

In addition to viewing user models, the interface enables users to **edit** them. In order to change interest degree in a certain item, the user can simply drag it to the



(a) (b)

Figure 6.3.: Details-on-demand in IntrospectiveViews v.1: (a) relations, (b) infobox

corresponding color zone. To decrease the interest degree, the user drags the item to the edge of the circle. To increase it, the user drags the item to the circle center. By dragging an item that has related items, the interest of related items will be automatically updated and their labels will be repositioned accordingly.

Also, users can add new items into their user models through the *Add new term* window (Figure 6.2). The new item will be placed in the circle (by default in the red ring). Users can also remove items from the model. This can be done by dragging an item onto the recycle bin. In this case, the interest status of the item will be set to *blocked* (for more information on interest status, see description of user model in Section 6.1). It means that the system will not track the user browsing history on this item anymore and will not use it for personalization.

In the first version of IntrospectiveViews, we also provided a mockup-based implementation of two further tasks for information visualization: **history** and **export**. The *History* slider displayed on the bottom of the interface would allow users to get a retrospective view on the user model, i.e., see what interests they had in the past. Though the *Export* window, users would be able to extract a certain part of the model and save it in a file on the disk.

6.3.2. Evaluation of Usefulness and Usability

To evaluate the usefulness and usability of IntrospectiveViews, we conducted a formative evaluation of the interface. The results of this evaluation are published in [18]. For this study, we invited 26 participants: 15 of which were experts in at least one of three areas of interest (user modeling and adaptation, information visualization, and usability), whereas the other 11 were non-expert users. Figure 6.4 displays the experience of subjects from both groups with respect to the three areas of interest.

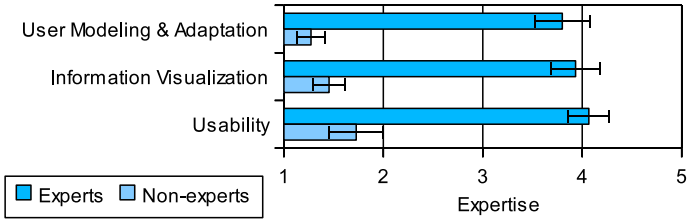


Figure 6.4.: Subjects of the user study of IntrospectiveViews v.1

The aims of the evaluation were twofold. First, when designing IntrospectiveViews, we followed Shneiderman's recommendations for information visualization [118]. Our aim was to determine how much importance users actually put on certain features of a scrutable user model: Would they like an overview? How important would it be to filter terms? Do they want to export the model? The findings of this first part of the evaluation are only very weakly coupled to IntrospectiveViews and offer a guideline for anyone developing a scrutable user model. Second, we wanted to evaluate how well IntrospectiveViews met user requirements, i.e., whether this implementation was considered successful and how we could further improve it.

To achieve these goals, the subjects were first introduced to the interface and the problems that it aims to solve. Then they were asked to test the interface on their own and evaluate it using an evaluation questionnaire. In the questionnaire, we listed 24 tasks that the interface can support (Appendix C). The tasks were classified into eight categories: *overview*, *zoom*, *filter*, *details on demand*, *relate*, *history*, *extract*, and *modify*. The first seven categories belong to Shneiderman's task by data type taxonomy for information visualizations [118]. The *modify* category was introduced to cover the edit tasks specific to IntrospectiveViews. 23 out of 24 tasks are functionally implemented or implemented as mockups. Table 6.1 describes the eight categories of tasks and their GUI implementation. Figure 6.5 displays the GUI components of IntrospectiveViews implementing the tasks.

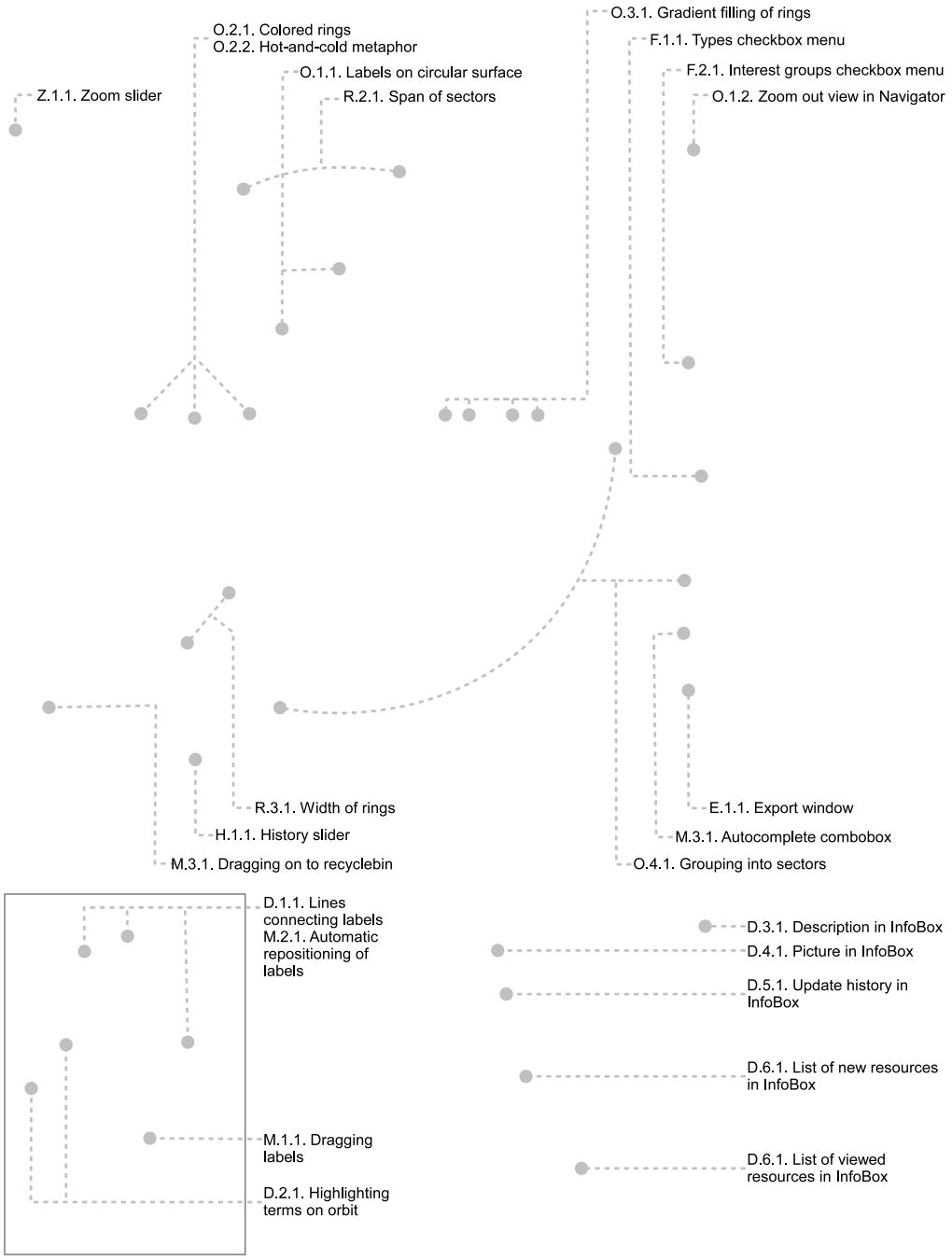


Figure 6.5.: Implementation of tasks for scrutable user modeling by Introspective-Views v.1

Overview	
O.1.	The task of getting an overview of the entire user model is implemented in two ways. First, user interests are represented as labels on a circular surface (***O.1.1***), where the position denotes exact interest degree. Second, a zoomed out copy of the entire model is shown in *Navigator window* (***O.1.2.***).
O.2.	The interest groups available in the user model are represented as colored rings (***O.2.1.***), which are painted according to the hot-and-cold metaphor (***O.2.2.***).
O.3.	The fuzziness between the interest groups is represented by the gradients filling of border areas of rings (***O.3.1.***).
O.4.	To get an overview of the available types of user interests, they can be grouped into circular sectors labeled with the names of types (***O.4.1.***).
Zoom	
Z.1.	Zooming can be performed by dragging the pointer in *Zoom slider* (***Z.1.1.***) as well as by rotating the mouse wheel (***Z.1.2.***).
Filter	
F.1.	Interests can be filtered by type by selecting/deselecting corresponding checkboxes in *Display types* window (***F.1.1.***).
F.2.	Interests can be filtered by interest group by selecting/deselecting corresponding checkboxes in *Display interest groups* window (***F.2.1.***).
Details-on-demand	
D.1.	Related terms can be viewed by pressing mouse left button on a term, which will display connecting lines from the selected term to its related terms (***D.1.1.***).
D.2.	Terms with the same interest can be viewed on the orbit that is displayed when the user presses mouse left button on a term (***D.2.1.***).
D.3.	Textual description of a term can be obtained from the term's InfoBox (***D.3.1.****).
D.4.	Graphical depiction of a term can be seen in the term's InfoBox (***D.4.1.****).
D.5.	Evidence of user interest in a certain term is represented as the update history in the term's InfoBox (***D.5.1.****).
D.6.	Relevant resources that the user has not viewed yet are represented as a list of hyperlinks the term's InfoBox (***D.6.1.****).
D.7.	Relevant resources that the user has already viewed are represented as a list of hyperlinks the term's InfoBox (***D.7.1.****).
Relate	
R.1.	The relation of a term to its frequency in the browsing log is encoded in the term's font size (***R.1.1.***).
R.2.	The relation of a term type to its size, i.e., number of user interests of that type, is encoded in the sector's span (***R.2.1.***).
R.3.	The relation of an interest group to its importance is encoded in the width of the ring containing the terms of that interest group (***R.3.1.***).
History	
H.1.	View on a snapshot of the user model in the past can be obtained by setting the pointer in the *History slider* to the desired date (***H.1.1.****).

Table 6.1.: Questionnaire (elements marked with * were implemented as a mockup)

Extract	
E.1.	The entire collection of user interests or a selected part of it can be exported into a file on disk by specifying the corresponding export options in the *Export* window and clicking *Export* button (***E.1.1.****).
Modify	
M.1.	Interest in a term can be increased or decreased by dragging the term's label closer to or further from the circle center (***M.1.1.***).
M.2.	Interest in related terms is changed automatically when the user drags a label of term that has related terms (***M.2.1.***).
M.3.	New terms of interest can be entered into the user model by typing them in the autocomplete box in *Add New Term* window (***M.3.1.***).
M.4.	Terms can be blocked from being tracked and used for adaptation by dragging them on to *Recyclebin* (***M.4.1.***).
M.5.	Modify the color scheme of the rings (the task is not implemented).

Table 6.1.: Questionnaire (elements marked with * were implemented as a mockup)

For each task, the participants were asked to cast two votes. First, how important did they find the task. This importance was rated on a 1-5 scale: from 1-not at all important to 5-very important. Second, how usable is the implementation of the task in IntrospectiveViews. The usability of the GUI implementation was rated on a 1-5 scale: from 1-not at all usable to 5-very usable. Additionally, the participants could leave free-text comments either regarding the tasks, or implementation, or the interface in general.

Results

Key goals of the evaluation were to assess the usefulness of the interface by examining the importance of the tasks it can support and to evaluate the usability of the GUI elements implementing the tasks. As it can be seen from Figure 6.6, the overall feedback regarding the importance of tasks and the usability of GUI elements was very favorable. The participants of both focus groups provided a number of valuable free-text comments. The rest of this section details these results.

Overview. All tasks in the overview category received relatively high importance. Especially interesting was that the task of getting an overview of available types in the user model received the highest importance in its category. The participants of both focus groups confirmed our hypothesis that organizing interests by type can improve perception of the entire model. With respect to the usability of the implementation, representing user interests on a circular surface consisting of colored rings filled out according to the hot-and-cold metaphor appeared to be very intuitive and comprehensible for both experts and non-experts. However, despite the task of getting an overview of types was rated as very important, its imple-

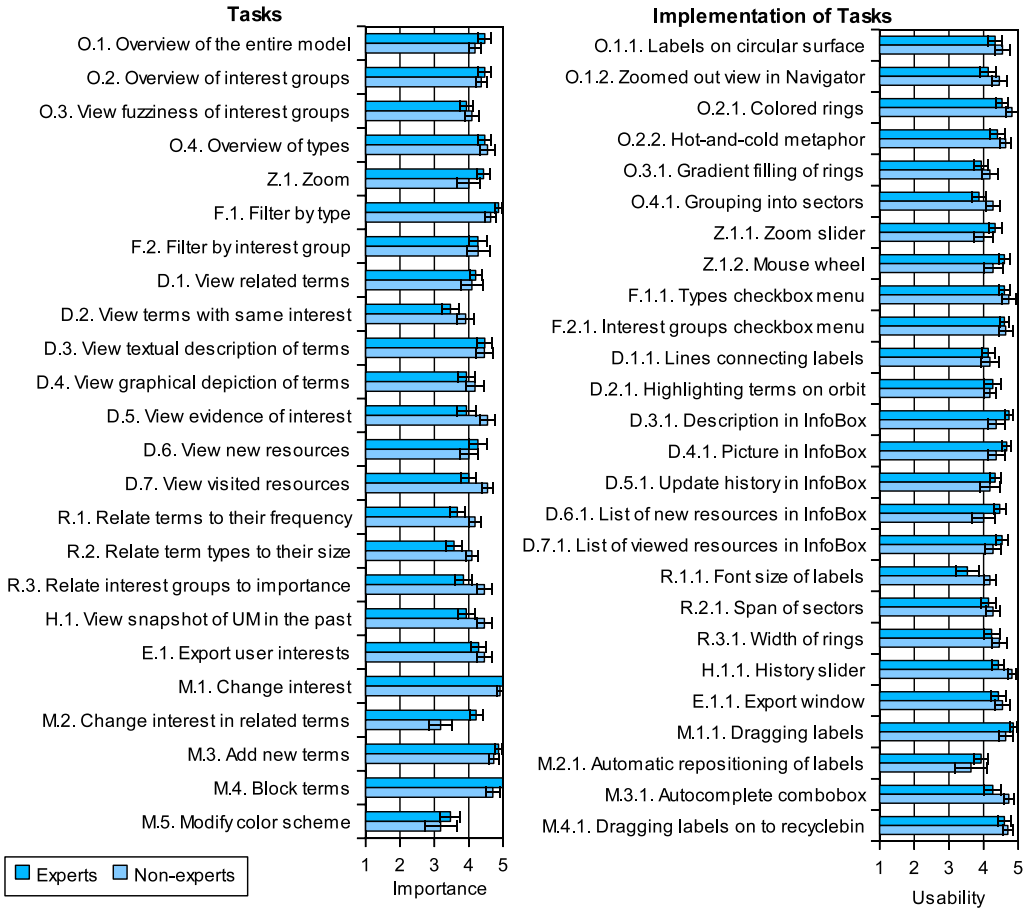


Figure 6.6.: Rating of importance of tasks (left) and usability of implementation (right)

mentation, i.e., grouping into sectors painted in different shades of grey, received the lowest usability in its category. In the free-text comments the participants of both groups mentioned that the information about types is very important, hence it must be more prominent. Experts suggested making the sectors visible not only outside of the circle, but also within it.

Zoom. The importance of zoom task was differently rated by experts and non-experts. The majority of experts considered the task as substantially important, whereas the average rating of the task's importance by non-experts appeared to be considerably lower than the experts' one. However, the usability rating of the two methods implementing the zoom task, i.e., Zoom slider and rotating the mouse wheel, was very favorable.

Filter. The average rating of both filtering tasks, filter by type and filter by interest group, is above 4, which indicates their high importance. The usability of their implementation is also rated very high. However, there was a useful suggestion from an expert on improving the usability of the filtering features. The expert suggested adding animation effects to smoothen the transition of terms when one of the filtering options is changed, e.g., smooth expansion or shrinking of sectors.

Details-on-demand. Four of the tasks in this category received relatively high importance. These are: viewing related terms, viewing textual description of terms, viewing new resources about a term, and viewing visited resources about a term. However, regarding the last two tasks, a number of respondents mentioned that their importance strongly depends on the application. For instance, one expert highlighted high importance of the task of viewing visited resources in a website that does not have a fixed navigation topology, e.g., YouTube, where it is relatively difficult to find the previously viewed content. The usability of implementation of the tasks in this category was rated good. An interesting observation is the relatively high difference between the rating of experts and non-experts with respect to the implementation of the two tasks of viewing resources about a term. Non-experts rated usability lower then the experts. In the free-text comments some of them suggested having a structure within the list, e.g., grouping by topic.

Relate. The overall importance of the relate tasks is considerably lower then the importance of tasks in other categories, in average between 3-somewhat important and 4-important. With respect to the usability of their implementation, encoding the size of a term type (number of interests of that type) into the sector's span was rated as very intuitive and usable. So was rated the encoding importance of interest groups into the ring's width. However, encoding the term's frequency into the label's font size received a relatively low rating of usability. In the free-text comments, many respondents mentioned that it is unintuitive and misleading use of font size. For them, bigger font size means higher interest, which in the current implementation is not the case. It could be that uninteresting terms appear large (see example of BBC explained in Sect. 6.3), whereas some interesting ones can appear small.

History. The average rating of the task of viewing a snapshot of the user model in the past is above 4-important and the usability of its implementation, history slider, was also rated very high. In the free-text comments, one of the experts suggested adding a bookmarking feature to the slider, which would allow the user to bookmark certain time frames, e.g., when she was interested in Italy, and jump to this frame later on by simply clicking on the bookmark.

Extract. The importance of exporting user model was rated very high. Many respondents considered this task useful and practical if many websites support this format. The implementation, Export window, received good rating of usability.

Modify. Three modify tasks were rated as very important: change interest, add new term, and block terms. The importance of other two tasks, change interest in related terms and modify color scheme, received relatively low importance. Regarding the automatic change of interest in related terms, many respondents mentioned that the relations among terms captured by the system do not always reflect the relations that influence their personal interest. Except for the task of changing interest in related terms, the usability of implementation of other tasks received very high rating. Most respondents rated the implementation of the interest change task, i.e., dragging labels closer to or further from the circle center, as very usable. An interesting suggestion on improving the usability of adding new terms was made by one of the experts. The expert suggested implementing this feature in a way that new terms could be added in a specific place of the model by making right click on that place and selecting terms from a popped-up menu.

General Comments. A number of participants said that it is important for them to be able to organize the terms in their user models according to their own categories. Some of them would also like to be able to define their own relations among terms. Regarding the overall usability of the interface, a number of experts requested such features as a search box for quickly jumping to a certain term and an undo feature.

6.3.3. IntrospectiveViews v.2

The results of usefulness and usability evaluation of IntrospectiveViews v.1 provided valuable insights on the importance of functions for scrutable user modeling and the usability of their implementation in the IntrospectiveViews interface. Also, they give useful suggestions and interesting ideas on improving and extending the interface. Based on the collected feedback, we redesigned IntrospectiveViews. Figure 6.7 displays the second version of the interface. It visualizes an ontology-based user interest model for a social network. This model represents objects and concepts users in a social network deal with, i.e., friends, groups, organizations, geographical places, topics, etc. In this section, we present the changes made in the second version with regard to the results of the evaluation of IntrospectiveViews v.1.

Overview Tasks

The results of the usefulness and usability evaluation show that users deem very important to have an overview of available semantic types in the user model. Many spoke out that they want to see the semantic types of displayed items at all time. To achieve that, we introduced a function for highlighting active semantic types. When a user hovers over the circular surface, the highlighting function detects the active sector, brings all items and the sector itself to the foreground, and dims other

Figure 6.7.: IntrospectiveViews v.2

sectors and their items. It also highlights the semantic type of the active sector in the *Types* window. This gives users information about the semantic grouping of items at all times and allows easily find all items of a specific type.

Relate Tasks

One of the problems in the *Relate* category was the encoding of term frequency computed from the browsing log in the font size of labels representing user interests. Many subjects of the evaluation mentioned that it is a rather unintuitive to display uninteresting items in large font just because they occur frequently in the browsing log. In the second version, instead of the term frequency, the font size of items encodes their interest weights, which is also encoded in the position of items. As a result, all interesting items appear big in the center and less interesting items appear smaller towards the edge. This approach allows users to find the most important information quicker and easily.

Modify Tasks

The most serious problem in the *Modify* category was the task for automatic propagation of interest change. Most subjects spoke out that the relations among items in the user model do not always reflect their personal associations. Therefore, they do not want that an interest change initiated by the user is automatically propagated to all related items. To fix this problem, we removed the function for automatic propagation of interest changes by users. The function for interest propagation (inference-based update) implemented by the user modeling service (Section 6.2) is triggered only by the log-based updates. Inference-based updates are invoked only for those items for which the user model does not have any evidence of explicit changes by users. This approach, on the one hand, ensures the predictability of user model updates initiated by users and, on the other hand, compensates the scarcity of information in the user model by identifying new items of interest based on the domain concepts related to the ones users read about in the portal.

Another change in the *Modify* category is an improvement of the function for adding new items in the user model, which was made by an expert in the evaluation of IntrospectiveViews v.1. The expert suggested enabling users to add new items into a user-defined place in the model. We incorporated this suggestion by introducing a context menu in the interface. New items can be added into a certain location on the circular surface by making a left mouse click on that place and selecting the *Add interest here* item from the context menu. This command opens the *Add interest* window (Figure 6.8), through which the user can select an item to add. Note: the list of items in the auto-completion field of the window is determined by the sector in which the context menu was requested. The field displays only items of the semantic class that the selected sector represents.

Figure 6.8.: Adding items in IntrospectiveViews v.2

Also, we introduced a context menu for items (Figure 6.9). Through this menu users can obtain additional information about items, set privacy settings (applies to shared user models), and delete items. This approach provides users more control

over the information in the user model and allows manipulating it in a more direct manner.

Figure 6.9.: Item context menu in IntrospectiveViews v.2

6.3.4. Evaluation of Usability and Hedonic Qualities

In our previous study (Section 6.3.2), we evaluated the usefulness and usability of IntrospectiveViews as a generic interface for scrutable user modeling. The results of that study showed which features users deem important and whether they are sufficiently intuitive and easy-to-use. Based on the results, we redesigned the interface. We removed the features that were deemed unnecessary or disturbing and redesigned the important ones to make them easier to find and use. However, IntrospectiveViews must not only have good usability (Requirement Q3: Usability of scrutinizing tools), but it must also be motivating, engaging, and visually attractive (Requirement Q4: Foster curiosity and engage users). To evaluate these properties, we conducted a study of IntrospectiveViews, in which we included not only the techniques evaluating the inherent usability (pragmatic quality), but also the methods evaluating hedonic qualities and the outward appearance. The evaluation consisted of an expert evaluation and a user study. In this thesis, we report on the user study only. The user study was published in [13]. For the complete description of the whole study, including the expert evaluation, refer to the master thesis of Tobias Hennig [65] (in German).

The user study took place at the Usability Lab of Erfurt University of Applied Sciences. For this study, we used IntrospectiveViews v.2 and a user interest model for a social network. It was conducted with 10 subjects: 5 male and 5 female, in age between 20 and 23. All subjects had medium or high level of technical knowledge. While we are aware that this group is rather homogenous, we believe that it represents typical users of many social networks.

In the study, we used seven techniques for usability testing, namely, task-based experiment, audio and video recording, mouse and eye tracking, interviewing, and a questionnaire. We chose the combination of these techniques in order to find as many usability problems as possible, to better understand why users experience

these problems, and to find solutions to them. The task-based experiments reveal how successful users can achieve their goals with the interface and what difficulties they experience. Filming, eye tracking, and click stream recording during the task completion help the usability experts to perform a more detailed analysis of the user's interaction. Recording the user's eye movement can be especially helpful for answering why the user experiences problems while completing a certain task, e.g., if the user is searching for the required control element on a wrong toolbar. Interviews and questionnaires are helpful for getting the user's subjective reaction and opinions. They help to find what the users like and what they do not like. Also, in interviews the participants can provide valuable insights into how to improve the system.

The course of the study was organized as follows: After a short introduction, the subject was asked to sign a consent form and fill out a personal details form. Then the person was asked to complete eight tasks using the interface. During the completion of tasks, the subject was filmed using audio/video recording equipment. Also, we recorded the participant's gaze and mouse movement. Afterwards, the subject was interviewed and asked to fill out an evaluation question. The rest of this section provides a more detailed description of each method and its results.

Task-Based Experiment

In general, task-based experiments are conducted to test how intuitive an interface is. To measure this, the participants are given specific tasks to solve using the interface. For our experiment we identified eight tasks representing the most common goals that users may want to achieve working with IntrospectiveViews. During the experiment each user had to complete each of these tasks using the interface.

Task 1. Acquaint yourself with the interface. Guess what purpose it has and speak out on it. The goal of this task was to determine whether users can understand the purpose and function of IntrospectiveViews without any introduction or manual. To describe the first impression, the participants used such words as: globe, tag-cloud, portal, diagram, eye, and mindmap. Having worked with the interface for a little, most subjects noticed that items are radially positioned on the circle by interest and grouped into slices by type. Also, they relatively quickly understood the functions of the three panels located on the right side, i.e., *Navigator*, *Interest Groups*, and *Types*. With respect to the purpose of the interface, the subjects either had no idea at all or expressed themselves very vaguely about it. However, it is important to take into consideration that the interface was not launched from a real social network, which makes it more difficult to guess its purpose. Nevertheless, an important observation here is that most participants understood the meaning of the way the items are displayed on the surface and noticed the affordability to interact with them. After the subject had completed this

task, the purpose and possible applications of the interface were explained by the evaluator.

Task 2. Show all persons you are interested in. The goal of the second task was to determine whether users comprehend the filtering options of the interface, namely, filter by interest degree and filter by type. After having acquainted themselves with the interface, most participants completed this task without problems.

Task 3. View privacy settings for an item. Here, we aimed to determine whether users can easily find out how to access the privacy settings of items. Most subjects had problems with this task. As the first attempt, many clicked the *Settings* button on the toolbar, which was supposed (was not implemented at the time of study) to show the global settings of the system. After failing to complete the task that way, almost all participants tried to find the item on the surface by minimizing the number of items on the screen using the filter by type function. Once the item was found, all participants except one made right click on the item and selected the appropriate menu item from the popped-up context menu (Figure 6.9). One person tried to access the settings by making double click on the item. From this task, we have learned that performing a task on a specific item might be complicated by the difficulty to find the item on the surface. Hence, the interface should provide a better means to quickly find items by name. This can be, e.g., a search box or displaying the full list of items in alphabetical order underneath of the corresponding types in the *Types* window.

Task 4. Find out how the system determined your interest in an item. As the first step, the subjects searched for the given item, for which all but two subjects filtered out items of the irrelevant types and then found the given one. Afterwards, they opened its context menu by making right click. However, in the context menu, only two subjects selected the right item, *Your Interest*. All others could not find it from the first try, but simply were checking all menu items one after another. This means that the *Your Interest* label is not self-explanatory and understandable, hence should be renamed or else the information about the user interest should be accessed through some other means.

Task 5. Display relations of an item. Most participants successfully completed this task by making right click on the given item and selecting the corresponding command from its context menu. Visualizing semantic relations among items was appraised by most subjects as a useful feature. However, a number of them complained that the connecting arcs disappear when an item is moved. This means that the interface should provide a better control on visualizing the relations and allow the user to control not only the turning on, but also the turning off of the lines.

Task 6. Change interest degree in an item. In IntrospectiveViews, this task can be completed by dragging items on the surface. The task was successfully completed by all participants. Many appraised the changing font size of items

during the dragging. However, we observed two difficulties with this task. First, for some subjects, it took quite a long time to find the necessary item. Second, some subjects had problems to seize the item when it was overlapped by other items.

Task 7. Delete an item. All subjects completed this task successfully. Four subjects deleted the item by dragging it onto the recycle bin and the rest did it through the item's context menu.

Task 8. Add a new item in the model. All subjects completed this task without serious problems since they had already (during the completion of previous tasks) noticed the *Add Interest* item in the surface's context menu that can be opened by right click on the surface. Many subjects appraised that the new interests are added in the place from which the context menu was opened.

Eye-Tracking

During the task-based experiment, the participants' gaze was recorded using eye-tracking equipment. The results of eye-tracking show the series of gaze fixations the participants made while looking at the interface and the density of fixations. Figure 6.10 shows a plot of the series of gaze fixation one participant has made during the first ten seconds with the interface. The analysis of gaze fixations for all the participants shows, that users make the first gazes at the labels in the red area of the surface. This is a good result since this area contains the items the user is strongly interested in, which is the most important information in the interface. The next gazes were made either at the side windows, e.g., *Navigator*, or the toolbar. Also, many participants spent a considerable amount of time on reading the type labels identifying the circular sectors.

The cumulate gaze density in the first twenty seconds (Figure 6.11) shows that during this period of time the participants looked more at the control elements (toolbar and side windows) than at the labels. This, however, can be caused by the novelty of the interface and the lack of knowledge about its purpose and function. Having looked at the labels on the surface, the participants could not fully understand their meaning and tried to find it using the control elements. Based on this finding, we can conclude that the meaning of labels and shades should be made more visible and self-explanatory, so that the user could quickly notice and comprehend it.

Interviews

After the task-based experiment, the subjects were asked to speak out on the positive and negative characteristics of IntrospectiveViews as well as on their suggestions to further improve the interface.

Figure 6.10.: Series of gaze fixations of one participant in the first 10 seconds [65]

Positive Characteristics. The participants spoke out on the interface as novel, innovative, cool, and easy-to-use. Many said that it would be a very useful and helpful tool in social networks. With respect to the hot-and-cold color metaphor, half of the participants said that it is an appropriate and comprehendible use of colors. The changing font size of labels on dragging and the highlighting of circular sectors on mouse over were also rated as positive. The interaction concept for changing the interest degree by dragging the label on the surface was rated as intuitive and easy-to-use. The visualization of semantic relations among items was rated especially positive, useful, and novel.

Negative Characteristics. One of the most often criticized problems was the overlapping of items, which makes it difficult to read and manipulate them. Often, in order to read the names of overlapped labels, the participants moved them in different directions, which caused unintended change of interest degree. Also, some participants said that the interface has too many control elements (especially on the toolbar) and this may confuse the user for the first time. However, another important critique was the poor visibility of some control elements: Some participants discovered the context menus of items and the surface by chance after having

Figure 6.11.: Gaze density of all participants in the first 20 seconds [65]

used the interface for a while. In the beginning, it was not clear for them that the items and the surface can be manipulated through the context menu.

Suggested Improvements. During the interview, the participants made a number of interesting suggestions to further improve the interface. One of them was a suggestion to provide a short tutorial explaining the important functions of the interface. The explanation can be provided in the form of callouts displayed next to the corresponding controls. This tutorial could be automatically displayed upon the first time with the interface and invoked later upon request. Another suggestion was to change the zoom function to show different levels of details at different zoom levels, so that in a zoomed out view only the most interesting items would be shown, but by zooming in the interface would be incrementally showing more of the less interesting items. Apart from that, it was suggested to allow the user to define own types and organize the items according to the self-defined types. Finally, a number of participants said that the interface should allow changing the color scheme of the circular surface.

Questionnaire

To evaluate the attractiveness and joy-of-use of IntrospectiveViews, we used the AttrakDiff^{TM} tool[2], which includes an online questionnaire and a number of diagrams visualizing the collected feedback. The AttrakDiff^{TM} questionnaire addresses the subjective attractiveness of a product as a composite characteristic influenced by four qualities:

- **Pragmatic quality (PQ):** the inherent usability of a product that indicates how successful the users can achieve their goals with the product.
- **Hedonic quality - identity (HQ-I):** the ability to develop the identity and help the user to establish personal connection with the product.
- **Hedonic quality - stimulation (HQ-S):** the ability to stimulate the need for further use.
- **Attractiveness (ATT):** the general outward appearance of a product.

The questionnaire consists of 28 word-pairs (Figure 6.12) organized according to the four qualities. For each pair, the subject had to cast a vote on a seven-value likert scale.

human	○	○	○	○	○	○	○	technical
isolating	○	○	○	○	○	○	○	connective
pleasant	○	○	○	○	○	○	○	unpleasant
inventive	○	○	○	○	○	○	○	conventional
simple	○	○	○	○	○	○	○	complicated

Figure 6.12.: Example of word-pairs in the AttrakDiff^{TM} questionnaire

5 out of 10 participants completed the questionnaire online. Figure 6.13 shows an overview of the received feedback with respect to the pragmatic (x-axis) and hedonic (y-axis) qualities. As it can be seen from the average value V, the interface was deemed to have a good hedonic quality and an average pragmatic quality. With respect to the hedonic quality, the confidence rectangle shows that the rating is relatively consistent, whereas for the pragmatic quality, it shows a significant deviation. The possible reasons for the significant disagreement on the pragmatic quality might be the small number of respondents and the prototypical state of the interface.

Figure 6.14 shows the mean values for each four qualities influencing the overall attractiveness of IntrospectiveViews. As it can be seen from the figure, the rating

[2] `http://www.attrakdiff.de/`

Figure 6.13.: Average rating with respect to the hedonic and pragmatic qualities [65]

of HQ-S is way above average. It shows that the interface has a good ability to develop the stimuli for further use, which fulfills Requirement Q4: Foster curiosity and engage users. HQ-I received a somewhat lower rating than HQ-S. However, it is important to notice, that the identity is strongly affected by the product's brand and the user's personal memories and associations with it. These variables are out of scope of this study. The ratings of the pragmatic quality and attractiveness indicate that the both qualities still have room for improvement.

Figure 6.15 provides a detailed view on the ratings of each of the four qualities by showing the mean values for the all 28 word pairs. As it can be seen from the diagram, all values are in the positive range. The most interesting information here is the extreme values, which show the most problematic characteristics as well as the characteristics that have been especially well resolved. Considering the extreme positive values, it can be concluded that IntrospectiveViews is a very practical, inventive, innovative, and novel interface. Considering the extreme low values, we see the following problems: First, in the pragmatic quality dimension, two word pairs received relatively low ratings. These are "cumbersome - straightforward" and "confusing - clearly structured". This is partially caused by such problems as the relatively large number of control elements displayed within the interface, over-

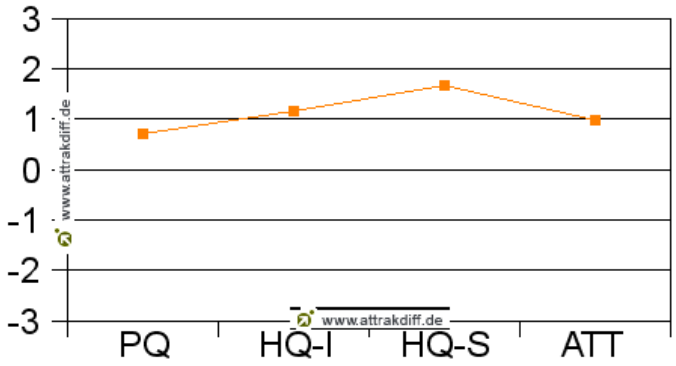


Figure 6.14.: Mean values of the ratings with respect to the four qualities [65]

lapping labels, and poor visibility of important control elements (e.g., controls for adding new interests and obtaining detailed information about a specific interest). Another cause of the low ratings in this dimension is the prototypical state of the interface. At the time of evaluation, some functions had not been implemented or were malfunctioning.

The rather average rating of the attractiveness is partially caused by the difference in the opinion regarding the color scheme. Half of the participants liked the hot-and-cold metaphor, but the other half found it unpleasant and even dangerous. Another reason is the relatively large number of control elements being displayed, the small size of the toolbar buttons, and the Java Swing look-and-feel of the interface elements. In the interview, some participants spoke out on the Java look rather negatively.

6.3.5. IntrospectiveViews v.3

The outcomes of the evaluation of usability and hedonic qualities show that users deem IntrospectiveViews as a useful, novel, and motivating tool. They also reveal a number of usability problems and provide insights for solving these problems and improving the interface. Some of these problems are fixed in the third version of the interface. This version also implements some of the suggestions collected in the two previous studies. Among others, the third version supports functions for domain modeling, which were requested by many subjects in both studies. In particular, it allows creating user own semantic types, connecting items, and adding new instances into the domain model.

This section presents the third version of IntrospectiveViews (Figure 6.16). It describes the improvements made in this version with respect to the problems identified in the previous studies. It also presents the functions for editing the domain

Figure 6.15.: Mean values of the word pairs [65]

knowledge model. We describe the new version on the example of a user interest model for the biochemical literature portal presented in Section 5.2.2. This model represents such categories of interests as organisms, genes, enzymes, substrates, etc.

Similar to the previous two versions, the third version of IntrospectiveViews was also evaluated in a user study. However, unlike its predecessors, the third version was evaluated in a context of a real personalized portal. Results of this study are presented in the Validation chapter (Section 8.1).

Comprehending Main Purpose

From the observations we made during the completion of *Task 1. Acquaint yourself with the interface and guess what purpose it has* (see Task-Based Experiment in Section 6.3.4), we found that users cannot understand the main function and purpose of the interface in isolation by just using it. Since the concept of scrutable user models is new to an average web user, the interfaces representing user models

Figure 6.16.: IntrospectiveViews v.3

should convey a clear statement of their purpose and function. This can be achieved by showing users the connection between the user model and the personalization effects generated based on its content. This connection is provided by personalizable portlets of the personalization unit (Section 7.4). These portlets display IntrospectiveViews next to the personalized content and immediately project all the changes in the user model on the personalized content. In addition to that, we introduced several features that help users to comprehend the purpose of IntrospectiveViews and learn how to use it.

The meaning of information in IntrospectiveViews is displayed by the legend located on the status bar (Figure 6.17). This legend replicates the color spectrum of the circular background and displays labels for the interesting and uninteresting zones of the spectrum. It also shows an indicator (yellow bar) referring the current interest degree, which is based on the radial distance of the mouse pointer to the circular sector. In addition, the interface displays a hint telling users what actions they can perform at the current part of the model or on the current item. Hints are displayed on the status bar and as a tooltip next to the mouse pointer. It displays hints telling how to add items, change interest degree, view full information of an item, and filter items by type (hide/show circular sectors). These improvements allow users to understand the semantics of information displayed even at the first

time of use. They also help users to learn how they can manipulate this information without reading manuals or looking up in the help system.

Figure 6.17.: Statusbar in IntrospectiveViews v.3

Finding Items by Name

During the task-based experiment (Tasks 3-7) and the interview we found that the most difficult and time-consuming task while performing actions on a specific item was to locate the item on the surface. This problem becomes especially serious in large models consisting of several hundred items. To solve this problem we introduced a function for searching items by name. The interface displays an auto-completion search field beneath the navigator window (Figure 6.18). Upon a user typing the first several characters of the item's name in the search box, the interface displays a list of matching items, highlights them on the surface, and dims all other items. Using this function, users can easily and quickly locate the items they are looking for.

Figure 6.18.: Search items by name in IntrospectiveViews v.3

Excessive Number of Controls

During the interviewing, many subjects spoke out that their interaction was complicated by the relatively high number of control elements displayed in the interface, namely, many buttons on the toolbar and the dialog windows for filtering displayed on the right. To reduce the number of elements, we redesigned certain functions of the interface. We removed the *Types* window that was used for filtering items

by type. In the new version, the sectors representing semantic types can be minimized/expanded by clicking the sector. Additionally, users can filter items by semantic types using the *Class* menu (Figure 6.19). This menu can be opened by clicking the *Class* label on the status bar. In this menu, users can select and deselect checkboxes to show and hide items of corresponding types. Also, we replaced the *Interest Groups* window by the legend for interest zones on the status bar. In addition, we removed the toolbar and moved some of its functions to the context menu. These changes helped us to reduce the clutter and achieve a more simplistic design of the interface.

Figure 6.19.: Filtering by class in IntrospectiveViews v.3

Perception of Color Metaphor

We learnt from the interviews that the color metaphor of the interest zones in IntrospectiveViews is perceived differently by different users. Half of the participants found the hot-and-cold metaphor appropriate for the color scheme of the circular surface, but the other half found it less suitable. Some said it is too technical and unattractive. Attractiveness, however, is an important requirement for the interface since it influences the motivation to use and the ability to engage users and to foster curiosity. To improve the attractiveness of the color scheme, we introduced a function for changing colors of the circular background. By clicking the interest zones legend on the status bar, the interface will display a menu for choosing color schemes (Figure 6.20). A click on a color label will paint the circular surface according to the color scheme (Figure 6.21). Also, it will immediately project the color

change on the personalized content in the portal (see Section 7.4 for more details). We believe that this function can influence the user's engagement and motivation to use interface.

Figure 6.20.: Menu for setting color schemes in IntrospectiveViews v.3

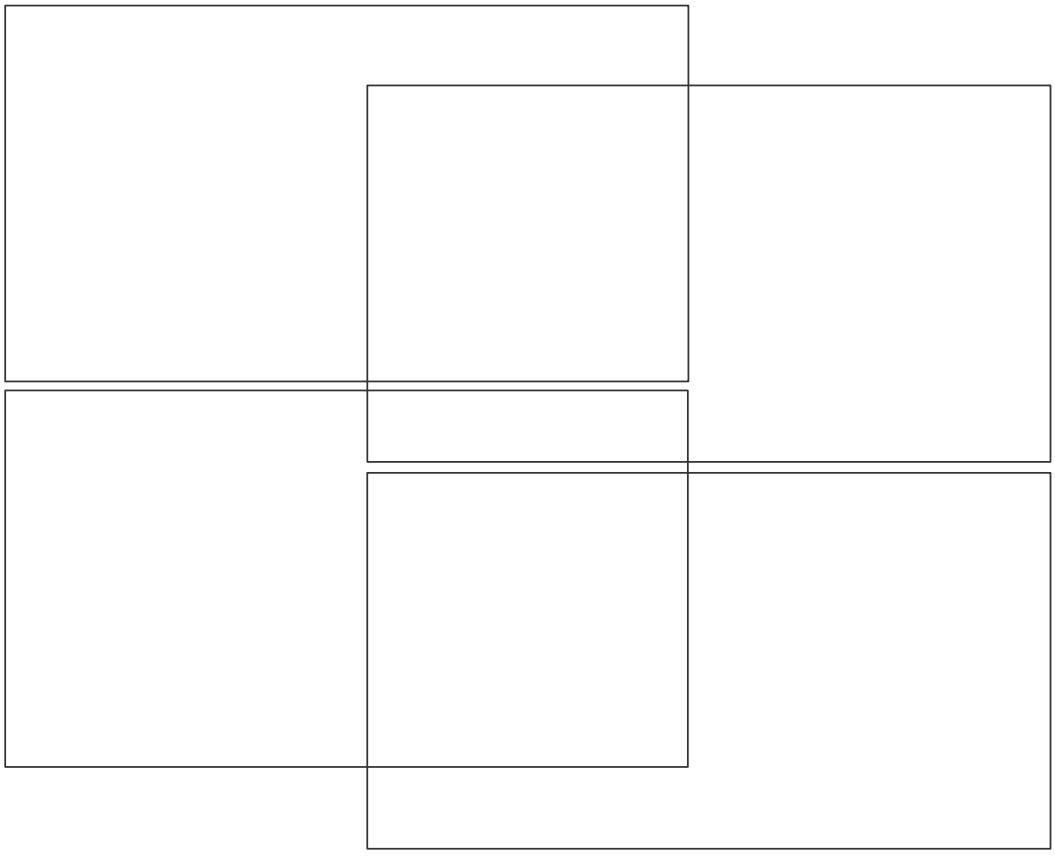


Figure 6.21.: Color schemes in IntrospectiveViews v.3

Domain Knowledge Modeling

From both studies of IntrospectiveViews, we learnt that users want to be able to define their own types/categories of interests and organize items in their models according to these categories. Also, some subjects spoke out that they would like to be able to define their own relations among items in the model. The user-defined types, categorization of items, and relations contain valuable information for adaptive systems. This information can be used to update and extend the domain knowledge model of the system. To leverage the willingness of users to semantically organize items in their user models, we introduced functions for domain knowledge modeling in IntrospectiveViews v.3. These functions allow adding new instances into the domain model, creating new classes, classifying instances, and creating semantic relations among instances.

New classes can be added as types of user interests through the *Class* menu (Figure 6.22). In the string field located on the top of the menu, user types the class name and hits Enter key. The interface invokes the method for adding ontology class of the domain modeling service. Note: user contributions to the domain model are stored in the user's personal version of the model (see Section 5.3.1 for more details). Once the class is added to the domain model, the interface creates a respective sector on the circular surface as it is displayed in in Figure 6.22 (*MyClass* sector). User-created classes can be deleted from the context menu of corresponding sectors.

Figure 6.22.: Adding classes and instances in IntrospectiveViews v.3

New instances can be added as user interests through the *Add new item* menu (Figure 6.22). This menu can be opened by double clicking on an empty place within the circular surface. The menu contains an auto-completion field displaying

a list of matching instances that already exist in the domain model. If the user types a name that does not match to any of the existing instances, the interface invokes the method for adding new instance of the domain modeling service. The new instance will be created as an instance of the semantic class corresponding to the class of the sector in which user opened the menu for adding interest. Once, the instance is added to the user personal version of the domain model, the interface invokes the method for adding a user interest of the user modeling service and places the corresponding item on the circular surface. The interest of the new item is computed based on the coordinates where the user requested the menu for adding interest.

Relations between instances can be created as connections between user interests through the *Connect items* menu displayed in Figure 6.23. This menu is opened when the user drags one item onto another. In the menu, the user can select either one of the existing relation types or type a new one. Upon clicking the *Connect* button, the interface will invoke the method for creating instance relations of the domain modeling service. After the relation is created, the interface will display it along with other existing relations as arcs to the corresponding items (Figure 6.24).

Figure 6.23.: Connecting instances in IntrospectiveViews v.3

User-defined classes, instances, and instance relations contributed through IntrospectiveViews are stored in user individual versions of the domain knowledge model. Knowledge elements of user individual versions of the domain knowledge model can be also used for annotation of portal content by users. This allows further automatic tracking of user interest change on explicitly entered user interests since the metadata of portal pages accessed by users is used for automatic updates of the user model. Moreover, individual domain knowledge models of the entire user community can be analyzed to identify the most frequent classes, instances, and relations. This is valuable information since it represents the wisdom of the user community. It can be used to update and extent the shared version of the domain model. Due to the limited scope of this thesis, the aspect of aggregation of user contributions to the domain knowledge model is not further elaborated.

Figure 6.24.: User-defined relations in IntrospectiveViews v.3

However, we believe that this aspect has a good potential for research and should be investigated in the future work.

6.3.6. Conclusions

Our iterative approach to the development and evaluation of IntrospectiveViews helped us to reveal a number of important aspects of scrutinizing semantic user models. We found what functions are important for such models and how they must be realized. We learnt what information should be displayed to users and in what form. This approach also helped us to come to a novel and efficient design for graphical user interfaces for scrutable user modeling. On top of that, we learnt about the attitude of users to scrutable user models and the factors that influence their motivation to use such models.

Results of the usefulness and usability evaluation of IntrospectiveViews v.1 (Section 6.3.2) unveil important functions for scrutinizing semantic user models. They show that displaying content of user models groped by type or category helps users to perceive and understand it better. Such grouping also helps users to navigate through and search in user models. The results show that users deem very useful to have an option for viewing content relevant to items in their models, e.g., related documents. They like to see what content was used for updating the user model. All users deem very important to be able to override the system-generated updates of user model. Users also like to be able to view semantic relations among items. However, users do not want these relations to be used for propagating the changes they make explicitly. Another interesting observation we made is the willingness of users to semantically enrich their user models. Users are willing to add new

items, define their own types, categorize items by type, and create semantic relations among items. For some users, the support for functions for organizing items in the user model determines the acceptance of the model.

The evaluation results show that users deem IntrospectiveViews as a novel, motivating, and easy-to-use realization of functions for scrutinizing user interest models. Users like the metaphor of displaying interests on circular zones partitioned into slices, where each zone represents items of certain interest degree and each slice represents items of a specific type. Users like the interaction pattern for changing interest degree, where they can set a higher interest by dragging items to the center and a lower interest by dragging them to the edge of the circle. Another interesting observation is that the subjective attractiveness of IntrospectiveViews is influenced by the ability of users to customize the interface, e.g., the ability to choose a color scheme they like for the circular zones.

We believe that these findings provide useful insights for those who design scrutable user models. They show important information and features these models should support. Also, our findings show that IntrospectiveViews provides a comprehensible metaphor for visualizing user models as well as it implements efficient and intuitive interaction patterns for manipulating them. Therefore, these visualization metaphors and interaction patterns can be reused in other adaptive systems for viewing and editing user models.

6.4. Summary

In this chapter, we have presented the user modeling unit of the framework for scrutable adaptivity in community-enabled web portals. The unit is responsible for storing and managing a user model providing semantic description of user features. The unit consists of the user model itself, a user modeling service implementing the logic of update and access operations on the model, and a graphical user interface for scrutable user modeling IntrospectiveViews.

The user model is designed as an overlay model. It defines user features as a weighted overlay of concepts from the domain knowledge model, represented as an OWL-ontology. In the proof-of-concept implementation, we focus on user interests. For each concept, the user model stores the interest status and interest weight denoting the degree to what the user is interested in it. It also stores evidence of interest, i.e., the source from which the system computed or inferred the interest.

All operations on the user model are performed by the user modeling service. The service implements three types of methods: query, modify, and modeling methods. Query methods are invoked to retrieve content of the user model. Modify methods are used to alter information in the model, e.g., user changes made through IntrospectiveViews. Modeling methods implement the logic of automatic updates

of the model. The service implements methods for two types of automatic updates, namely, the methods for computing interest degree based on the user browsing history and the methods for propagating interest degree from one item to another using the semantic relations between items defined in the domain knowledge model.

IntrospectiveViews is a graphical user interface that provides users an access to the user model. It visualizes user models using a metaphor of circular zones partitioned into slices, where each zone represents items of certain interest degree and each slice represents items of a specific type. It provides functions for getting overview, zooming, filtering, navigation, and search. It also displays relevant content and semantic relations among items. In addition to viewing, IntrospectiveViews allows editing information in the model. It allows adding and deleting items, changing interest degree, organizing items by type, defining user own types, and creating semantic relations among items.

Also in this chapter, we have reported on two studies of IntrospectiveViews: the evaluation of usefulness and usability and the evaluation of usability and hedonic qualities. Findings of these studies unveil important functions for scrutable user modeling and show what information of user models and in what form must be displayed to users. They also show that users deem IntrospectiveViews as a comprehensible and clear visualization of semantic user models and that it provides an intuitive and easy-to-use realization of functions for manipulating such models.

The next chapter presents the personalization unit of the framework. It demonstrates how the user model is used for personalizing portal content. It also explains how IntrospectiveViews is accessed in the portal and used for controlling personalization in individual portlets.

CHAPTER 7

Personalization Unit

This chapter presents the personalization unit, the last part of the framework for scrutable adaptivity in community-enabled web portals. This unit is responsible for tailoring portal content to needs and preferences of individual users. The unit consists of three main components: personalization rules, a personalization service, and a Java class for personalizable portlets. *Personalization rules* implement the logic of transformation procedures that tailor portal content to user interests and preferences. The *personalization service* acts as a mediator between portal pages and personalization rules. More specifically, it applies personalization rules to portal content. Finally, the *PersonalizablePortlet* class extends the standard functionality of portlets by implementing methods for delivering personalized content and allowing users to scrutinize and control the personalization behavior.

Parts of research presented this chapter were published in our earlier papers. Our work on personalization modeling is published in [14]. The research related to the portal that we use as a scenario for user-controlled personalization is published in [21, 22]. Also, a patent application [20] has been made for the concept and interaction patterns of user-controlled personalization in portlets.

7.1. Personalization Rules

Personalization rules govern how the portal content is adapted and presented to users given their interest profiles. They are defined as a sequence of transformation procedures that applied to the original content to transmute it into a personalized state. Each procedure denotes a certain personalization effect requested by the user for a given portlet. The framework allows setting personalization effects at the level of individual portlets, which fulfills Requirement F11: Portlet-specific personalization rules. A portlet may support more than one personalization ef-

fect. For instance, news stories in a news portlet can be sorted according to the relevance to the user's interest profile and additionally interesting fragments can be foregrounded. The selection of personalization effects supported by a portlet depends on the content type of the portlet. The proof-of-concept implementation of the framework supports two content types for portlets: document set portlet and item set portlet. This distinction is based on the annotation level of the portlet content. In a document set, a document is annotated with one or multiple semantic concepts, whereas, in an item set, each item is annotated with only one semantic concept.

A document set portlet displays a number of documents satisfying a certain criterion or a set of criteria. Examples of such portlets include, but not limited to, a portlet displaying news stories of a given portal category, e.g., business or politics (see example of the news aggregating portal in Section 5.2.1) or a portlet displaying a list of most recent scientific papers matching a certain user query (see example of the portal for biochemical literature in Section 5.2.2). For the proof-of-concept implementation, we developed three personalization procedures for document set portlets: sort documents by interest, highlight interesting documents, and highlight interesting items. The first personalization procedure orders documents from the most interesting to the less interesting. The second procedure highlights the most interesting documents by a color marker. The third procedure foregrounds the most interesting keywords in the text of documents.

An item set portlet displays a number of items matching certain criteria. For example, in the portal for biochemical literature, the Index portlet displays a set of named entities extracted from a single publication or from a set of publications. Another example of item set portlets is the Map portlet in the news aggregating portal. This portlet provides a map displaying geographical locations extracted from a single news story or from stories of a certain portal category. The framework supports two personalization procedures for portlets displaying a set of items: sort items by interest and highlight interesting items. The first procedure orders the items from the most interesting to the less interesting. The second procedure colors the items that match user interests according to the color scheme selected by the user (see selection of color scheme in Section 6.3.5).

The logic of the five above mentioned personalization rules is implemented in the Java programming language. More specifically, for each of the two portlet content types, the framework contains an interface. The *DocumentSetPersonalizationRule* interface implements the personalization rule for portlets displaying a set of documents. The *ItemSetPersonalizationRule* implements the personalization rule for portlets displaying a set of items.

7.1.1. Document Set Personalization Rule

Listing 7.1 displays a pseudo code of the method implementing the logic of personalizing a document set. The method takes three objects as input parameters: a JSON array containing original content of documents that have to be displayed along with a set of named entities extracted from the content, user identifier, and portlet identifier. At the initial step, the method calls the user modeling service (Section 6.2) to retrieve a JSON array of interests of the given user. It also retrieves a JSON object of portlet personalization preferences. This object contains information about the personalization effects that the user requested for the given portlet. This information is stored in the *preferences* part of the user model (Section 6.1).

```
FUNCTION personalizeDocumentSet(docs, user, portlet)
  userInterests = UMS.getInterests(user)
  portPersPrefs = UMS.getPortletPersonalizationPreferences(user, portlet)

  // compute similarity of documents to user interests
  IF portPersPrefs.sortByInterest OR portPersPrefs.highlightInterestingDocs
    FOR EACH doc IN docs
      docSim.put(doc.id, computeSimilarity(doc.tags, userInterests))

  // sort documents
  IF portPersPrefs.sortByInterest
    docs = sortBySimilarity(docs, docSim)

  // highlight interesting documents
  IF portPersPrefs.highlightInterestingDocs
    colorScheme = UMS.getColorScheme(user)
    FOR EACH doc IN docs
      relSim = docSim.get(doc.id).sim * 100 / docSim.getMax()
      IF relSim > 50
        color = colorScheme.getColor(relSim)
        doc.highlightColor = color

  // highlight interesting items
  IF portPersPrefs.highlightInterestingItems
    IF colorScheme IS NULL
      colorScheme = UMS.getColorScheme(user)
    FOR EACH doc IN docs
      FOR EACH tag IN doc.tags
        IF userInterests.get(tag).interestStatus = interesting
          color = colorScheme.getColor(userInterests.get(tag).interestWeight)
          doc.content.highlightFragment(tag.start, tag.end, color)

  RETURN docs
```

Listing 7.1: Personalization rule for a document set

If the user has requested the personalization effect for sorting documents by interest or the effect for foregrounding the most interesting documents, the method computes a document similarity map. For each document, it computes a coefficient denoting the similarity of the semantic tags of the document to the user's interest profile. For the computation of the document similarity, we leverage the Block Distance function provided by Similarity Metric, an open source library for similarity algorithms developed at Sheffield University[1].

After the document similarity is computed, the method checks the user's preferences on each of the three personalization effects available for a document set. First, it checks whether the user requested the effect for sorting documents by interest. If true, the method sorts the document array based on the document similarity map. Second, the method checks whether the user requested the effect for highlighting interesting documents. If true, the method calls the user modeling service to retrieve the color scheme selected by the user. Then it iterates through the document set and for every document computes a relative document similarity. The relative similarity is computed based on the document absolute similarity and the highest absolute similarity in the entire set of documents. The documents with the relative similarity above 50% are highlighted. Third, the method checks whether the user requested the effect for highlighting interesting items in the text of documents. If true, it iterates through document tags of each document in the document array. For every tag, the method checks whether the user profile contains a matching item. If the method finds a matching item with the status *interesting*, it highlights all occurrences of this tag in the document text using the color that was computed based on the interest degree of the matching item. After all three personalization effects are checked, the method returns the transformed documents as a JSON array.

7.1.2. Item Set Personalization Rule

The logic of the method for personalizing an item set is displayed in Listing 7.2. Same as the method for personalizing a document set, this method takes three objects as input parameters: a JSON array containing original content of items that have to be displayed, user identifier, and portlet identifier. Initially, the method calls the user modeling service to retrieve a JSON array of interests of the given user and retrieves a JSON object of portlet personalization preferences. If the user has requested any of the two supported personalization effects, either the effect for sorting items by interest or the effect for highlighting items by interest degree, the method updates the set of items with the interest weight. For each item, it checks whether the user profile contains a matching interest item. If a match is found,

[1] `http://sourceforge.net/projects/simmetrics/`

the method assigns its interest weight to the item. If no match found, the interest weight is set to 0.

```
FUNCTION personalizeItemSet(items, user, portlet)
  userInterests = UMS.getInterests(user)
  portPersPrefs = UMS.getPortletPersonalizationPreferences(user, portlet)

  // compute interest degree for items
  IF portPersPrefs.sortByInterest OR portPersPrefs.highlightByInterest
    FOR EACH item IN items
      IF item IS IN userInterests
        item.interestWeight = userInterest.get(item).interestWeight
      ELSE
        item.interestWeight = 0

  // sort items
  IF portPersPrefs.sortByInterest
    items = sortByInterest(items)

  // highlight interesting items
  IF portPersPrefs.highlightByInterest
    colorScheme = UMS.getColorScheme(user)
    FOR EACH item IN items
      color = colorScheme.getColor(item.interestWeight)
      item.highlightColor = color

  RETURN items
```

Listing 7.2: Personalization rule for an item set

After the items are updated with interest weights, the method checks the user's preferences on each of the two personalization effects available for an item set. First, it checks whether the user requested the effect for sorting items by interest. If true, the method sorts the item set using the interest weights assigned to items. Second, the method checks whether the user requested the effect for highlighting items by interest degree. If true, the method calls the user modeling service to retrieve the color scheme selected by the user. Then it iterates through the item set and for every item computes a color based on the assigned interest degree. At the end, the method returns the transformed items as a JSON array.

7.2. Personalization Service

As described in detail in Section 7.3, the portal content is delivered to users through personalizable portlets. Users can view portlets in standard or personalized states. If a portlet is requested in personalized state, the portlet calls the personalization service to get the personalized content. The process of personalizing portlet content is illustrated in Figure 7.1. We describe this process using an example of the

publications portlet (see Portal for Biochemical Literature in Section 5.2.2). This portlet displays a list of scientific publications matching a certain user query. The logic of the method for personalizing the publications portlet is displayed in a pseudo code in Listing 7.3.

```
FUNCTION getPersonalizedPublications(user, portlet, query)
  // get content
  pubs = RMS.getPublications(query)

  // get metadata
  FOR EACH pub IN pubs
    pub.tags = RMS.getMetadata(pub)

  // personalize
  pubs = DocumentSetPersonalizationRule.personalizeDocumentSet(pubs, user, portlet)

  RETURN pubs
```

Listing 7.3: Personalization of a publication list

As the first step, the personalization service (PS) obtains the content that has to be displayed in the publications portlet, namely a set of scientific papers matching the query that the user has entered in the portlet. For this purpose, it invokes the *getPublications* method of the resource management service (RMS). This method returns the matching publications as a JSON array. Then, PS iterates through the array and for every publication it retrieves the publication metadata using the *getMetadata* method of RMS. The metadata represents a set of instances from the domain ontology (Section 5.3.1) describing the meaning of content. The method updates each publication record with the retrieved metadata. Afterwards, the method applies the *DocumentSetPersonalizationRule* to the publications. As described in Section 7.1, this rule transforms the set of publications according to the personalization effects requested by the user for the publications portlet. Finally, the method returns the personalized set of publications to the publications portlet.

Implementation

Same as the other services of the framework, the personalization service is implemented as a Java Application. For access from JavaScript- or Applet-based GUIs, e.g., the IntrospectiveViews interface, the framework provides a **restful web service for personalization**. The web service is implemented as a Java Web Application.

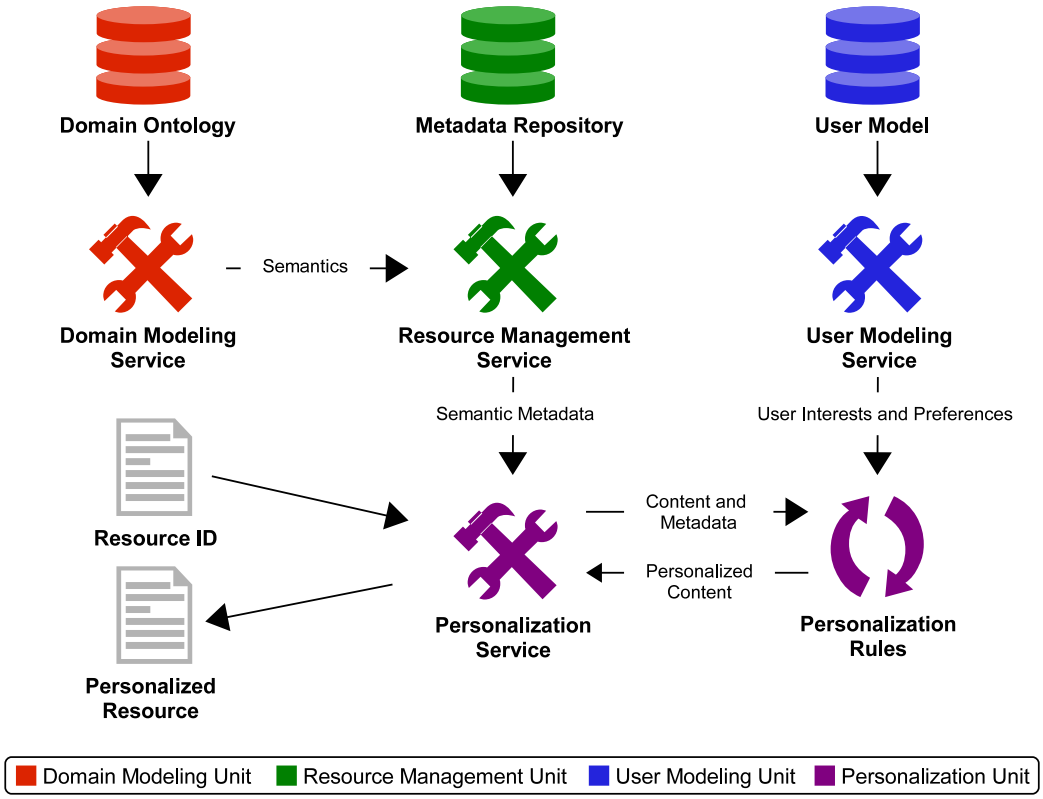


Figure 7.1.: Personalization workflow

7.3. Personalizable Portlet Class

PersonalizablePortlet class is intended for portlet application developers who need programming portlets that allow end users to control personalization. This class is developed as an extension of the *GenericPortlet* class of the Portlet Specification JSR 286 [104], which provides a default Java language implementation of portlet data model, lifecycle, and functionality. More specifically, the *GenericPortlet* class implements methods for portlet instantiation, initialization, request handling, rendering, etc.

In the *PersonalizablePortlet* class, we added a number of additional methods that enable user-controlled personalization. The class implements a method allowing users to activate and deactivate personalization in the portlet (fulfills Requirement F13: Turning personalization on and off). It provides functions for fine-tuning of personalization effects at the level of individual portlets (fulfills Requirement F14: User access to personalization rules and Requirement F11: Portlet-specific person-

alization rules). Also, it implements methods allowing users to access and edit their user models and see the direct impacts of a model change on the personalization (fulfills Requirement F15: Explain implications of altering models).

In the *PersonalizablePortlet* class, we overrode the *doView* method that it inherits from the *GenericPortlet* class. In *GenericPortlet*, the *doView* method is used for rendering portlet content in view mode. The overridden version of this method in the *PersonalizablePortlet* class renders the portlet content in either personalized or standard views depending of the user's preferences. Listing 7.4 displays a flow of instructions executed in the overridden *doView* method.

```
FUNCTION doView(renderRequest, renderResponse)
  // get user and portlet identifiers
  user = renderRequest.getRemoteUser()
  portlet = getClass().getCanonicalName()

  // get portlet personalization preferences
  portPersPrefs = UMS.getPortletPersonalizationPreferences(user, portlet)

  // if portlet is rendered first time for the user
  IF portPersPrefs IS NULL
    portPersPrefs = getDefaultPortletPersonalizationPreferences()
    UMS.setPortletPersonalizationPreferences(user, portlet, portPersPrefs)

  // personalization turned off
  IF portPersPrefs.isPersonalizationOn IS FALSE
    // show menu for activating personalization
    addPersonalizationMenuOff()
    // render portlet in standard view
    doStandardView(renderRequest, renderResponse)

  // personalization turned on
  ELSE
    // show menu for tuning or deactivating personalization
    addPersonalizationMenuOn()
    // render portlet in personalized view
    doPersonalizedView(renderRequest, renderResponse)
```

Listing 7.4: *doView* method in *PersonalizablePortlet* class

Initially, the *doView* method sets the identifiers for the user and portlet. The portlet identifier is the canonical name of the underlying class as defined by the Java Language Specification, e.g., "myapp.portlets.MyPorlet". Then the method calls the user modeling service (UMS) to retrieve personalization preferences for the given user and portlet from the preferences part of the user model (Section 6.1). If the portlet is displayed first time for the user, the user model does not contain any personalization preference for this portlet. In this case, the method initializes the preferences object with the default preferences set by the portlet developer in the portlet deployment descriptor portlet.xml (see Portlet Specification JSR 286 [104]).

Also, the method calls UMS to write the default personalization preferences in the user model.

After the personalization preferences object is retrieved from the user model or initialized with default preferences, the method reads the *isPersonalizationOn* variable of this object to determine whether the portlet must be rendered in personalized view or standard view. If the variable is set to *false*, i.e., personalization is deactivated, the method adds to the portlet markup a drop-down personalization menu. This menu provides a command that allows the user to turn the portlet personalization on. Having generated the menu, the method passes the render request and render response to the *doStandardView* method. This method renders the portlet content in impersonalized state. The logic for generating a standard markup must be implemented by overriding the *doStandardView* method in concrete portlets inheriting the *PersonalizablePortlet* class.

In case the *isPersonalizationOn* variable of the preferences object is set to *true*, i.e., personalization is activated in the portlet, the *doView* method adds a drop-down personalization menu containing two commands: a command for opening the portlet personalization preferences and IntrospectiveViews visualizing user interests (Figure 7.2) and a command for turning personalization off. After the personalization menu is added, the *doView* method passes the render request and render response to the *doPersonalizedView* method, which renders the portlet content in personalized view. The logic for generating a personalized markup must be implemented by overriding *doPersonalizedView* in concrete portlets inheriting the *PersonalizablePortlet* class.

Also, for the *PersonalizablePortlet* class, we implemented a number of JavaScript functions to enable an asynchronous communication between the personalization preferences window (Figure 7.2) and the services for user modeling and personalization. Changes made by users on the portlet personalization preferences or interest profile trigger execution of JavaScript functions. These functions invoke corresponding methods in the *PersonalizablePortlet* class to modify the user model and update the personalized content. Due to this asynchronous communication, a user change on personalization preferences or interest profile is immediately projected on personalized markups of the affected portlets without reloading the entire portal page (fulfills Requirement F15: Explain implications of altering models). The next section describes in more detail the user interaction with personalizable portlets in a portal for biochemical literature.

7.4. User-Controlled Personalization

The framework for scrutable adaptivity proposed in this thesis was integrated into two portals, a news aggregating portal (Section 5.2.1) and a portal for biochemi-

cal literature (Section 5.2.2). Both portals are deployed on IBM WebSphere Portal Server[2]. Content of these portals is delivered to users through personalizable portlets implemented using the *PersonalizablePortlet* class, which we described in the previous section. These portlets allow users to turn personalization on and off and fine-tune personalization effects. In this section, we describe the user-controlled personalization using an example of the portal for biochemical literature.

The main page that users see after they have logged into the portal (Figure 5.5) consists of a number of portlets providing different types of content and functions. The *Query* portlet displays a list of user search queries, which are used by the portal to retrieve publications from scientific databases. Upon a mouse click on a query, the portal will display a list of matching publications in the *Listing* portlet. In the *Index* portlet, users can display results of various semantic assistants, e.g., mycoMINE assistant [93], which extracts entities and facts related to fungal enzymes, such as enzymes, organisms, assays, genes, substrates and pH, temperature or activity assay conditions. Mentions of these entities can also be highlighted in the text of publications in the *Listing* portlet.

Both the *Listing* and *Index* portlets are implemented using the *PersonalizablePortlet* class, which means that they provide users full control over the personalization. Users can turn personalization on and off using a drop-down menu, which can be opened by clicking the heart icon on the portlet title bar. By selecting the *Turn personalization off* command, the portlet will be turned into standard view, i.e., no personalization will be applied on the content of the portlet. By selecting the *Turn personalization on*, the portlet will be turned back into personalized view.

In the personalized view, users can view and edit their interest profiles as well as define how the portlet content should be personalized. This can be done in a personalization options window, which can be opened by selecting the *Personalization options & interest profile* command from the portlet personalization menu. The personalization options window is displayed as an overlay on top of the portlet. Figure 7.2 displays a personalization options window for the *Listing* portlet. In the upper part, the window shows the options for personalization supported by the portlet. Content of the *Listing* portlet is personalized according to the *DocumentSetPersonalizationRule* (Section 7.1.1). It supports three personalization effects: (1) publications can be sorted with respect to the relevance to user interests; (2) the most interesting publications can be highlighted by a color marker; (3) mentions of items from the user interest profile can be highlighted in the publications list.

The *Index* portlet adapts content using the *ItemSetPersonalizationRule* (Section 7.1.2). It supports two personalization effects: (1) named entities can be sorted according to the user's interest profile; (2) entities can be highlighted by

[2] `http://www.ibm.com/software/websphere/portal/`

Figure 7.2.: Portlet personalization preferences and user interest profile displayed as an overlay window. A screencast demonstrating user-controlled personalization in portlets is available at `http://www.minerva-portals.de/research/introspective-views/v.3`.

interest using the color scheme selected by the user in the IntrospectiveViews interface. By selecting corresponding checkboxes in the personalization options window users can achieve personalization effects they like. User changes on the personalization options are immediately projected on the portlet content. For instance, having checked the option for sorting in the *Publications* portlet, the papers in the portlet will be immediately ordered from the most interesting to the less interesting. Similarly, having checked the option for sorting items in the *Index* portlet, the named entities in the portlet will be reordered by interest degree.

Figure 7.3 displays two states of the *Index* portlet: the states before and after selecting the option for sorting. As seen from the figure, this change affects the sorting of entities within categories and the sorting of entity categories. The sorting of entities within a category is based on the interest weights of matching items from the user interest profile. For example, in the Enzyme category, five entities have a match from the interest profile: BGL, cellobiase, FPC, endoglucanase, and Bgl. These enzymes are color coded according to the hot-and-cold color scheme and

sorted from the most interesting to less interesting. The sorting of entity categories is based on the cumulative interest weights of all entities within the category.

In addition to portlet personalization options, the overlay window shows the user's interest profile displayed through the IntrospectiveViews interface. As described in Section 6.3, the interface visualizes user interests using a metaphor of circular zones partitioned into slices, where each zone represents items of certain interest degree and each slice represents items of a specific type. The hot zone in the center displays items that users are strongly interested in. The cold zone at the circle edge displays items that users are not interested in. Items are grouped into circular sectors by type. The profile shown in Figure 7.2 displays items of such types as enzyme, gene, organism, strain, and some others.

In addition to viewing, IntrospectiveViews allows editing information in the model. It allows adding and deleting items, changing interest degree, organizing items by type, defining user own types, and creating semantic relations among items. To change interest degree of an item, the user needs to drag the item to the corresponding interest zone. Dragging to the center increases interest and dragging to the edge decreases interest. New items can be added into the profile by double clicking on an empty place of the circular surface. Items can be blocked from personalization by dragging them onto the recycle bin.

Similarly to changes on personalization options, all changes in the interest profile made through IntrospectiveViews are immediately projected on the personalized content. However, unlike personalization options which affect content of only one portlet, user model updates affect content of all portlets in personalized state displayed on the current page. For example, upon a change of interest degree of an item in the interest profile, the publications in the *Listing* portlet will be resorted and the color markers of most relevant publications will be updated. Also, if the *Index* portlet contains an entity corresponding to the item which interest degree the user has modified, the color coding of this entity will be updated as well.

7.5. Summary

In this chapter, we have presented the personalization unit, the last part of the framework for scrutable adaptivity for web portals. This unit provides rules and mechanisms for tailoring portal content to needs and preferences of individual users. It also provides graphical user interfaces for delivering personalized content and allowing users to scrutinize and control the personalization behavior. The unit consists of three main components, namely, personalization rules, a personalization service, and a *PersonalizablePortlet* class.

Personalization rules implement the logic of transformation procedures that tailor portal content to user interests and preferences. These rules are defined is such

Figure 7.3.: Personalized list of named entities before (left) and after (right) selecting the option for sorting

a way that the execution flow of transformation procedures can be dynamically adapted based on the user's personalization preferences. These procedures can be applied at the level of individual portlets, which fulfills Requirement F11: Portlet-specific personalization rules. This gives portlet application developers and portal administrators a high level of flexibility in personalizing portal content.

The personalization service acts as a mediator between portal pages and personalization rules. More specifically, it applies personalization rules to portal content. The service can be invoked from any portlet or portal page. Having been invoked, it fetches the content that has to be personalized, retrieves the content metadata providing machine-processable semantics of the content, and passes them to the corresponding personalization rule. After the personalization rule has been applied, the service returns the personalized content to the requestor. Additionally, the framework provides a restful web service for personalization. The restful web service enables personalization of portal content in an asynchronous manner. It can be used for personalizing individual fragments of a portal page without reloading the entire page.

The *PersonalizablePortlet* class is designed for portlet application developers who need programming portlets that provide users full control over personalization. This class implements a method that embeds a GUI element showing users what personalization effects have been made in the portlet. In this element, users can also adjust personalization effects to their preferences. The availability of this GUI element in portlets fulfills two functional requirements for scrutable adaptivity, namely, Requirement F12: Explanation of personalization and Requirement F14: User access to personalization rules. Moreover, the class implements a method allowing users to activate and deactivate personalization in portlets, which fulfills Requirement F13: Turning personalization on and off. Finally, personalizable portlets place an entry point to the user's interests profile next to the personalized content (fulfills Requirement Q1: Findability of scrutinizing tools), so that, in case of incorrect personalization behavior caused by a wrong assumption about users' interests, users can easily access their profile and override the system's belief. A change in the interest profile is immediately projected on the personalized content. The availability of this function in the system fulfills Requirement F15: Explain implications of altering models.

The interaction methods and interfaces provided by the *PersonalizablePortlet* class help users understand the connection between their user models and the end personalization effects. In its turn, this should make the adaptive behavior more transparent and comprehensible for users. This should also help users develop a metal model of affordability to control the adaptive behavior. Section 8.1 of the next chapter shows whether these goals are achieved by our approach. In that section, we report on a user study in which we evaluate the impacts of controllable

personalization on the usability, usefulness, user satisfaction, transparency, and trustworthiness of adaptive systems.

CHAPTER 8

Validation

This chapter presents a validation of our approach. In Section 8.1, we report on a user study of scrutable adaptivity. This study assesses the impacts of our approach on the usefulness, usability, user satisfaction, transparency, and trustworthiness of personalized portals. In Section 8.2, we examine the compliance of our framework with fifteen functional requirements imposed on the user model, domain knowledge, metadata, and personalization rules and effects. In that section, we also analyze the compliance with four qualitative requirements for scrutable adaptivity. Finally in Section 8.3, we provide a summary of the validation.

8.1. User Study of Scrutable Adaptivity

This section reports on a user study that we conducted to evaluate the impacts of our approach to scrutable adaptivity on the user subjective perception of a personalized portal. The main goal of this study was to test our hypothesis that our approach can improve the use acceptable of personalized systems. Also, we aimed to collect user feedback with respect to possible improvements and enhancements of our approach.

8.1.1. Research Hypothesis

The main research hypothesis of this study was: our approach to providing users full control over adaptivity can improve the user acceptance of personalized systems. More specifically, we believed that our approach can improve the usefulness and usability of personalized systems, increase the user satisfaction, and make the personalized systems more transparent and trustworthy.

8.1.2. Study Design

To test this hypothesis, we conducted a user study using a personalized portal that provides users full control over adaptivity. The study was conducted at the Concordia Centre for Structural and Functional Genomics (CSFG)[1]. The subjects of this study were seven employees of the center working in the Genozymes project, where the goal is to find novel ways of creating bioproducts and biofuels from green waste. Part of their work is the curation of characterized glycoside hydrolases[2] of fungal origin from the domain literature. The main background of the subjects is biology, chemistry, and genomics.

For this user study, we used the personalized portal for biochemical literature presented in Section 5.2.2. The portal provides a personalized single-point of access to abstracts of publications harvested from multiple scientific databases and supports a further analysis of these abstracts. More specifically, it allows users to process the harvested content with a number of semantic assistants.

The study was conducted in two phases. In the first phase, the subjects were provided with a version of the portal in which the control over personalization was blocked. The subjects knew that the portal collects information about their interests and personalizes its content. More precisely, publications in the portal were automatically sorted according to interests of individual users. Also, the text fragments of publications matching to items in the user model were highlighted by color. However, the subjects did not have access to the user model. They were also not able to switch personalization off or change personalization effects. The subjects were using this version of the portal for the entire duration of the first phase, which lasted two weeks.

At the end of the first phase, we asked the subjects to fill out a questionnaire. The questionnaire was designed based on the USE question set of Arnold Lund [88]. It includes questions regarding to the usefulness, ease of use, ease of learning, and satisfaction. Additionally, we added questions on the novelty and trust. In total, the questionnaire consists of 36 questions grouped into five categories. Figure 8.1 displays the complete lists of questions. For each question, the subjects were asked to cast a vote on a six-point Likert scale. The questionnaire was available online in the portal. The subjects were able to fill it out on their own computer at any convenient time. We collected responses from all seven subjects of this study. After all subjects had completed the questionnaire, we started the second phase.

In the second phase, the subjects were provided with a version of the portal that gives users full control over personalization as it is described in Section 7.4. In this version, they were able to access the user model, edit it and see the effects on personalization. They were also able to switch personalization on and off and fine

[1] `http://genomics.concordia.ca/`

[2] family of enzymes used to break down plant cell walls

Portal - Usefulness			**1**	**2**	**3**	**4**	**5**	**6**	
A1	It helps me be more effective.	strongly disagree	○	○	○	○	○	○	strongly agree
A2	It helps me be more productive.	strongly disagree	○	○	○	○	○	○	strongly agree
A3	It is useful.	strongly disagree	○	○	○	○	○	○	strongly agree
A4	It makes the things I want to accomplish easier to get done.	strongly disagree	○	○	○	○	○	○	strongly agree
A5	It saves me time when I use it.	strongly disagree	○	○	○	○	○	○	strongly agree
A6	It meets my needs.	strongly disagree	○	○	○	○	○	○	strongly agree
A7	It does everything I would expect it to do.	strongly disagree	○	○	○	○	○	○	strongly agree
Portal - Ease of Use			**1**	**2**	**3**	**4**	**5**	**6**	
B1	It is easy to use.	strongly disagree	○	○	○	○	○	○	strongly agree
B2	It is simple to use.	strongly disagree	○	○	○	○	○	○	strongly agree
B3	It is user friendly.	strongly disagree	○	○	○	○	○	○	strongly agree
Portal - Ease of Learning			**1**	**2**	**3**	**4**	**5**	**6**	
C1	I learned to use it quickly.	strongly disagree	○	○	○	○	○	○	strongly agree
C2	I easily remember how to use it.	strongly disagree	○	○	○	○	○	○	strongly agree
C3	It is easy to learn to use it.	strongly disagree	○	○	○	○	○	○	strongly agree
Portal - Satisfaction			**1**	**2**	**3**	**4**	**5**	**6**	
D1	I am satisfied with it.	strongly disagree	○	○	○	○	○	○	strongly agree
D2	I would recommend it to a friend.	strongly disagree	○	○	○	○	○	○	strongly agree
D3	It is fun to use.	strongly disagree	○	○	○	○	○	○	strongly agree
D4	It works the way I want it to work.	strongly disagree	○	○	○	○	○	○	strongly agree
D5	I feel I need to have it.	strongly disagree	○	○	○	○	○	○	strongly agree
D6	It is pleasant to use.	strongly disagree	○	○	○	○	○	○	strongly agree
D7	It is novel.	strongly disagree	○	○	○	○	○	○	strongly agree
D8	It is engaging.	strongly disagree	○	○	○	○	○	○	strongly agree
Portal - Trust			**1**	**2**	**3**	**4**	**5**	**6**	
E1	It is transparent.	strongly disagree	○	○	○	○	○	○	strongly agree
E2	I feel I have full control over it.	strongly disagree	○	○	○	○	○	○	strongly agree
E3	I feel I can trust it.	strongly disagree	○	○	○	○	○	○	strongly agree

Figure 8.1.: Evaluation questionnaire for controllable personalization

tune personalization effects for individual portlets. The subjects were using this version for a period of two weeks. Afterwards, they were asked again to respond to the same questionnaire that they filled out in the first phase. Please note that we were able to collect responses on the second questionnaire from five of the seven subjects that took part in this study.

Additionally, at the end of the second phase, we interviewed six subjects of this study. In the interview, we requested them to speak out on the positive and negative aspects of the portal. We also asked them about additional functions and content they would like to have in the portal. The rest of this section presents the results of the two surveys and the interviews.

8.1.3. Results and Discussion

A percentage distribution of responses on two questionnaires is shown in Figure 8.2. Please note that the results of Questionnaire I include responses from all seven subjects of this study, whereas the results of Questionnaire II include responses from five of the seven subjects. Figure 8.3 shows average values of responses with respect to individual questions of both questionnaires. Figure 8.4 displays average ratings with respect to the five categories of the questionnaires, namely, usefulness, ease of use, ease of learning, satisfaction, and trust.

As it can be seen from these figures, providing users control over personalization makes considerable impacts on the usefulness, usability, and user satisfaction of the personalized system. On average, the second version of the portal, in which users had control over personalization, received a 23% better rating than the version without the control. In the rest of this section, we discuss the results of the surveys and interviews with respect to the five categories of the questionnaire. We also present our findings with respect to the user modeling and user-driven changes on domain knowledge models of personalized systems.

Usefulness

The first version of the portal received relatively low rating in the *Usefulness* category. We attribute this to the prototypical state of the system and insufficient coverage of literature. In the interviews, most respondents said that they need the portal to be able to harvest publications from all the major scientific databases and to have at least two years of retrospective coverage. They also need a number of additional features, such as advanced search, functions for sharing publications, commenting, and rating.

However, the second version of the portal, which allows fine tuning of personalization, received a 24% better rating of the usefulness than the first version. Despite the prototypical state and low coverage of literature, the respondents rated the second version as a very useful tool that helps them to be more effective and productive. From the interviews, we found that users like very much to be able to view and edit their interest profiles. They also like the functions for applying different personalization effects to the portal content. Users especially like to be able to sort content according to the interest profile. The function for highlighting items from the user model in the text of portal pages was deemed useful as well.

Also, we received a number of interesting suggestions to further enhance the usefulness of controllable personalization in the portal. For example, some respondents said that they would like to have a function for setting alerts on certain parts of the user model. The system should send a notification to the user when new content relevant to the defined interests becomes available.

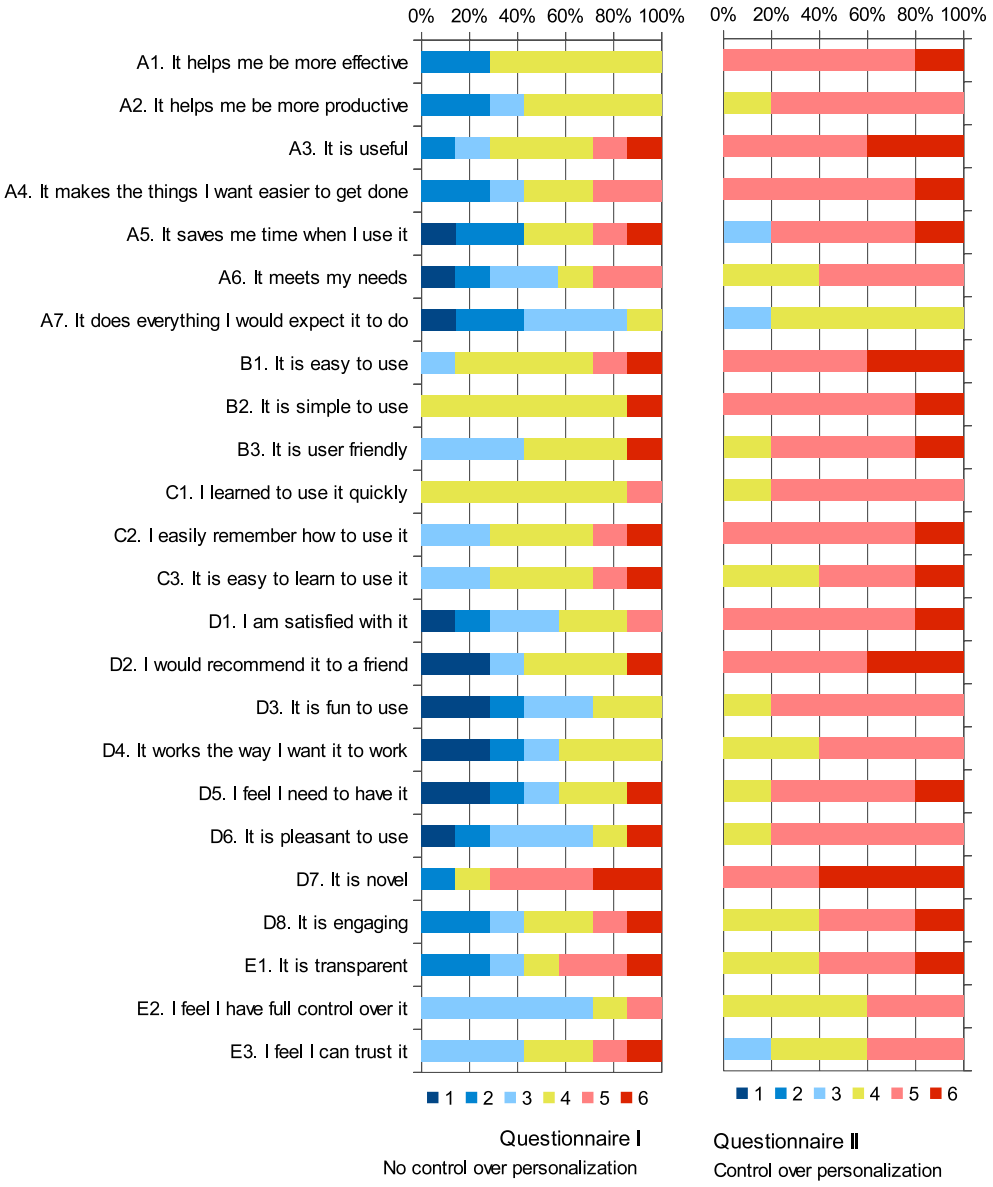


Figure 8.2.: Percentage distribution of questionnaire responses

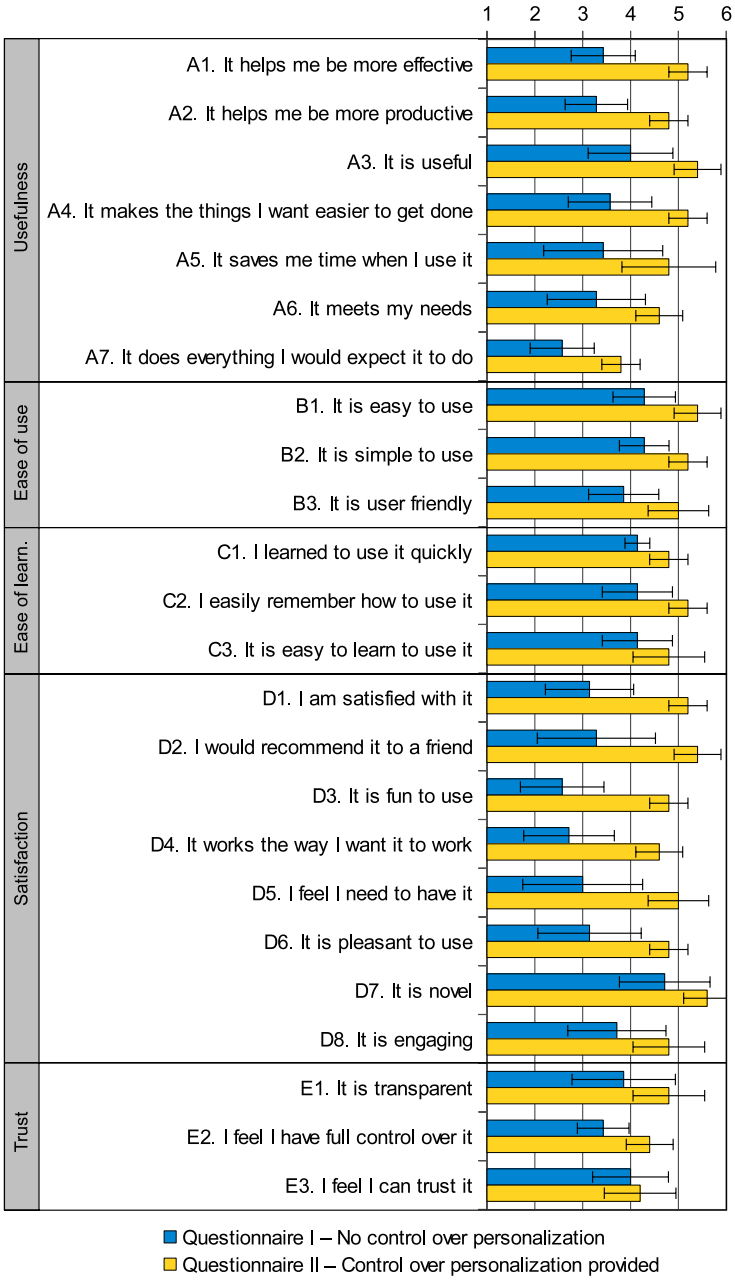


Figure 8.3.: Average values of questionnaire responses

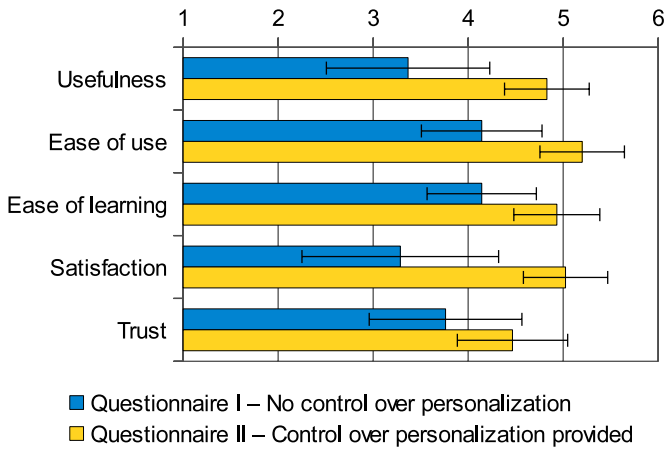


Figure 8.4.: Questionnaire responses by category

Ease of Use and Learning

The version with controllable personalization received better ratings of the ease of use (18% increase) and the ease of learning (13% increase). The better rating in these categories is a surprising observation since the second version had a higher complexity than the first version, i.e., a higher number of control elements and functions. We attribute the improvement of the perceived usability of the second version to the gain of its outward attractiveness. A very strong correlation between the perceived ease of use and the subjective attractiveness of computer systems was first detected by Kurosu and Kashimura [84] and then confirmed in a number of replicated experiments by Tractinsky [123]. These studies reveal that users deem a visually attractive system more usable than its analog with a less attractive appearance.

In the case of the second version of the portal, the improvement of attractiveness is likely to be caused by IntrospectiveViews, the interactive visualization of user interests. The results of our previous study of the usability and hedonic aspects of this visualization (Section 6.3.4) show that users deem the interface visually attractive, engaging, pleasant, and fun to use. Hence, we argue that integration of this visualization into a personalized system can improve not only its attractiveness, but also the perceived usability.

Satisfaction

In this category, we observed the highest increase of the rating in comparison with other four categories. The respondents were 29% more satisfied with the portal in

which they had control over personalization than with the portal where they did not have such control. All respondents were either satisfied or very satisfied with the second version.

We also observed a very strong impact of controllable personalization on the emotional perception of the portal by users. The results show that the second version is deemed to be 37% more fun-to-use, 28% more pleasant, and 18% more engaging. Also, the willingness to recommend the system to a friend is 35% higher and the user's need to have the system is 33% stronger.

These results show that our approach successfully fulfills two qualitative requirements for scrutable adaptivity, namely, Requirement Q4: Foster curiosity and engage users and Requirement Q3: Usability of scrutinizing tools. These requirements are important for scrutable adaptivity because the emotional aspects influence the users' motivation to use their interest profiles for adjusting personalization. Hence, users are more likely to keep the information in the profile up-to-date and accurate. The accurate and complete information in user profiles, in its turn, ensures the quality of personalization effects and the precision and recall of content recommendation.

Trust

We observed a 16% increase of the transparency. The user's feeling of having control over the system is increased too at the same rate as the transparency. During the interviewing, all respondents said that they like to be able to know whether the content is personalized or not and what adaptations have been made. They also appreciated the access to their interest profiles and the ability to choose what personalization effects have to be applied to the content.

However, the results show only a marginal 3% increase of the trustworthiness. From the interviews, we found that some users do not care whether the portal collects information about their interests as long as it uses it for recommending relevant content, whereas others have significant concerns about their privacy. The difference of user attitude to the collection and usage of personal information can also be seen from a relatively high deviation in responses on question E3. From these results we can conclude that different users have different concerns about privacy, hence they need different levels of control over their personal data. Those users who wish the highest degree of control should be able to delegate the management of their personal data to some trusted authority or to manage it locally on their desktops. They should be able to control what applications use their personal data and for what purpose.

User and Domain Modeling

The overall feedback with respect to user modeling is very positive. Users liked the function of using interest profiles for personalizing portal content. They liked that these profiles can be updated in both ways: unobtrusively based on the user interaction with the portal and explicitly by users. They also liked very much the visual interface for displaying and editing their profiles.

During the interviewing, the subjects made a number of interesting suggestions with respect to further enhancement of user modeling. Many subjects suggested allowing users to have multiple interest profiles. A user may work for several projects at one time. In different projects, users may be responsible for different tasks. Therefore, the portal should allow users to create multiple profiles and to be able to select a profile for personalization effects in a given portal session. Also, respondents suggested having a group profile that reflects interests of a project team or a group of friends. Similar to personal profiles, the portal should allow users to select group profiles for generating personalization effects. In addition, some users are willing to share their interest profiles with their colleagues or friends. This function can be especially useful in corporate portals for senior staff and experts willing to share their knowledge with less experienced staff members, e.g., new employees or interns.

Also, one of the factors that had a negative impact on the user satisfaction with the portal is the incorrect classification of certain entities in the user profile. Users found it irritating. However, some of them said that they would be willing to correct the occasional misclassification by rearranging items in their interest profiles, i.e., by moving entities to the corresponding sectors on the interface for user model visualization. These corrections can be used by the portal to update the ontology that represents machine-processable semantics of the domain knowledge. This information can be further used to improve the search and personalization in the portal.

8.1.4. Conclusions

The results of this study confirm our research hypothesis to a large extent. They show that our approach to scrutable adaptivity significantly improves most of the factors that influence the user acceptable of personalized systems. The function for fine-tuning of personalization effects using the interactive visualization of user profiles IntrospectiveViews helps users to find the information they need in an easy-to-use and efficient manner. In addition, the visualization makes significant impacts on the user's emotional perception of personalized systems. They deem such systems more fun-to-use and more pleasant. Also, it improves the transparency of personalized systems. However, our study did not confirm the claim that our

approach to scrutable adaptivity helps to establish a better trust between the user and personalized systems. In spite of giving users full access to the information that the system uses for personalization, they still do not have sufficient trust in such systems.

It is important to mention the limitations of this study. The reported user study was conducted with a relatively small group of users in laboratory settings using a prototype system. Also, it ran for a rather short period of time. In order to be able to draw more solid conclusions, this approach needs a long-term evaluation using a real system. It also should involve a larger number of users and leverage not only the users' self-reported metrics, but also the system's usage statistics.

8.2. Compliance with Requirements for Scrutable Adaptivity

In this section, we examine the compliance of the proposed framework to the requirements for scrutable adaptivity that we set in Chapter 3. More specifically, we discuss on the fulfillment of fifteen functional requirements, namely, requirements to user model, domain knowledge, resource metadata, and personalization rules and effects. Also, we review the compliance of the framework to four qualitative requirements for scrutable adaptivity.

8.2.1. Requirements to the User Model

User models are a key component for personalized systems. They represent information about individual users, such as their interests, expertise, or traits. In Section 3.1.1, we identified four functional requirements to user models in systems with scrutable adaptivity. These requirements refer to the representation of user features, methods for automatic elicitation of user features, accessibility of models by users, and types of information provided in user models.

Requirement F1: Semantic representation of user model

This requirement enforces that the user model is represented in a formal language capable of expressing rich semantic information about user features. The model must provide information about classes of modeled features, their properties, and semantic relations among features. Such representation must allow for reasoning. For example, it must enable propagation of user interest or knowledge from one item to another.

The framework wholly fulfills this requirement by representing user features, in the prototypical implementation user interests (Section 6.1), as an overlay of concepts defined in the domain knowledge model. The domain knowledge is formal-

ized as an OWL ontology (Section 5.3.1). It defines ontological classes and their instances, data properties, and relations. This semantics enables the framework to reason about user interests. The framework can infer new interests by propagating user interest from items that already exist in the model to new items using relations between instances and the class hierarchy defined in the domain ontology. Automatic propagation of user features compensates the scarcity of information in user models.

Requirement F2: User access to user model

To ensure transparency and predictability of a personalized system as well as to ensure users' right of access to their personal information, it is important that the user model is open to the user. Users need to be able not only to view information stored in the model, but also to modify it. The edit access to the user model is essential since it enables users to detect and correct wrong assumptions or beliefs the system could make about users.

This requirement is fully met in our approach. The proposed framework provides users full access to their user models through the IntrospectiveViews interface (Section 6.3). In this interface, users can view the entire set of assumptions the system makes about their interests. If necessary, users can override or delete these assumptions. Also, they can add new items into their models. Allowing users to access and edit the user model helps achieving that the information in the model is accurate and as complete as possible.

Requirement F3: Automatic update of user model

In large information portals with dynamically updated content, user interests, knowledge, and needs may evolve rapidly. In such a context, it is burdensome and time-consuming for users to keep the user model up-to-date. Therefore, personalized portals must provide mechanisms for automatic updates of user models using some evidence about users' interaction with the portal.

The proposed framework fulfills this requirement by providing a method for automatic updates of user interest profiles. This method is implemented by the user modeling service (Section 6.2). It can infer about user interests from two sources.

First, user interests can be elicited from the content accessed by users. More specifically, the method builds a log of named entities that users encounter on portal pages. Using the relative frequency of an entity in the user's log, the method computes a probability of the given user being interested in the entity.

Second, user interests can be inferred by reasoning over the semantic information defined in the domain model. More precisely, the method propagates user interest

from one item to another via the object properties inter-connecting these items in the domain ontology.

Both types of updates are performed by the framework fully automatically, in an unobtrusive manner, so that users can focus on their main tasks. They help to ensure that the information about user interests is up-to-date and as complete as possible.

Requirement F4: Explain sources of information in user model

The fourth requirement in this category refers to the degree of transparency in the user model. Personalized systems must allow users to access not only the resulting beliefs about user features, but also the information explaining the source of these beliefs. Upon a user request, the system must show all the sources that it used for inferring about a certain feature.

In our approach, the source of beliefs the system makes about users can be requested through the IntrospectiveViews interface (Section 6.3), which displays user interest profiles containing both system generated assumptions and items entered explicitly by users. To request an explanation about how a certain item made to the profile, users need to click this item in the visualization.

In the explanation window (Figure 6.7), the interface provides detailed information about the item. Among others, it shows pie charts representing the overall interest degree as well as the degree of interest from three sources. First, it shows whether the interest was computed based on the browsing history. If true, it displays a list of documents from which this interest was elicited. Second, it shows whether the interest was propagated from other items in the model. In this case, it displays a complete list of items that the interest was propagated from. Third, it shows whether the user set interest explicitly. If true, it displays the date and time of the corresponding update.

Providing this information to users ensures the transparency and predictability of personalized systems. This was confirmed by the user study of scrutable adaptivity, which we reported in the previous section.

8.2.2. Requirements to the Domain Knowledge

Domain knowledge models provide a vocabulary of concepts in a given domain. Such vocabularies are used in a broad range of adaptive systems to define user features in the user model and annotate content of the system. In Section 3.1.2, we identified three functional requirements with respect to the domain knowledge model in a portal supporting scrutable adaptivity. These requirements refer to the representation of domain knowledge model and methods for accessing and updating this model.

Requirement F5: Semantic representation of domain knowledge

According to this requirement, the domain knowledge model must be represented in a formal machine-processable language capable of expressing rich semantic information of the domain. It must provide information about the semantic class of entities and their properties. Also, it must represent semantic relations existing between entities.

As described in Section 5.3.1, the solution proposed in this thesis fully complies to this requirement by representing the domain knowledge model as an ontology formalized in the Web Ontology Language (OWL). The domain knowledge model consists of two parts: a terminological component (TBox) and an assertion component (ABox). TBox contains definition of ontology classes and their object and data properties, whereas ABox provides instantiation of the classes and properties defined in TBox. E.g., for the concept *Country*, ABox defines actual countries. Instances can be interconnected through object properties defined in TBox.

Providing information about the semantic class of entities allows disambiguation of entities of different classes having the same name. Also, it is used for helping users understand information in their user models, which are represented as an overlay of entities of the domain model. More specifically, this information is used for organizing and filtering entities in the visualization of user interests IntrospectiveViews (Section 6.3).

The information about semantic relations between entities in the domain model is used for two purposes. First, it is used for propagating user interest from items for which the model contains an evidence of interest to new items. Second, these relations are visualized in the IntrospectiveViews interface to help users to find related interests.

Finally, rich semantic information provided by the domain model is used for annotating portal content. As described in the next section (Requirement F8: Semantic representation of metadata), this information is leveraged by the adaptive system to solve problems of synonyms and polysemy and recommend related content to users.

Requirement F6: Automatic generation of domain knowledge

This requirement was set for the framework in order to address the challenges related to the dynamics of content change in community-enabled portals. The requirement enforces the framework to provide a method for automatic update of the domain knowledge. This method must be able to update the assertion part of the domain model based on semantic entities appearing in the portal content. Also, it must be able to infer semantic relations among items in the model.

The requirement is also met in the proposed solution. The domain modeling service (Section 5.3.2) of the proposed framework implements methods for adding instances and instance relations from the results of semantic annotation of portal content. More specifically, the service uses results of analysis of portal documents with tools for natural language processing to update the domain model. It converts entities and entity relations extracted from portal documents into instances and instance relations of the domain ontology.

Requirement F7: User community access to domain knowledge model

The last requirement imposed on the domain knowledge model refers to the accessibility of this model by the user community. Users must be enabled to contribute to the domain model of the portal. They must be allowed to create new entities and define relations between entities. In order to ensure that the model reflects different world views of users, users must be able to create own subsets of the model. The system must be able to use individual knowledge fragments to update the shared domain model.

As described in Section 5.3.1, the proposed architecture allows storing different versions of the domain model using the notion of *ConceptScheme* defined in the SKOS[3] standard (Simple Knowledge Organization System). All contributions of users are stored in the concept scheme of the corresponding user. In the proposed solution, users can contribute to the domain knowledge models in two ways. First, users can contribute to the model implicitly by providing semantic tags to portal resources through the *tagging interface* (Section 5.4.3) proposed in our earlier work [83, 82]. Also, users can indirectly update the domain ontology by adding and semantically connecting items in their user model using the *IntrospectiveViews* visualization (Section 6.3.5).

Due to the main focus of this thesis on adaptation and scrutability of adaptation, we treat the domain knowledge model as an auxiliary component. Therefore, in the prototype of the proposed framework, we implemented only the methods for automatic generation of domain model and methods for user-initiated updates of individual versions of the model. In this prototype, we did not implement the method for updates of the shared version of the domain knowledge model based on contributions of the user community. However, we believe that the analysis of user-created semantic annotations and user changes of semantics in the user model can provide valuable information for extending and enriching the shared domain knowledge model. Hence, this aspect should be further investigated in the future work.

[3] `http://www.w3.org/TR/skos-reference`

8.2.3. Requirements to Metadata

In order to achieve automatic selection of documents that match needs of individual users, an adaptive system need metadata conveying machine-processable meaning of these documents. In Section 3.1.3, we identified three functional requirements for metadata in portals supporting scrutable adaptivity. Similar to the requirements to domain knowledge, these requirements refer to the representation of metadata and methods for metadata access and update.

Requirement F8: Semantic representation of metadata

This requirement applies to the formalism for representing metadata. Like the content of user model, metadata must be defined in a language capable of conveying rich semantics of the domain. It must support disambiguation of entities having different meanings but the same spelling. Also, it must allow defining synonyms and different spellings of the same concept. Finally, it must allow retrieving related content.

This requirement is wholly fulfilled by the proposed solution. As described in Section 5.4.1, metadata in the framework is defined using the semantic vocabulary provided by the domain knowledge model, which is formalized in the Web Ontology Language (OWL). This language provides a high degree of expressiveness. It allows disambiguating same-spelled items that have different meanings. It allows definition of synonyms of an item by listing its all existing aliases. Also, it allows connecting items with arbitrary relations, which can be used by the framework for recommending related content.

Requirement F9: Automatic generation of metadata

In order to ensure that content is annotated in a timely manner, the framework must provide a method for automatic generation of metadata. The framework must be able to apply this method fully autonomously on every newly added or edited portal document without participation of the portal administrator.

This requirement is fully met by the proposed solution. The method for automatic generation of metadata is implemented by the resource management service as described in Section 5.4.2. For the annotation, the service leverages two external tools for Natural Language Processing, namely, OpenCalais[4] and the mycoMINE pipeline [93] deployed on the Semantic Assistants framework [132]. Both tools provide methods for named entity extraction. OpenCalais is used for named entity extraction in the news domain. Among others, it extracts such entities as company, technology, product, person, country, city, and many others. The mycoMINE

[4] `http://www.opencalais.com`

pipeline extracts entities and facts related to fungal enzymes involved in lignocellulose degradation, such as enzymes, organisms, genes, substrates, pH, temperature or activity assay conditions, etc.

The resource management service applies either of tools to every document at the moment of its assertion into the portal. The service converts the results of named entity extraction into the format of the domain ontology, generates an annotation record containing URIs of the matching instances, and writes it into the metadata repository. This ensures that even in portals with dynamically updated content every document is semantically annotated. This means that, from the moment of inclusion into the portal, documents can be used for search and personalization.

Requirement F10: User community access to metadata

Since automatically generated metadata does not always represent the entire meaning of the document or sometimes may even convey incorrect meaning, the framework must also allow the user community to contribute metadata. It must provide interfaces enabling users to add, edit, and semantically enhance metadata or portal documents.

This requirement is met in the framework by integrating the interface for semantic tagging of portal content (Section 5.4.3) that was developed in the context of the master thesis project of Alexander Kreiser [82, 83]. This interface can be displayed next to any document in the portal though a side portlet. The interface allows users to browse through the domain ontology and select tags for annotation. Users can also organize tags semantically by creating tag hierarchies and relations. Combining the automatic generated metadata with user-created tags can help to achieve a timely provisioning of high quality semantic metadata of portal content, which, in its turn, can improve the quality of search and personalization in the portal.

8.2.4. Requirements to Personalization Rules and Effects

Personalization rules define the logic of procedures that adapt content to needs of individual users. In Section 3.1.4, we identified functional requirements with respect to personalization rules. Also in that section, we set requirements that apply to the end personalization effects that are generated by the adaptive system based on the personalization rules.

Requirement F11: Portlet-specific personalization rules

Since personalization effects may vary from portlet to portlet, the framework must allow definition of personalization rules at the portlet level. It must allow portlet application developers to define personalization rules for each portlet individually.

This requirement is wholly fulfilled by the framework. As described in Section 7.1, personalization rules in the framework are defined as executable objects that can be applied to two types of portlet content: a document set and an item set. For each of these content types, personalization rules define a flow of transformation procedures that adapt the content. The selection of transformation procedures can be determined by the developer for each portlet individually using the *PersonalizablePortlet* class (Section 7.3). More specifically, in order to enable certain personalization effects in a given portlet, the developer needs to list the rules the portlet must support in the portlet deployment descriptor. Upon rendering the portlet in the portal, its content will be personalized according to the rules defined by the developer unless the user overrides them.

Requirement F12: Explanation of personalization

In order to make the adaptivity transparent for users, the framework must provide a comprehensible overview of personalization taking place on portal pages. Users need to know what fragments are adapted, what personalization effects are applied, and what user information is used for adaptation.

The proposed solution fully complies with this requirement. The *PersonalizablePortlet* class described in Section 7.3 provides methods for generating GUI elements that explain personalization. First, a personalizable portlet places a button for personalization menu to the title bar. This button contains a heart icon (Figure 7.2) showing the status of personalization in the portlet. A blue heart indicates that the personalization is turned off; whereas a red heart signals that the personalization is turned on.

By clicking the button, the portlet will open a window (Figure 7.2) providing more information about the portlet personalization. First, this window lists all personalization effects supported by the portlet. For each effect, it displays a checkbox showing whether the effect is applied to the content. Second, the window embeds the IntrospectiveViews visualization (Section 6.3) displaying the user's interest profile that has been used for personalizing the portlet content.

Results of the user study of scrutable adaptivity that we reported in Section 8.1 show that our approach to explaining personalization makes a significant impact on transparency of personalized systems. As seen from Figure 8.3, in a portal that provides explanation of personalization effects, we observed a 16% increase of subjective transparency in comparison with a portal that does not provide such explanation. Also, from the interviews, we learned that users like to be able to know whether the content is personalized or not. Also, they like to see a list of adaptations made and the interest profile that is used for personalizing the content.

Requirement F13: Turning personalization on and off

According to this requirement, the framework must provide a function for deactivation personalization. Users need to be able to turn personalization on and off. This function should be available for every portlet providing personalized content.

The proposed solution fully complies with this requirement. On the portlet title bar, personalizable portlets (Section 7.3) display a button for personalization menu. This menu allows users to switch the portlet from personalized view into standard view and vice versa. Availability of such function is an important requirement for ensuring the controllability of personalized systems. Users who do not need proactive behavior or perceive it as a disturbance can easily deactivate it.

Requirement F14: User access to personalization rules

Similarly to Requirement F13: Turning personalization on and off, this requirement also holds for the controllability of personalized systems. In addition to allowing users to view a list of personalization effects applied on the content, users need to be able to select or deselect effects according to their preferences. They need to be able to customize personalization effects at the level of individual portlets.

This requirement is wholly fulfilled by the proposed framework. As described in Section 7.4, personalizable portlets allow users to request a window displaying a list of personalization effects available for the portlet. These effects represent transformation procedures of the corresponding personalization rule (Section 7.1). In the list of personalization effects, users can select effects they want. Upon a change in the list, the content of the corresponding portlet will be updated according to the modified personalization rule. This function allows users to fine-tune adaptive behavior for each portlet individually, which gives them a high level flexibility of control over personalization in the portal.

Requirement F15: Explain implications of altering models

This requirement enforces an adaptive portal to explain users how a change in a personalization model, namely, in the domain knowledge model, metadata repository, user model, and personalization rules, affects adaptation in the portal. Users need to understand the connection between these models and the personalization.

In the proof-of-concept implementation of the framework, users can see direct impacts of changes in the user model and personalization rules. As described in Section 7.4, for each personalizable portlet, users can request a personalization menu (Figure 7.2). In this menu, they can modify the personalization rule of the portlet and edit their interest profile that is used for portlet personalization. A change on either the personalization rule or the profile is immediately projected on

the content, i.e., the content is updated according to the modified personalization rule and interest profile.

Regarding to the other two personalization models, the domain knowledge model and metadata repository, the proof-of-concept implementation of framework does not provide users an explicit explanation of changes in these models on personalization. However, users can indirectly see how these two models influence personalization. To illustrate this, let us consider an example of a user tagging a blog entry about the Titanic movie. The user assigns two tags to the content, Kate Winslet (person) and Titanic (movie), and creates a semantic connection between them to indicate that these tags are related. Before creating this connection, the interest profile of this user had an entry only for Kate Winslet, but not for Titanic. After this connection has been created, an entry Titanic is automatically added to the user's profile and user interest is propagated from Kate Winslet to Titanic. This will affect the personation, i.e., the user will be recommended not only content tagged with the Kate Winslet tag, but only content tagged with the Titanic tag. Also, search results for query Kate Winslet will be personalized by including hits related not only to Kate Winslet but also to Titanic.

8.2.5. Qualitative Requirements

In addition to functional requirements, we identified four qualitative requirements for scrutable adaptivity in community-enabled web portals (Section 3.2). These requirements apply to the usability of scrutinizing tools and their ability to motivate users to use them.

Requirement Q1: Findability of scrutinizing tools

Since scrutability of adaptation has not yet become an integral feature of all personalized systems, an average user does not have the mental model of affordance to scrutinize personalization. Therefore, the tools for scrutinizing personalization must be provided in a way that users can easily find them. The framework must place entry points to these tools in the most possible proximity to the personalized content.

This requirement is fulfilled in the framework by injecting an additional button to the title bar of personalizable portlets. A mouse click on this button opens a window for editing the portlet's personalization rules and the user's interest profile (Figure 7.2). Availability of this button on the portlet title bar indicates users that they have a function for customizing personalization in this portlet. Users can access this function directly from the personalized fragment.

Requirement Q2: Unobtrusiveness of scrutinizing tools

This requirement enforces that the framework provides tools for scrutability in an unobtrusive manner. User must be able to find these tools when they need. But these tools must not distract users from their main tasks.

In order to ensure compliance with this requirement, the GUI element for scrutinizing personalization is displayed only upon user request. To open this interface, users need to click the button for personalization located on the portlet title bar and select a corresponding command from the personalization menu. The tool is displayed as an overlay window on top of the portlet (Figure 7.2). Users can move the window on the page to obtain a suitable view on both the personalized content and the tool for controlling personalization. After users have adjusted personalization to their needs and preferences, they can close the tool and continue working with the personalized content.

Requirement Q3: Usability of scrutinizing tools

As users may not use tools for scrutinizing adaptivity often, they may not want to spend time on reading manuals or looking up into the help system in order to understand how these tools work. Therefore, it is important that these tools have good usability. Users need to be able to use these tools easily and efficiently without any preliminary training.

In order to fulfill this requirement, the key tool for scrutable adaptivity of this framework, the IntrospectiveViews interface, was developed iteratively following best practices of user-centric design. In the first iteration (Section 6.3.1), we implemented a large number of functions for various tasks of scrutable user modeling. We conducted a user study (Section 6.3.2) to evaluate the usefulness of these functions and the usability of their implementation. Based on results of the first evaluation, we redesigned the interface. In the second version of IntrospectiveViews (Section 6.3.3), we improved the functions that were rated as most important, removed the less important ones, and redesigned the functions which usability was rated low. The second iteration of IntrospectiveViews was again evaluated in a user study (Section 6.3.4). In the second study, we evaluated not only the inherent usability (pragmatic quality), but also the hedonic qualities and perceived appearance of the interface. The findings of the second study helped us to improve the perceived usability and appearance of the interface. The improved design was implemented in the third version of IntrospectiveViews (Section 6.3.5).

Finally, the third version of IntrospectiveViews was evaluated in a context of a personalized portal. As seen from results of this study (Section 8.1), a portal providing IntrospectiveViews as a tool for controlling personalization has a good rating of usability. Users deem it user friendly, easy and simple to use.

Requirement Q4: Foster curiosity and engage users

Since scrutinizing adaptivity is not the main task users want to accomplish in a personalized system, they may not have necessary motivation to use scrutinizing tools and contribute to the adaptation models, namely, to the user model, domain knowledge model, etc. However, user contributions to these models are important as they ensure their accuracy and completeness. Therefore, the tools for controlling personalization must stimulate users to contribute to these models by fostering curiosity, engaging users, and making the interaction with these models pleasurable and fun.

To comply with this requirement, we developed the IntrospectiveViews interface (Section 6.3) as a rich visualization that allows users to explore and modify their models in a highly interactive manner. The framework ensures that every user change done through IntrospectiveViews is immediately projected on the personalized content. This allows users to play with personalization using what-if scenarios. In its turn, this can increase the user's curiosity and motivation to update the user model and adjust personalization rules to their preferences.

Results of the user study of controllable personalization (Section 8.1) show that the scrutinizing tools provided by the framework have a good ability to engage and motivate users. As seen from Figure 8.3, a portal providing such scrutinizing tools received very good ratings of the user satisfaction. Subjects deemed the portal fun and pleasant to use, novel, and engaging.

8.3. Summary

In this chapter, we validated our approach to scrutable adaptivity in community-enabled web portals. First, we evaluated our approach in a user study using an application of a personalized portal for biochemical literature. In this study, we assessed the impacts of our approach on the user subjective perception of a personalized portal. Second, we examined the compliance of our approach with functional and qualitative requirements for scrutable adaptivity. For each of the nineteen requirements that we identified, we described the fulfillment of the requirement by our solution.

In the user study, we evaluated the impacts of our approach on the usefulness, usability, user satisfaction, transparency, and trustworthiness of a personalized portal. More specifically, we compared the user perception of two personalized portals: a portal without control over personalization and a portal where control over personalization is provided following our approach. Results of this study show that our approach makes significant impacts on most of these aspects. The portal with scrutable adaptivity received way better ratings on the usefulness, usability, user satisfaction, and transparency. From the respondents of this study, we learnt that

they like to be able to view and edit their interest profiles and customize personalization effects.

However, the user study results reveal that our approach makes only a marginal impact on the trustworthiness. In spite of full control over personalization, users' trust in the system is still relatively low. From the interviews with respondents, we found that some users do not care whether the portal collects information about their interests as long as it uses it for recommending relevant content, whereas others have significant concerns about their privacy. Therefore, we can conclude that different users might need different levels of control over their personal data. Those users who wish the highest degree of control should be able to delegate the management of their personal data to some trusted authority or to manage it locally on their desktops. They should be able to control what applications use their personal data and for what purpose.

In the second part of the validation, we analyzed the compliance of our framework with fifteen functional requirements imposed on the user model, domain knowledge, metadata, and personalization rules and effects. Also, we examined the compliance to four qualitative requirements for scrutable adaptivity. As seen from this analysis, the framework successfully fulfills all these requirements. For the representation of the user model and metadata, the framework uses the semantic vocabulary of the domain knowledge model. This vocabulary is defined in a formal language capable of expressing rich semantic information. The framework provides mechanisms for automatic updates of these three models as well as interfaces for editing these models by the user community. The semantic representation of these models, mechanisms for automatic updates, and interfaces for user-community contributions help to achieve a higher accuracy and completeness of information in the user model, domain knowledge model, and metadata repository. In its turn, this helps to achieve a higher quality of personalization and content recommendation and improve information search.

Also, the framework provides mechanisms giving users full control over the end personalization effects in the portal. Personalization rules in the framework are defined in a way that users can adjust them to their preferences and needs. These rules can be adjusted at the level of individual portlets. Changes on personalization rules or in the user model are immediately projected on the personalized content, so that users can easily see and comprehend the connection between the adaptation models and end personalization effects. Additionally, the framework provides a mechanism for deactivating personalization in individual portlets. The combination of these methods gives users a high flexibility in customizing personalized systems according to their needs and preferences.

Finally, the analysis of compliance to the qualitative requirements shows that the tools for scrutinizing adaptivity provided by the framework have good usability. Users are able to use these tools easily and efficiently without any preliminary

training or need to look up into the help system. Entry points to these tools are displayed at the closest proximity to the personalized content, so that users can find them quickly. The entry points and scrutinizing tools are displayed in a way that does not distract users from completing their primary tasks. Also, the analysis shows that the scrutinizing tools make impacts on the users' emotional perception. They have a good ability to motivate users to edit their user models. Indirectly, they can also motivate users to contribute to the domain knowledge model. This ability is important for a scrutable adaptive system since it ensures the accuracy and completeness of adaptation models.

In summary, our approach provides a successful solution for giving users full control over adaptivity in community-enabled web portals. It provides all necessary components, methods, and graphical user interfaces for such control. Also, it ensures user acceptance of this control. In the next chapter, we provide a summary of the whole thesis and make conclusions. Also, we outline the directions for possible extensions of this work and further research.

CHAPTER 9

Conclusions and Outlook

The main goal of this thesis was to develop a framework for scrutable adaptivity in community-enabled web portals. The aim of this framework is to make adaptive behavior of portals more understandable and give users full control over this behavior. The need for this framework was motivated by a number of challenges related to usability problems of adaptive systems. These problems embrace the facts that hindering users from accessing the user model and other personalization-related components of adaptive systems violates such principles as predictability, transparency, controllability, and unobtrusiveness [67, 70]. Also, it violates privacy legislation and negatively impacts the trustworthiness of adaptive systems [80, 117]. Finally, preventing users from controlling personalization may cause that the models that the system uses for adaptation are incomplete or contain obsolete or inaccurate information. This, in its turn, can lead to wrong adaptive behavior and irrelevant recommendations.

However, enabling scrutability of adaptive behavior and letting users control it is not trivial. Scrutability and controllability of adaptive behavior is complicated by a number of other challenges, namely, challenges related to the complexity of adaptation as well as a relative novelty of scrutability and its rather auxiliary function in adaptive systems [50]. More specifically, these challenges embrace the following facts. First, in case of systems applying complex adaptation rules, it might be difficult for an average user to understand the adaptation process and adjust it to personal needs. Second, since scrutability has not yet become an integral part of all adaptive systems, an average user might not have developed the mental model of affordability to control personalization. Third, as scrutinizing adaptation is not the main task that users want to accomplish while working with an adaptive system, they may not have enough motivation to do it. Forth, the presence of tools for

scrutinization may be perceived by some users as an obtrusion and may distract them from completing their primary tasks.

In the framework proposed in this thesis, we aimed to address all challenges that we mentioned above. This framework provides four models that an adaptive system needs for personalization: (1) a user model representing information about individual users, (2) a metadata repository providing annotations of content, (3) domain model defining machine-processable semantics of the domain knowledge used for modeling user features and annotating content, and (4) personalization rules defining the logic of adaptation. Also, the framework provides methods for keeping these models up-to-date, complete, and accurate. Among others, these methods leverage the willingness of the user community to contribute and annotate content. Furthermore, it provides graphical user interfaces and interaction patters for allowing users to view and adapt these models. Apart from that, our approach takes into consideration the distinctive features of web portals, systems that enable aggregation of content and processes by providing users a combination of portlets, where each portlet delivers fragments of content or functions of a certain application. More precisely, our approach provides methods for controlling personalization at the level of individual portlets, so that this control is tailored to the content and functions of each single portlet.

In the rest of this chapter, we revisit the main challenges that motivated this research and discuss how the approach presented in this thesis addresses them. Afterwards, we summarize the main contributions of our research and generalize about them. We discuss how our approach and findings of studies that we conducted contribute to theoretical foundations of the field of adaptive systems. Also, we report on the research limitations of this work and outline directions for possible enhancements of our approach and further research. Finally, we summarize this thesis and make concluding remarks.

9.1. Main Challenges Revisited

In the introduction chapter, we identified fourteen challenges that apply to scrutable adaptivity in community-enabled web portals. In this section, we revisit these challenges and discuss how they are addressed by our approach.

Challenge 1: Predictability and transparency. It was argued in the research literature [67, 70] that hiding from users the information that a system uses for adaptation hinders users in anticipating the system's behavior. It also occludes the system's status and makes its action incomprehensible for users. This challenge is addressed in our approach by opening all models that the system uses for personalization, namely, the user model, domain model, metadata, and personalization

rules. Through the IntrospectiveViews interface (Section 6.3), users are granted access to the information that the system collects about them. Users can view annotations assigned to portal content through the Tagging interface (Section 5.4.3). Also, both interfaces partially show users the semantic information of the domain knowledge: users can view semantic classes of entities and relations among them. Finally, users are allowed to view the logic of rules that the system uses for adaptation. User access to these models ensures that adaptive behavior of the system is fully transparent to users. It helps them to understand this behavior and foresee the system's actions.

Challenge 2: Controllability. It was also argued [67, 70] that blocking access to adaptation models hinders users from bringing the system in the state they want. We address this challenge by granting users a privilege of editing these models. Users are provided full access to their user models through IntrospectiveViews (Section 6.3). The interface enables users to change status of items in the model, add new items, and delete items. The Tagging interface (Section 5.4.3) enables edit operations on the metadata repository. It allows users to assign and modify annotations of documents. User contributions made through these interfaces are used to update the domain model. User can enter new items into the domain model, inter-connect items, and change the semantic class of items. Also, users are given full control over the end adaptation effects (Section 7.4). Users can adjust these effects to their personal needs and preferences. These privileges enable users to adjust proactive behavior of adaptive systems to their needs and preferences.

Challenge 3: Obtrusiveness of adaptive behavior. Adaptive system have been often criticized for their obtrusive way of providing personalization effects that distracts users from accomplishing their main tasks [70]. Different users perceive proactive behavior differently: some users find it helpful, whereas others perceive it rather obtrusive. To ensure unobtrusiveness of adaptive behavior, our framework enables users to turn personalization on and off (Section 7.4). Those users who feel distracted by the adaptive behavior have an option to completely deactivate it.

Challenge 4: Privacy. The majority of adaptive systems collect information about users for tailoring their content, navigation, or presentation. Hindering users from accessing this information is a serious violation of privacy legislation in many countries [80, 117]. It also causes users to feel insecure and decreases the system's trustworthiness. In our approach, we attempted to address this challenge by giving users full control over the user model and providing a comprehensive explanation of sources from which the system collects information about user features. Through the IntrospectiveViews interface (Section 6.3), users can view the entire set of be-

liefs the system makes about them. They can override these believes or delete them. Also, they can block any item in the model from being tracked by the system and used for personalization. However, results of our studies (Section 8.1) show that even though the system provides a full control over the user model and explains the sources of its beliefs, some users still have concerns about their privacy and do not trust such system. We, therefore, leave the question of ensuring user privacy open for further investigation.

Challenge 5: Breadth of experience. As Jameson argues in [70], by filtering out irrelevant or uninteresting (from the system's point of view) content, adaptive systems narrow the user experience. In an adaptive system, a user's knowledge about domains not captured by the user model is poorer than it would be in a non-adaptive system. This challenge is addressed by our approach in a number of ways. First, users have edit access to the user model (Section 6.3). They can explicitly 'tell' the system what information it must use for recommendation and what information it should filter out. Second, users have edit access to personalization rules (Section 7.4). They can set a rule for less aggressive personalization effects. For example, instead of filtering out irrelevant or uninteresting documents, they can select an option for sorting documents by interest or an option for foregrounding interesting documents. In this case, users are still provided with an entire set of documents. Third, users can deactivate personalization (Section 7.4), in which case, the system will display a complete set of documents as it would be in a non-adaptive system. Users can switch between impersonalized and personalized states in order to, on the one hand, have a full picture on all domains present in the system, and, on the other hand, take advantage of adaptive behavior to quickly and efficiently access relevant or interesting documents.

Challenge 6: Complete and accurate user model. Preventing users from accessing the user model may cause the model to be obsolete, incomplete, or contains wrong assumptions. This will result in a wrong adaptive behavior, e.g., issuing irrelevant recommendations or not recommending relevant content. We addressed this challenge by developing a hybrid approach to modeling user interests that combines system-generated updates with user explicit changes. The framework provides two methods for automatic identification of user interests (Section 6.2), namely, a method for eliciting user interests based on the analysis of user browsing history and a method for propagating user interest leveraging semantic relations defined in the domain ontology. Additionally, the framework gives users full access to the user model through the IntrospectiveViews interface (Section 6.3). It allows users to get an overview of all items in the model. Users can change the status of items, add new items, and delete outdated or wrong information. Our hybrid approach

to updating user models helps to achieve appropriate coverage of user features required for personalization. Also, it ensures that the user model is up-to-date and accurate.

Challenge 7: Complete and accurate domain knowledge. The domain knowledge model is an essential component for knowledge-based personalization systems. It defines machine-processable semantics of entities and classes that are used for modeling user features in the user model and for annotating system's content in the metadata repository. Therefore, inaccuracy or incompleteness in the domain knowledge may have serious impacts on the user modeling, information retrieval, and personalization. To address this challenge, we provide a method for automatic update of the domain knowledge model based on semantic analysis of portal content (Section 5.3.2). In addition, we designed the framework in a way that enables the user community to contribute to the domain model. First, the design of the domain model allows storing individual versions of domain knowledge created by users. Second, the framework provides users indirect access to the domain model through two graphical user interfaces, namely, the tagging interface and IntrospectiveViews. The tagging interface (Section 5.4.3) allows users to contribute to the domain model by semantically annotating portal documents, whereas the IntrospectiveViews interface (Section 6.3.5) enables users to indirectly populate and semantically enrich domain knowledge by entering new interests, creating semantic connections between interests, and defining semantic class of interests.

Challenge 8: Timely and accurate annotation of resources. Having machine-processable description of documents, i.e., metadata is an important requirement for a personalized system. It ensures that the system is able to select resources that match needs of individual users or provide relevant information. Not having such metadata or having it incomplete or inaccurate may result in the system recommending wrong content or not recommending relevant content. This challenge is addressed in our approach by combining system-generated annotations with annotations contributed by the user community. First, our framework provides a method for annotation of portal documents leveraging a stack of tools for Natural Language Processing (Section 5.4.2). This method is applied to newly created or newly edited documents. This ensures a timely description of every document in the portal, which, in its turn, ensures that every document can be used for personalization. Second, the framework provides a tagging interface (Section 5.4.3), through which the user community can annotate portal documents. This helps to achieve high-quality annotations that represent machine-processable semantics defined by human beings.

Challenge 9: User-tailored adaptation rules. Depending on the user personal preferences and tasks users need to accomplish while using a personalized system, they may want different personalization effects. For example in a news portal, some users may want to have news stories sorted according to their interests, whereas others may prefer a list of news stories sorted chronologically, where the most interesting stories are foregrounded. In our approach, we address this challenge by letting users customize personalization rules. The framework defines personalization rules in a way that allows adapting them dynamically according to the user's individual preferences (Section 7.1). Also, it provides a graphical user interface and an interaction method that enable users to view and easily modify personalization rules (Section 7.4). According to this interaction method, every user change on personalization rules takes effect immediately on the personalized content.

Challenge 10: Lack of awareness of scrutability. Since scrutability has not yet become an integral feature of all adaptive systems, users might not have developed the mental model of affordability to scrutinize and control adaptive behavior. Not every user might know that the adaptive behavior is determined by a user model and personalization rules and that he or she can edit these components and thereby control the adaptivity. Our solution to this problem is to place a visual cue of personalization to every adapted fragment. This cue serves not only as an indicator showing the fact of personalization, but also as an entry point to the scrutinization tools. In our approach, this cue is realized as a button with a heart icon placed on the portlet title bar (Section 7.3), which is already known by portal users as a container of control elements for operations on the portlet, e.g., for customizing, minimizing, closing, etc. The color of this icon shows the state of personalization in the portlet: a red icon indicates that the content is personalized and a blue icon signals that the personalization is turned off. Clicking this button opens a personalization menu (Section 7.4), through which users can turn personalization on and off as well as customize personalization rules and edit their user models. We believe that placing such cues in the direct proximity of personalized fragments can help users develop the mental model of affordability to scrutinize not only in web portals, but also in other web systems, e.g., in content management systems, social networks, wikis, blogging systems, and many others. However, in order to confirm our claim, a long-term evaluation of scrutable adaptivity is needed.

Challenge 11: Lack of motivation to scrutinize. As it was mentioned above, user contributions to the user model, domain knowledge, metadata, and personalization rules are important to ensure the accuracy and completeness of these models, which in its turn ensures the quality of end adaptation effects. However, contributing to these models is not the main task that users want to accomplish while working with

a personalized system; hence they might not have sufficient motivation to do it. To address this challenge, we designed the scrutinizing tools that are able to engage users, foster their curiosity to learn what the system knows about them, and explain how users can benefit from contributing to these models. The user modeling interface IntrospectiveViews (Section 6.3) allows users to explore and modify their models in a highly interactive and intuitive manner. This interface, when combined with an editor for personalization rules, can be used for fine-tuning personalization following a 'what-if' interaction pattern - every user change on the user model or personalization rules takes effect on the personalized content (Section 7.4). In this way, users see direct benefits of their contributions to the user model and personalization rules, i.e., better personalization effects, more interesting content, less irrelevant recommendations, etc. To motivate users to contribute metadata, the framework provides a tagging interface (Section 5.4.3) that enables users to annotate interesting or relevant content in order to be able to find and reuse this content later. User-created annotations are used by the framework to update the metadata repository, which is used for search and personalization. Finally, with respect to user contributions to the domain knowledge: Though the framework does not provide any interface that enables users to directly edit the domain model, it provides a method for updating this model and enriching its semantic using user contributions made on the user model and metadata. The content- and semantic-related contributions that users make through the tagging and IntrospectiveViews interfaces are stored in the personal version of the domain model of the corresponding user. A personal version contains such information as user-created tags or interests, semantic relations that users create between tags or interests, etc. Individual versions of the domain model can be statistically processed and aggregated in order to identify new knowledge elements (instances), classes of these elements (concepts), and relations between elements (object properties). This information can be used for updating the shared domain model. Both the user study of IntrospectiveViews (Section 6.3.2) and the study of scrutable adaptivity (Section 8.1) revealed the willingness of users to contribute domain semantics. Some subjects of these studies said that they would be willing to organize items of interest according to their own categories and create semantic relations between items that reflect their own associations. To summarize, we believe that the combination of graphical user interfaces and methods of our approach provides a solid basis for keeping content of adaptation models accurate and complete by leveraging contributions of the user community.

Challenge 12: Complexity of adaptation. Our approach, as a number of other approaches for adaptivity, leverages a semantically-rich user model (Section 6.1), which is designed as an overlay of knowledge elements defined in a domain ontology

(Section 5.3.1). This model has a high degree of complexity. It contains elements of different classes. Its elements may have a number of different properties and can be connected to other elements. The information about classes, properties, and relations influences the system's beliefs and personalization effects. For example, the system may use relations between knowledge elements for propagating user interest from one element to another. Due to the high complexity of this information, it might be difficult for users to understand and adapt it. This problem can get even worse in case of large models containing several hundred elements. In order to ensure that users can comprehend and adapt information in the user model, we designed the user modeling interface IntrospectiveViews (Section 6.3) following the information seeking mantra of Ben Shneiderman [118]. Shneiderman's principle for interaction with large and complex information sets can be summarized as: "overview first, zoom and filter, then details-on-demand". This principle is reflected in the interaction pattern of IntrospectiveViews as follows. The interface offers users an overview over all items present in the model. It allows zooming into different parts of the model to get a better view and allows filtering items according to different criteria to reduce the number of items displayed. Also, it provides detailed information of an item upon user request and allows revealing relationships among items. Finally, the interface allows users to add, edit, and delete items in their user models in an easy-to-use and efficient manner. The intuitiveness and usability of this interface was thoroughly evaluated in a number of user studies (Sections 6.3.2, 6.3.4, and 8.1). Results of these studies show that users deem the visualization intuitive, easy to understand, and user-friendly.

Challenge 13: Obtrusiveness of scrutability. Since scrutinizing adaptivity does not belong to users' main goals, the presence of tools for scrutinization might be perceived by some users as an obtrusion and distract them from completing their primary tasks. To ensure unobtrusiveness, the scrutinizing tool of our approach is displayed only upon user request. Users can request this tool by clicking the personalization button located on the title bar of personalized portlet (Section 7.4). The tool is displayed as an overlay window on top of the portlet. Users can move the window on the page to obtain a suitable view on both the personalized content and the tool for controlling personalization. After users have adjusted the personalization to their needs and preferences, they can close the tool and continue working with the personalized content. This approach ensures that users can access scrutinizing tools any time when they need them. But these tools do not distract users from completing their primary tasks.

Challenge 14: Portlet-level personalization. Since in this thesis we focus on adaptivity in web portals, we had to consider the distinctive features of portals.

More specifically, we needed to take into account the fact that content in portals is delivered through portlets and personalization effects may drastically vary from portlet to portlet. Also, users may want to have personalization in some portlets, but in others have it deactivated. To address this challenge, the framework enables personalization at the level of individual portlets. It provides an implementation of a Java class for personalizable portlets (Section 7.3). This class can be used by developers for programming portlets that support portlet-level personalization. More specifically, developers can define personalization options for each portlet individually. These options will be available for end users when the portlet is rendered in the portal, so that they can select options they like for each portlet independently. Also, personalizable portlets enable users to deactivate adaptivity completely in portlets where it is undesired.

9.2. Contributions and Theoretical Insights

By solving the above-mentioned problems we achieved the three main goals of our research that we set in the Introduction chapter (Section 1.2).

- **Goal 1:** we developed tools for scrutinizing personalization models required for adaptivity in web portals, namely, tools for scrutinizing the user model, domain knowledge, content metadata, and adaptation rules. These tools enable portal users to view and edit these models.
- **Goal 2:** we developed a visual method for explaining personalization to portal users. This method helps users to comprehend adaptive behavior of portals and raise user awareness of scrutable adaptivity.
- **Goal 3:** we validated our approach. We proofed that our approach allows users to scrutinize adaptivity in web portals in an easy-to-use and efficient way. Also, we determined that it makes significant impacts on the user's perception of personalized portals. Users are more satisfied with personalized portals that leverage our approach and find them more useful.

In this section, we summarize main contributions of this thesis. Also, we relate results of our research to theoretical foundations of adaptive systems. These contributions are sorted according to their importance for the field of adaptive systems and the degree of their elaboration in this thesis.

9.2.1. Scrutinization of User Models

We presented a novel interface for scrutinizing user models IntrospectiveViews (Section 6.3). IntrospectiveViews is a highly interactive visualization of large and complex semantic user models. It displays such models leveraging a metaphor of circular

zones partitioned into slices, where each zone represents items of certain interest degree and each slice represents items of a specific type. The visualization provides functions for getting overview, zooming, filtering, navigation, and search. It also displays relevant content and semantic relations among items. The metaphor and interaction patterns of IntrospectiveViews ensure that users can easily comprehend and interpret information stored in the user model. In addition to viewing, IntrospectiveViews allows users to edit the model in an easy-to-use and efficient manner. It allows adding and deleting items, changing status of items, organizing items by type, defining user own types, and creating semantic relations among items. Though we presented this interface using an example of user interest models, it can be also used for scrutinizing other types of feature-based user models, such as models of user knowledge and tasks. For example, in [11, 12, 68, 69], we describe applications of IntrospectiveViews for tracking student's progress and knowledge in e-learning environments.

Also, we presented results of three user studies evaluating various aspects of IntrospectiveViews. In the first study (Section 6.3.2), we evaluated the usefulness and usability of IntrospectiveViews as a generic interface for scrutable user modeling. Results of this study show that displaying content of user models groped by type or category helps users to perceive and understand it better. Such grouping also helps users to navigate through and search in user models. We found that users deem very useful to have an option for viewing content relevant to items in their models, e.g., related documents. They like to see what content was used for updating the user model. All users deem very important to be able to override the system-generated updates of user model. Users also like to be able to view semantic relations among items. However, users do not want these relations to be used for propagating the changes they make explicitly.

Results of the second study (Section 6.3.4) reveal the subjective usability and hedonic qualities of IntrospectiveViews. They show that users deem IntrospectiveViews as a motivating and easy-to-use realization of functions for scrutinizing user models. Users like the metaphor of displaying interests on circular zones partitioned into slices, where each zone represents items of certain interest degree and each slice represents items of a specific type. Users like the interaction pattern for changing interest degree, where they can set a higher interest by dragging items to the center and a lower interest by dragging them to the edge of the circle. Another interesting observation is that the subjective attractiveness of IntrospectiveViews is influenced by the ability of users to customize the interface, e.g., the ability to choose a color scheme they like for the circular zones.

In the third study (Section 8.1), we evaluated the application of IntrospectiveViews as a tool for scrutinizing adaptivity in a context of a personalized portal. Results of this study show that integrating IntrospectiveViews into a personalized system significantly improves the usefulness, usability, user satisfaction, and trans-

parency of the system. Even though inclusion of this interface into a personalized system increases the actual complexity of the system, users deem the system easier and simpler to use. We attribute this phenomenon to the gain of the outward attractiveness of the system achieved by IntrospectiveViews. A strong correlation between the perceived ease of use and the subjective attractiveness of computer systems was first detected by Kurosu and Kashimura [84] and then confirmed in a number of replicated experiments by Tractinsky [123]. These studies reveal that users deem a visually attractive system more usable than its analog with a less attractive appearance.

In the third study, we also measured the impacts on the trustworthiness of personalized systems. Despite IntrospectiveViews gives users unlimited access to the entire set of the system's beliefs stored in the user model, it does not improve the system's trustworthiness. From this study we found that some users do not care whether the system collects information about them as long as it uses it for recommending relevant content, whereas others have significant concerns about their privacy. Therefore, we can conclude that different users need different levels of control over their personal data.

9.2.2. Scrutinization of Personalization Rules and Effects

In this thesis, we elaborated an approach (Section 7.4) to scrutinizing personalization rules and effects in web portals. Our approach provides users an overview of all adaptations that the portal makes. Also, it allows users to influence these adaptations by either changing personalization rules or editing the user model. The tool for scrutinizing personalization is placed in the direct proximity to the personalized content. Any change that the user makes on personalization rules or the user model takes effect immediately on the personalized content. This helps users to understand the connection between these two components and the end personalization effects. Also, it allows them to easily adjust the adaptive behavior to their needs and preferences.

Our approach allows fine-tuning of personalization effects at the portlet level. Users can customize personalization rules and, if necessary, deactivate personalization completely for each portlet individually. In order to make this functionality available for other portlet programmers, we developed a *PersonalizablePortlet* class that implements all necessary methods for scrutinizing and controlling personalization in portlets. More specifically, the class implements a method for turning personalization on and off. It provides a method displaying rules that the portlet uses for personalization and the user model. Also, it provides a method that projects user changes on personalization rules or the user model on the personalized content. Portlet application developers can easily program portlets supporting this functionality by simply extending the *PersonalizablePortlet* class. Since we

implemented the *PersonalizablePortlet* class according to the Portlet Specification JSR286 [104], this class can be used for programming portlets for a broad spectrum of portal servers supporting this standard, including such servers as IBM WebSphere Portal, Liferay, GateIn, and many others.

Finally, we reported on results of a user study evaluating our approach to scrutinizing personalization (Section 8.1). These results show that a portal providing functions for personalization control is perceived by users more useful than a portal without these functions. Users like to be able to know what content is personalized and what adaptation effects are applied. They also like to be able to influence this. Finally, our approach improves the user satisfaction. Users deem a portal with functions for personalization control more pleasant and fun-to-use.

9.2.3. Domain Knowledge Modeling

Though in this thesis we mainly focused on the scrutability of user models and personalization rules, we made a number of interesting observations related to domain knowledge modeling. Results of the first two user studies of IntrospectiveViews (Sections 6.3.2 and 6.3.4) reveal that users want and need to be able to edit not only the interest status of items in the user model, but also semantics of these items. Users want to be able to create own categories and organize items in the user model according to these categories. Also, some users said that semantic relations among items represented by the system do not always reflect their personal associations. Hence, they want to be able to create own relations.

In the study of scrutable adaptivity (Section 8.1), we found that one of the factors that had a negative impact on the user satisfaction with the personalized portal for biochemical literature was the incorrect classification of certain entities (e.g., genes, enzymes, substrates, etc.) in the user profile. Users found it irritating. However, some of them said that they would be willing to correct the occasional misclassification by rearranging items in their interest profiles, i.e., by moving entities to the corresponding sectors on the IntrospectiveViews interface for user model visualization.

This willingness to change semantics of items in the user model can be leveraged by personalized systems in order to enrich the underlying domain ontology. The user-created categories can be converted into ontology classes. Items that users create under own categories can be inserted as instances of the corresponding classes. Finally, user-created relations between items can be translated into object properties connecting the corresponding instances in the ontology. The solution that we proposed in this thesis provides a suitable design and effective graphical user interfaces for user contributions to the domain knowledge. The domain model (Section 5.3.1) is designed in a way that allows storing user contributions in individual versions of the domain model. These contributions can be made through both the

user modeling interface IntrospectiveViews (Section 6.3.5) and the interface for tagging portal content (Section 5.4.3). Individual versions of domain knowledge can be statistically analyzed and aggregated in order to identify most frequent instances, classes of these instances, and instance relations. This information can be further used to update the shared version of the domain ontology.

9.3. Research Limitations

Although all goals of this research have been reached, it has some limitations and shortcomings.

First, since domain modeling and semantic annotation of portal content are out of the main focus of this dissertation, the implementation of the units for domain modeling and resource management includes only the components and functions that are required for user modeling and personalization. A realistic adaptive portal might require a more elaborate design and implementation of these units. Among others, it needs functions for update of the shared domain knowledge model that aggregate contributions of individual users. It also needs functions for merging user-generated annotations of portal content with the system-generated ones.

Second, in the implementation of the units for user modeling and personalization, we focused mainly on the scrutability functions. The implementation of the methods for automatic updates of user interests and the implementation of the personalization rules follow rather simplistic approaches. However, a realistic implementation of an adaptive portal might require more sophisticated algorithms. In its turn, this requires an extensive analysis of user modeling algorithms and techniques of recommender engines.

Third, due to the time constraints, it was unrealistic to conduct a full evaluation of the proposed approach in the context of this dissertation. The reported evaluation has a number of limitations, namely a rather small population, laboratory settings, and a short duration. In order to be able to draw more solid conclusions, this approach needs a long-term evaluation using a real system. It also should involve a larger number of users and leverage not only the users' self-reported metrics, but also the system's usage statistics.

9.4. Directions for Future Work

From the studies conducted in the scope of this thesis, we made a number of interesting observations with respect to the user perception of scrutable adaptivity. Results of these studies show that our approach successfully solves most of the problems of scrutable adaptivity and personalized systems in general. However,

they also show that some issues still remain unsolved or need to be further investigated. In this section, we discuss some of these issues and outline directions for future work.

9.4.1. Further Investigation of Trustworthiness

Results of the user study of scrutable adaptivity (Section 8.1) show that giving users control over the user model does not improve the trustworthiness of personalized systems. Even though users can view and edit the entire set of the system's beliefs stored in the user model, their trust in the system is still relatively low. We found that some users do not care whether the system collects information about them as long as it uses it for recommending relevant content, whereas others have significant concerns about their privacy. This reveals the need for a more flexible control over user's privacy and a more elaborate and comprehensive study of trustworthiness of personalized systems. Our hypothesis is that the trustworthiness could be improved if users could delegate the management of their personal data to some trusted authority or to manage it locally on their desktops.

9.4.2. Partitioning and Sharing of User Models

During the interviewing process of the user study of scrutable adaptivity (Section 8.1), the subjects made a number of interesting suggestions with respect to further enhancement of user modeling. Many subjects suggested allowing users to have multiple interest profiles, where each profile would represent interests or tasks related to one specific project. Also, respondents suggested having a group profile that reflects interests of a project team or a group of friends. In addition, some users are willing to share their interest profiles with their colleagues or friends. This function can be especially useful in corporate portals for senior staff and experts willing to share their knowledge with less experienced staff members, e.g., new employees or interns. We believe that the functions for partitioning user models into sub-profiles and functions for sharing user models is an interesting aspect of scrutable user modeling, hence it should be further investigated.

9.4.3. Analysis of User Contributions to the Domain Knowledge

Results of our studies show a clear-cut evidence of user willingness to contribute domain semantics through the user modeling interface IntrospectiveViews (Section 6.3). Users want to be able to create their own categories, organize items according to these categories, and create own relations among items. The last version of the interface provides this functionality. Also, the design of the domain knowledge model proposed in this thesis enables the storage of user-created semantics. This semantics is stored in individual versions of the domain ontology. Due to

the limited time frame of this work, we did not elaborate methods for aggregation of individual versions. However, we consider it as a promising direction for future research. We believe that analysis of user-created versions of domain ontology can enrich the domain knowledge significantly. It could help finding relevant ontology classes and instances. It could also help classifying knowledge elements and detecting relations among them.

9.4.4. Long-term Evaluation of Scrutable Adaptivity

We conducted a thorough usability evaluation of IntrospectiveViews. Also, we evaluated impacts of our approach as a whole on the usefulness, usability, user satisfaction, and trust of personalized systems. Though these studies provide valuable insights on scrutable adaptivity, they have a number of limitations. First, the number of subjects that took part in these studies is relatively low. Second, they were conducted in a rather laboratory settings using prototypes. Third, they run for a relatively short period of time. In order to be able to draw more solid conclusions, our approach needs to be evaluated in a study using a real system. This study should involve a larger number of users and run for a longer period of time. Also, in addition to user self-reported metrics, this study should leverage the system's usage statistics. This study may help to answer a number of important questions, such as the following ones. Do users prefer to have content in personalized or impersonalized states given a function for turning personalization on and off? Does the content type makes any influence on the user's decision to personalize or not personalize? Can this decision be influenced by the user's demographics and how? How often do users access their user profiles? What operations do they perform on the profiles? How do the profiles evolve over time? These and many other questions could be answered in a long-term study.

9.5. Concluding Remarks

In this thesis, we explored several aspects of scrutinizing adaptivity in community-enabled web portals. We proposed methods for making adaptivity more understandable and transparent. Also, we elaborated methods for letting portal users influence adaptivity in order to adjust it to their needs and preferences. They enable users to view and adapt personalization models in an easy-to-use and efficient manner. Users are granted full control over the user model and personalization rules. They can also contribute to the portal's domain knowledge and provide metadata of portal content.

We developed these methods in order to solve the problems that adaptive systems have been criticized for. By developing these methods we aimed to improve the transparency and predictability of adaptive systems. We aimed to improve the

trustworthiness of personalized systems by giving users a control over the personal information that these systems collect about them. Also, we intended to improve the quality of personalization and make the adaptivity better suit the user's attitude towards the proactive behavior of personalized systems.

Results of our studies show that our approach successfully solves most of these problems. It makes the adaptivity more transparent and understandable. It improves the usefulness of personalized portals and makes users be more satisfied with them. However, we also found that granting users full control over personalization models, including the user model, alone does not improve the trustworthiness of personalized systems. Users need better mechanisms to manage their personal information. Privacy management still remains an important issue for adaptive systems.

To conclude, we hope that findings of our studies will serve a strong motivation for the industry to give users more control over personalization in portals. We hope that the methods, graphical user interfaces, and interaction patters proposed in this thesis will help to design better and more user friendly portals. We believe that some of our solutions can also help designing scrutable adaptivity in other types of systems, for example, in social networks, search engines, e-learning systems, and many others.

References

[1] Act on the Protection of Personal Data Used in Teleservices (Gesetz über den Datenschutz bei Telediensten). Federal Law Gazette (Bundesgesetzblatt). Translation retrieved from http://www.iuscomp.org/gla/statutes/TDDSG.htm on April 12, 2012., 1997.

[2] P. Achananuparp, H. Han, O. Nasraoui, and R. Johnson. Semantically enhanced user modeling. In *Proc. of the 2007 ACM symposium on Applied computing*, 2007.

[3] E. Agirre, O. Ansa, E. Hovy, and D. Martínez. Enriching very large ontologies using the WWW. In *Proc. of ECAI Workshop on Ontology Learning*, 2000.

[4] J. Ahn and P. Brusilovsky. Adaptive visualization of search results: Bringing user models to visual analytics. *Information Visualization*, 8(3):167–179, 2009.

[5] J. Ahn, P. Brusilovsky, J. Grady, D. He, and S.Y. Syn. Open user profiles for adaptive news systems: help or harm? In *Proc. of the 16th int. Conf. on World Wide Web*, 2007.

[6] L. Ardissono, L. Console, and L. Torre. An adaptive system for the personalized access to news. *AI Communications*, 14:129–147, 2001.

[7] L. Ardissono and A. Goy. Tailoring the interaction with users in web stores. *User Modeling and User-Adapted Interaction*, 10(4):251–303, 2000.

[8] L. Aroyo, N. Stash, Y. Wang, P. Gorgels, and L. Rutledge. CHIP demonstrator: semantics-driven recommendations and museum tour generation. In *Proc. of the 6th International Semantic Web Conference*, 2007.

[9] S. Auer, S. Dietzold, and T. Riechert. OntoWiki - a tool for social, semantic collaboration. In *Proc. of the 5th International Semantic Web Conference*, 2006.

[10] E.H. Baehrecke, N. Dang, K. Babaria, and B. Shneiderman. Visualization and analysis of microarray and gene ontology data with treemaps. *BMC Bioinformatics*, 5(84), 2004.

[11] F. Bakalov, I.-H. Hsiao, P. Brusilovsky, and B. König-Ries. Progressor: Personalized visual access to programming problems. In *Proc. of the IEEE Symposium on Visual Languages and Human-Centric Computing*, 2011.

[12] F. Bakalov, I.-H. Hsiao, P. Brusilovsky, and B. König-Ries. Visualizing student models for social learning with Parallel IntrospectiveViews. In *Proc. of the Workshop on Visual Interfaces to the Social and Semantic Web held in conj. with the Int. Conf. on Intelligent User Interfaces*, 2011.

[13] F. Bakalov, B. König-Ries, T. Hennig, and G. Schade. Usability study of a semantic user model visualization for social networks. In *Workshop on Visual Interfaces to the Social and Semantic Web held in conj. with the Int. Conf. on Intelligent User Interfaces*, 2011.

[14] F. Bakalov, B. König-Ries, A. Nauerz, and M. Welsch. Ontology-based multidimensional personalization modeling for the automatic generation of mashups in next-generation portals. In *Proc. of the 1st Int. Workshop on Ontologies in Interactive Systems held in conj. with HCI*, 2008.

[15] F. Bakalov, B. König-Ries, A. Nauerz, and M. Welsch. A hybrid approach to identifying user interests in web portals. In *Proc. of the 9th Int. Conf. on Innovative Internet Community Systems*, Jena, Germany, 2009.

[16] F. Bakalov, B. König-Ries, A. Nauerz, and M. Welsch. A user modeling server for determination of semantically-enriched user interests. In *Proc. of the Int. Workshop on Ubiquitous User Modeling held in conj. with the 1st and 17th Int. Conf. on User Modeling, Adaptation and Personalization*, 2009.

[17] F. Bakalov, B. König-Ries, A. Nauerz, and M. Welsch. An interactive ring-shaped interface for visualization of semantically-enriched user interest models. Patent Application DE9-2010-014 (under evaluation), 2010.

[18] F. Bakalov, B. König-Ries, A. Nauerz, and M. Welsch. IntrospectiveViews: An interface for scrutinizing semantic user models. In *Proc. of the 18th Int. Conf. on User Modeling, Adaptation, and Personalization*, 2010.

[19] F. Bakalov, B. König-Ries, A. Nauerz, and M. Welsch. Scrutinizing user interest models with IntrospectiveViews. In *Adjunct proc. of the 18th Int. Conf. on User Modeling, Adaptation, and Personalization*, 2010.

[20] F. Bakalov, B. König-Ries, A. Nauerz, and M. Welsch. A visual rich-interaction approach to user-controlled personalization in web portals. Patent Application DE9-2010-0095 (under evaluation), 2010.

[21] F. Bakalov, M.-J. Meurs, B. König-Ries, B. Sateli, R. Witte, G. Butler, and A. Tsang. Personalized semantic assistance for curation of biochemical literature. In *IEEE Int. Conf. on Bioinformatics and Biomedicine*, 2012. (under review).

[22] F. Bakalov, B. Sateli, R. Witte, M.-J. Meurs, and B. König-Ries. Natural language processing for semantic assistance in web portals. In *6th IEEE Int. Conf. on Semantic Computing*, 2012.

[23] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American Magazine*, 5, 2001.

[24] D. Bogdanov, M. Haro, F. Fuhrmann, A. Xambó, E. Gómez, and P. Herrera. A content-based system for music recommendation and visualization of user preferences working on semantic notions. In *9th Int. Workshop on Content-Based Multimedia Indexing (CBMI)*, 2011.

[25] W. Borst. *Construction of Engineering Ontologies for Knowledge Sharing and Reuse*. PhD thesis, University of Twente, 1997.

[26] C. Boyle and A.O. Encarnacion. Metadoc: An adaptive hypertext reading system. *User Modeling and User-Adapted Interaction*, 4(1):1–19, 1994.

[27] S. Braun, A. Schmidt, and A. Walter. Ontology maturing: a collaborative web 2.0 approach to ontology engineering. In *Proc. of the WWW Workshop on Social and Collaborative Construction of Structured Knowledge*, 2007.

[28] J.G. Breslin, A. Passant, and S. Decker. *The Social Semantic Web*. Springer, 2009.

[29] P. Brusilovsky. Methods and techniques of adaptive hypermedia. *User Modeling and User-Adapted Interaction*, 6(2-3):87–129, 1996.

[30] P. Brusilovsky. Developing adaptive educational hypermedia systems: From design models to authoring tools. In T. Murray, S. Blessing, and S. Ainsworth, editors, *Authoring Tools for Advanced Technology Learning Environments: Toward cost-effective adaptive, interactive, and intelligent educational software*, pages 377–409. Kluwer, 2003.

[31] P. Brusilovsky, G. Chavan, and R. Farzan. Social adaptive navigation support for open corpus electronic textbooks. In *Proc. of the 3rd Int. Conf. on Adaptive Hypermedia and Adaptive Web-Based Systems*, 2004.

[32] P. Brusilovsky and D.W. Cooper. Domain, task, and user models for an adaptive hypermedia performance support system. In *Proc. of the 7th Int. Conf. on Intelligent User Interfaces*, 2002.

[33] P. Brusilovsky and E. Millàn. User models for adaptive hypermedia and adaptive educational systems. In *The Adaptive Web: Methods and Strategies of Web Personalization*, pages 3–53. Springer, 2007.

[34] J. Budzik, K.J. Hammond, and L. Birnbaum. Information access in context. *Knowledge-Based Systems*, 14(1-2):37–53, 2001.

[35] S. Bull and M. Britland. Group interaction prompted by a simple assessed open learner model that can be optionally released to peers. In *Proc. of Workshop on Personalisation in E-Learning Environments at Individual and Group Level (PING) held in conj. with the 11th Int. Conf. on User Modeling*, 2007.

[36] S. Bull and J. Kay. Student models that invite the learner in: The SMILI open learner modelling framework. *International Journal of Artificial Intelligence in Education*, 17(2):89–120, 2007.

[37] S. Bull, A. Mabbott, and A.S. Abu Issa. UMPTEEN: Named and anonymous learner model access for instructors and peers. *Int. Journal of Artificial Intelligence in Education*, 17(2):227–253, 2007.

[38] S. Bull and M. McKay. An open learner model for children and teachers: Inspecting knowledge level of individuals and peers. In *Proc. of the 7th Int. Conf. on Intelligent Tutoring Systems*, 2004.

[39] S. Bull and H. Pain. 'Did I Say What I Think I Said, And Do You Agree With Me?': Inspecting and questioning the student model. In *Proc. of the World Conf. on Artificial Intelligence and Education*, 1995.

[40] A. Bunt, G. Carenini, and C. Conati. Adaptive content presentation for the web. In *The Adaptive Web: Methods and Strategies of Web Personalization*, pages 409–432. Springer, 2007.

[41] R.D. Burke. Knowledge-based recommender systems. In *Encyclopedia of Library and Information Science*, 2001.

[42] F. Carmagnola, F. Cena, L. Console, O. Cortassa, C. Gena, A. Goy, I. Torre, A. Toso, and F. Vernero. Tag-based user modeling for social multi-device adaptive guides. *User Modeling and User-Adapted Interaction*, 18(5):497–538, 2008.

[43] C. Carmona, D. Bueno, E. Guzman, and R. Conejo. SIGUE: Making web courses adaptive. In *Proc. of the 2nd Int. Conf. on Adaptive Hypermedia and Adaptive Web-Based Systems*, 2002.

[44] C. Chen, M. Chen, and Y. Sun. Pva: A self-adaptive personal view agent. *Journal of Intelligent Information Systems*, 18(2-3):173–194, 2002.

[45] K. Cheverst, H.E. Byun, D. Fitton, D. Sas, C. Kray, and N. Villar. Exploring issues of user model transparency and proactive behaviour in an office environment control system. *User Modeling and User-Adapted Interaction*, 15(3-4):235–273, 2005.

[46] J. Collins, J. Greer, V. Kumar, G. McCalla, P. Meagher, and R. Tkatch. Inspectable user models for just-in-time workplace training. In *Proc. of the 6th Int. Conf. on User Modeling*, 1997.

[47] J. Conklin. Hypertext: An introduction and survey. *IEEE Computer*, 20:17–41, 1987.

[48] R. Cook and J. Kay. The justified user model: A viewable, explained user model. In *Proc. of the 4th Int. Conf. on User Modeling*, 1994.

[49] A. Corbett and J. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, 4(4):253–278, 1994.

[50] M. Czarkowski. *A scrutable adaptive hypertext.* PhD thesis, University of Sydney, 2006.

[51] C. da Costa Pereira and A. Tettamanzi. An evolutionary approach to ontology-based user model acquisition. In *Proc. of the 5th Int. Conf. on Fuzzy Logic and Applications*, 2003.

[52] K. Dave, S. Lawrence, and D.M. Pennock. Mining the peanut gallery: opinion extraction and semantic classification of product reviews. In *Proc. of the 12th Int. Conf. on World Wide Web*, 2003.

[53] P. De Bra, A. Aerts, D. Smits, and N. Stash. AHA! version 2.0, more adaptation flexibility for authors. In *Proc. of the AACE ELearn'2002 Conference*, 2002.

[54] P. De Bra and L. Calvi. AHA! an open adaptive hypermedia architecture. *The New Review of Hypermedia and Multimedia*, 4:115–139, 1998.

[55] V. Dimitrova. STyLE-OLM: Interactive open learner modelling. *International Journal of Artificial Intelligence in Education*, 17(2):35–78, 2003.

[56] P. Dolog and W. Nejdl. Semantic web technologies for the adaptive web. In *The Adaptive Web: Methods and Strategies of Web Personalization*, pages 697–719. Springer, 2007.

[57] D. Fensel, F. van Harmelen, I. Horrocks, D. McGuinness, and P. Patel-Schneider. OIL: An ontology infrastructure for the semantic web. *IEEE Intelligent Systems*, 16(2):38–45, 2001.

[58] E. Gansner and S. North. An open graph visualization system and its applications to software engineering. *Software – Practice & Experience*, 30(11):1203 – 1233, 1999.

[59] S.A. Golder and B.A. Huberman. Usage patterns of collaborative tagging systems. *Journal of Information Science*, 32(2):198 – 208, 2006.

[60] A. Gómez-Pérez, M. Fernández-López, and O. Corcho. *Ontological Engineering*. Springer, 2004.

[61] A. Goy, L. Ardissono, and G. Petrone. Personalization in e-commerce applications. In *The Adaptive Web: Methods and Strategies of Web Personalization*, pages 485–520. Springer, 2007.

[62] M. Greaves, L. Ding, J. Bao, and U. Bojars, editors. *Proc. of AAAI-SSS-09: Social Semantic Web: Where Web 2.0 Meets Web 3.0*, 2009.

[63] T. Gruber. Toward principles for the design of ontologies used for knowledge. *International Journal of Human-Computer Studies*, 43(5-6):907–928, 1995.

[64] T. Gruber. Where the social web meets the semantic web. In *Proc. of the 5th Int. Semantic Web Conf.*, 2006.

[65] Tobias Hennig. Usability Engineering am Beispiel einer Web 3.0-Applikation (in German). Master's thesis, Erfurt University of Applied Sciences, 2010.

[66] N. Henze and W. Nejdl. Adaptation in open corpus hypermedia. *Int. Journal of Artificial Intelligence in Education*, 12(4):325–350, 2001.

[67] K. Höök. Steps to take before intelligent user interfaces become real. *Interacting with Computers*, 12(4):64–67, 2000.

[68] I.-H. Hsiao, F. Bakalov, P. Brusilovsky, and B. König-Ries. Open social student modeling: Visualizing student models with Parallel IntrospectiveViews. In *Proc. of the 19th Int. Conf. on User Modeling, Adaptation, and Personalization*, 2011.

[69] I.-H. Hsiao, G. Julio, D. Parra, F. Bakalov, B. König-Ries, and P. Brusilovsky. Comparative social visualization for personalized e-learning. In *Proc. of the 11th Int. Working Conf. on Advanced Visual Interfaces*, 2012.

[70] A. Jameson. Adaptive interfaces and agents. In *The human-computer interaction handbook: Fundamentals, evolving technologies and emerging applications (2nd ed.)*, pages 433–458. Erlbaum, 2003.

[71] T. Joachims, D. Freitag, and T. Mitchell. WebWatcher: A tour guide for the world wide web. In *Proc. of the 15th Int. Joint Conf. on Artificial Intelligence*, 1997.

[72] A. Katifori, C. Halatsis, G. Lepouras, G. Vassilakis, and E. Giannopolou. Ontology visualization methods—a survey. *ACM Computing Surveys (CSUR)*, 39(4):10:1–10:43, 2007.

[73] J. Kay. The um toolkit for cooperative user modelling. *User Modeling and User-Adapted Interaction*, 4(3):149–196, 1994.

[74] J. Kay and B. Kummerfeld. Mneme: a framework to support people in their Sisyphean Tasks. Technical report, University of Sydney, 2010.

[75] J. Kay and A. Lum. Building user models from observations of users accessing multimedia learning objects. In *Adaptive Multimedia Retrieval*, pages 239–250. Springer, 2004.

[76] J. Kay and A. Lum. Ontology-based user modelling for the semantic web. In *Proc. of the Workshop on Personalisation on the Semantic Web*, 2005.

[77] A. Kerly and S. Bull. Children's interactions with inspectable and negotiated learner models. In *Proc. of the 9th Int. Conf. on Intelligent Tutoring Systems*, 2008.

[78] J. Kietz, A. Maedche, and R. Volz. A method for semi-automatic ontology acquisition from a corporate intranet. In *In Proc. of EKAW-2000 Workshop on Ontologies and Text*, 2000.

[79] J. Kliger. Model planes and totem poles: Methods for visualizing user models. Master's thesis, MIT Media Lab, 1995.

[80] A. Kobsa. Personalized hypermedia and international privacy. *Commun. ACM*, 45(5):64–67, 2002.

[81] M. Kravcik and M. Specht. Experience with winds virtual university. In *Proc. of the ED-MEDIA 2005 Conference*, 2005.

[82] A. Kreiser. Combining collaborative tagging with semantic web technology in web portals (in German). Master's thesis, Reutlingen University, 2009.

[83] A. Kreiser, A. Nauerz, F. Bakalov, B. König-Ries, and M. Welsch. A web 3.0 approach for improving tagging systems. In *Proc. of the Int. Workshop on Web 3.0: Merging Semantic Web and Social Web (in conjunction with the 20th Int. Conf. on Hypertext and Hypermedia 2009)*, 2009.

[84] M. Kurosu and K. Kashimura. Apparent usability vs. inherent usability. In *Proc. of the ACM SIGCHI Conf. on Human Factors in Computing Systems (CHI'95)*, 1995.

[85] G. Linden, J. Jacobi, and E. Benson. Collaborative recommendations using item-to-item similarity mappings. US Patent 6,266,649 (to Amazon.com), 2001. US Patent 6,266,649.

[86] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7:76–80, 2003.

[87] D. Lonsdale, Y. Ding, D. Embley, and A. Melby. Peppering knowledge sources with salt: Boosting conceptual content for ontology generation. In *Semantic Web meets Language Resources*, 2002.

[88] A. Lund. Measuring usability with the USE questionnaire. *Usability Interface*, 8(2), 2001.

[89] A. Mabbott and S. Bull. Alternative views on knowledge: Presentation of open learner models. In *Proc. of the 7th International Conference on Intelligent Tutoring Systems*, 2004.

[90] G. MacGregor and E. McCulloch. Collaborative tagging as a knowledge organization and resource discovery. *Library View*, 5(5):291 – 300, 2006.

[91] D. McDonald and M. Ackerman. Expertise recommender: a flexible recommendation system and architecture. In *Proc. of the 2000 ACM Conf. on Computer Supported Cooperative Work*, 2000.

[92] S. McNee, S. Lam, J. Konstan, and J. Riedl. Interfaces for eliciting new user preferences in recommender systems. In *Proc. of the 9th Int. Conf. on User Modeling*, 2003.

[93] M.-J. Meurs, C. Murphy, I. Morgenstern, G. Butler, J. Powlowski, A. Tsang, and R. Witte. Semantic text mining support for lignocellulose research. *BMC Medical Informatics and Decision Making*, 12(Suppl 1):35–42, 2012.

[94] A. Micarelli, F. Sciaronne, and M. Marinilli. Web document modeling. In *The Adaptive Web: Methods and Strategies of Web Personalization*, pages 155–192. Springer, 2007.

[95] P. Mika. *Social Networks and the Semantic Web*. Springer, 2007.

[96] T.M. Mitchell, R. Caruana, D. Freitag, J. McDermott, and D. Zabowski. Experience with a learning personal assistant. *Communications of the ACM*, 37(7):80 – 91, 1994.

[97] R. Mizoguchi, J. Vanwelkenhuysen, and M. Ikeda. Task ontology for reuse of problem solving knowledge. In *Towards Very Large Knowledge Bases–Knowledge Building and Knowledge Sharing*, 1995.

[98] S. Morinaga, K. Yamanishi, K. Tateishi, and T. Fukushima. Mining product reputations on the web. In *Proc. of the 8th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, 2002.

[99] A. Moukas. Amalthaea: Information discovery and filtering using a multiagent evolving ecosystem. *Applied Artificial Intelligence*, 11:437–457, 1997.

[100] T. Munzner. Exploring large graphs in 3d hyperbolic space. *IEEE Computer Graphics and Applications*, 18(4):18–23, 1998.

[101] R.Y. Nakamoto, S. Nakajima, J. Miyazaki, S. Uemura, H. Kato, and Y. Inagaki. Reasonable tag-based collaborative filtering for social tagging systems. In *Proc. of the 2nd ACM Workshop on Information Credibility on the Web held in conj. with the Conf. on Information and Knowledge Management*, 2008.

[102] A. Nauerz, F. Bakalov, B. König-Ries, and M. Welsch. Personalized recommendation of related content based on automatic metadata extraction. In *Proc. of the 18th Int. Conf. on Computer Science and Software Engineering*, 2008.

[103] A. Nauerz, M. Welsch, and B. König-Ries. Adaptive portals: Context adaptive navigation through large information spaces. In *Proc. of the 5th Int. Conf. on Adaptive Hypermedia and Adaptive Web-Based Systems*, 2008.

[104] M.S. Nicklous. JSR 286: Java Portlet Specification Version 2.0, 2007.

[105] J. Oberlander, M. O'Donnell, C. Knott, and C. Mellish. Conversation in the museum: Experiments in dynamic hypermedia with the intelligent labelling explorer. *The New Review of Hypermedia and Multimedia*, 4:11–32, 1998.

[106] C. Panayiotou and G. Samaras. mPERSONA: personalized portals for the wireless user: An agent approach. *Mobile Networks and Applications*, 9(6):663–67, 2004.

[107] J. Park, T. Fukuhara, I. Ohmukai, H. Takeda, and S. Lee. Web content summarization using social bookmarks: a new approach for social summarization. In *Proc. of the 10th ACM Workshop on Web Information and Data Management*, 2008.

[108] D. Parra and P. Brusilovsky. Collaborative filtering for social tagging systems: an experiment with CiteULike. In *Proc. of the 3rd ACM Conf. on Recommender Systems*, 2009.

[109] S. Pfeiffer. Rating sites flourish behind a veil of anonymity. *The Boston Globe*, 2006.

[110] M. Rey-López, R.P. Díaz-Redondo, A. Fernández-Vilas, J.J. Pazos-Arias, M. López-Nores, J. García-Duque, and A. Gil-Solla. T-MAESTRO and its authoring tool: using adaptation to integrate entertainment into personalized e-learning. *Multimedia Tools and Applications*, 40(3):409–451, 2008.

[111] U. Rueda, M. Larrañaga, A. Arruarte, and J. A. Elorriaga. DynMap+: A concept mapping approach to visualize group student models. In *Proc. of the 1st European Conference on Technology Enhanced Learning, Innovative Approaches for Learning and Knowledge Sharing*, 2006.

[112] A. Saha. Application framework for e-business: Portals. IBM developerWorks, 1999. Retrieved from http://www.ibm.com/developerworks/web/library/wa-portals/ on 03.02.2010.

[113] G. Salton and M. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[114] J.B. Schafer, D. Frankowski, J. Herlocker, and S. Sen. Collaborative filtering recommender systems. In *The Adaptive Web: Methods and Strategies of Web Personalization*, pages 291–324. Springer, 2007.

[115] O. Schimratzki. An approach for semantic enrichment of social media resources for context dependent processing. Master's thesis, Friedrich Schiller University of Jena, 2010.

[116] O. Schimratzki, F. Bakalov, A. Knoth, and B. König-Ries. Semantic enrichment of social media resources for adaptation. In *Proc. of the Workshop on Adaptation in Social and Semantic Web held in conj. with the 18th Int. Conf. on User Modeling, Adaptation, and Personalization*, 2010.

[117] J. Schreck. *Security and Privacy in User Modeling*. PhD thesis, University of Essen, 2001.

[118] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proc. of the 1996 IEEE Symposium on Visual Languages*, 1996.

[119] K. Siorpaes, M. Hepp, A. Klotz, and M. Hackl. myOntology: Tapping the wisdom of crowds for building ontologies. Technical report, STI, 2008.

[120] M. Specht, M. Kravcik, R. Klemke, L. Pesin, and R. Hüttenhain. Adaptive learning environment for teaching and learning in WINDS. In *Proc. of the 2nd Int. Conf. on Adaptive Hypermedia and Adaptive Web-Based Systems*, 2002.

[121] R. Studer, R. Benjaminsc, and D. Fensel. Knowledge engineering: Principles and methods. *Data and Knowledge Engineering*, 25(1-2):161–197, 1998.

[122] Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, and D. Wenke. OntoEdit: Collaborative ontology engineering for the semantic web. In *Proc. of the International Semantic Web Conference*, 2002.

[123] N. Tractinsky. Aesthetics and apparent usability: empirically assessing cultural and methodological issues. In *Proc. of the ACM SIGCHI Conf. on Human Factors in Computing Systems (CHI'97)*, 1997.

[124] T. Tsandilas and M. C. Schraefel. Usable adaptive hypermedia systems. *New Review of Hypermedia and Multimedia*, 10(1):5–29, 2004.

[125] M. Tvarozek, M. Barla, G. Frivolt, M. Tomsa, and M. Bielikova. Improving semantic search via integrated personalized faceted and visual graph navigation. In *Proc. of the 34th Conf. on Theory and Practice of Computer Science*, 2008.

[126] J. Uther and J. Kay. VlUM, a web-based visualisation of large user models. In *Proc. of the 9th Int. Conf. on User Modeling*, 2003.

[127] G. van Heijst, A. Schreiber, and B. Wielinga. Using explicit ontologies in KBS development. *International Journal of Human-Computer Studies*, 46(2-3):183–292, 1997.

[128] T. Vander Wal. Folksonomy coinage and definition. Website, February 2 2007. Retrieved from http://vanderwal.net/folksonomy.html on February 10, 2010.

[129] M. Völkel, M. Krötzsch, D. Vrandecic, H. Haller, and R. Studer. Semantic wikipedia. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(4):251–261, 2007.

[130] G. Weber and P. Brusilovsky. ELM-ART: An adaptive versatile system for web-based instruction. *International Journal of Artificial Intelligence in Education*, 12(4):351–384, 2001.

[131] G. Weber, H.-C. Kuhl, and S. Weibelzahl. Developing adaptive internet based courses with the authoring system NetCoach. In *Proc. of the 3rd Workshop on Adaptive Hypertext and Hypermedia*, 2001.

[132] R. Witte and T. Gitzinger. Semantic Assistants – user-centric Natural Language Processing services for desktop clients. In *3rd Asian Semantic Web Conference (ASWC 2008)*, volume 5367 of *LNCS*, pages 360–374, Bangkok, Thailand, 2008. Springer.

[133] J. Zapata-Rivera and J. Greer. Externalising learner modelling representations. In *Proc. of the Workshop on External Representations held in conj. with Int. Conf. on Artificial Intelligence in Education*, 2001.

Appendix

APPENDIX A

Domain Ontology for the News Aggregating Portal

Ontology IRI: http://www.minerva-portals.de/o/upper.owl
Format : RDF/XML
Namespaces
owl: ¡http://www.w3.org/2002/07/owl¿
protons: ¡http://proton.semanticweb.org/2005/04/protons¿
protont: ¡http://proton.semanticweb.org/2005/04/protont¿
protonu: ¡http://proton.semanticweb.org/2005/04/protonu¿
news: ¡http://www.minerva-portals.de/o/upper.owl¿

```
owl:Thing
    protons:Entity
        protont:Happening
            protont:Event
                protonu:ArtPerformance
                    news:EntertainmentAwardEvent
                    protonu:Concert
                    protonu:OperaPerformance
                    protonu:TheatrePerformance
                news:Disaster
                    news:NaturalDisaster
                news:PoliticalEvent
                news:Anniversary
                protonu:Meeting
                    protonu:OfficialPoliticalMeeting
                    protonu:BoardMeeting
                news:Holiday
                protonu:MilitaryConflict
```

```
                protonu:Project
                protonu:SportEvent
                    protonu:Tournament
                    protonu:OlympicGames
                    protonu:SportGame
                protonu:Accident
            protont:TimeInterval
                protonu:CalendarYear
                protonu:Week
                protonu:Date
                protonu:Quarter
                protonu:Month
            protont:Situation
                news:MedicalTreatment
                protont:Role
                protont:JobPosition
                    protonu:BoardMember
                    protonu:MemberOfParliament
                    protonu:Leader
                        protonu:President
                        protonu:Executive
                            protonu:Premier
                            protonu:CEO
                            protonu:Minister
                        protonu:Chairman
                    protonu:Employee
                        protonu:Manager
                            protonu:Executive
                                protonu:Premier
                                protonu:CEO
                                protonu:Minister
                    protonu:OfficialPosition
                news:MedicalCondition
        protont:Abstract
            protonu:BusinessAbstraction
                protonu:Market
                protonu:IndustrySector
            protonu:NaturalPhenomenon
                protonu:Disease
            protonu:SocialAbstraction
                news:Technology
                protonu:ResearchArea
                    protonu:Science
                protonu:Sport
                protonu:Money
                protonu:Art
                protonu:Profession
            protont:Number
                protonu:Percent
            protont:ContactInformation
```

```
            protonu:Address
                protonu:PostalAddress
            news:FaxNumber
            protonu:PhoneNumber
            protonu:InternetAddress
                news:URL
                protonu:WebPage
                    protonu:HomePage
                protonu:EMail
                protonu:IPAddress
                protonu:InternetDomain
        protont:GeneralTerm
        protont:Language
            news:ProgrammingLanguage
        protonu:TemporalAbstraction
            protonu:CalendarMonth
            protonu:TimeZone
            protonu:Festival
            protonu:Season
            protonu:DayOfWeek
            protonu:DayOfMonth
            protonu:DayTime
        protont:Topic
            news:IndustryTerm
    protont:Object
        protont:Statement
            protonu:Offer
            protont:InformationResource
                protonu:Patent
                protonu:Legislation
                protonu:DataSchema
                protonu:ResourceCollection
                    protonu:MagazineIssue
                    protonu:MeetingProceedings
                    protonu:NewspaperIssue
                protonu:Dataset
                protont:Document
                    protonu:Message
                    protonu:Contract
                    protonu:Report
                    protonu:PublishedMaterial
                        protonu:Announcement
                        protonu:Article
                        protonu:Book
                            protonu:MeetingProceedings
                        protonu:IssueOfPeriodical
                            protonu:MagazineIssue
                            protonu:NewspaperIssue
            news:MarketIndex
            protonu:Order
```

protonu:PieceOfArt
protonu:Vehicle
    protonu:Spacecraft
    protonu:Ship
protonu:Currency
protonu:Brand
    protonu:MediaBrand
        protonu:RadioStation
        protonu:TVChannel
        protonu:PeriodicalPublication
            protonu:Newspaper
            protonu:Magazine
protont:Product
    protonu:ChemicalCompound
        protonu:Drug
        protonu:Beverage
            protonu:AlcoholicBeverage
    protonu:WeaponModelOrSystem
    protonu:AirplaneModel
    news:SoftwareProduct
        news:OperatingSystem
    protonu:CarModel
    protonu:MediaProduct
        protonu:MusicalComposition
            protonu:AudioRecording
                protonu:Album
                protonu:Song
        protonu:Movie
        news:RadioProgram
        news:TVProgram
        protonu:PublishedMaterial
            protonu:Announcement
            protonu:Article
            protonu:Book
                protonu:MeetingProceedings
            protonu:IssueOfPeriodical
                protonu:MagazineIssue
                protonu:NewspaperIssue
protont:Location
    protonu:LandRegion
        protonu:Valley
            protonu:Canyon
        protonu:Gap
        protonu:Coast
        protonu:Island
            protonu:Archipelago
        protonu:Isthmus
        protonu:Waterfalls
        protonu:Continent
        protonu:BiogeographicRegion

```
                protonu:Tundra
                protonu:Desert
                protonu:Jungle
                protonu:Oasis
                protonu:Forest
            protonu:Mountain
                protonu:MountainSummit
                protonu:MountainRange
                protonu:Ridge
                protonu:Volcano
            protonu:Plain
            protonu:Cape
            protonu:Cave
            protonu:Beach
            protonu:Peninsula
            protonu:Reef
            protonu:Delta
            protonu:Crater
            protonu:Plateau
            protonu:WaterBank
        protonu:PopulatedPlace
            protonu:City
                protonu:Capital
                    protonu:LocalCapital
                    protonu:CountryCapital
        protonu:Facility
            protonu:TelecomFacility
            protonu:Monument
            protonu:RecreationalFacility
                protonu:Camp
                protonu:SportFacility
                    protonu:SportBuilding
                        protonu:Stadium
                    protonu:RaceCourse
                protonu:AmusementPark
                protonu:PerformanceSite
            protonu:OilField
            protonu:ReligiousLocation
            protonu:Building
                protonu:SportBuilding
                    protonu:Stadium
                protonu:Tower
                protonu:Hotel
                protonu:PerformanceSite
            protonu:MineSite
            protonu:Park
            protonu:Reserve
            protonu:ReferenceLocation
            protonu:TransportFacility
                protonu:Bridge
```

```
                protonu:Airport
                protonu:LaunchFacility
                protonu:Lock
                protonu:Roadway
                    protonu:Street
                protonu:Pipeline
                protonu:Tunnel
                    protonu:RailroadTunnel
                protonu:RailroadFacility
                    protonu:RailroadTunnel
            protonu:HydrographicStructure
                protonu:Reservoir
                protonu:Canal
                protonu:Lock
                protonu:Harbor
        protonu:PoliticalRegion
            protonu:MilitaryAreas
            protonu:UrbanDistrict
            protonu:Country
            protonu:County
            protonu:Province
        protonu:GlobalRegion
        news:NaturalFeature
        protonu:NonGeographicLocation
        protonu:AstronomicalObject
            protonu:Galaxy
            protonu:Star
            protonu:Satellite
                protonu:NaturalSatellite
                protonu:ArtificialSatellite
            protonu:Planet
        protonu:WaterRegion
            protonu:Sea
                protonu:Ocean
            protonu:Lake
                protonu:Reservoir
            protonu:Channel
                protonu:Canal
            protonu:Stream
                protonu:Spring
                protonu:River
            protonu:Harbor
            protonu:Bay
                protonu:Fjord
            protonu:Archipelago
            protonu:Creek
            protonu:Gulf
    protont:Service
    protonu:Account
    protont:Agent
```

```
                protont:Person
                    protonu:Man
                    protonu:Woman
                protont:Group
                    protont:Organization
                        protonu:GovernmentOrganization
                            protonu:Government
                            protonu:Ministry
                        news:MusicGroup
                        protonu:ReligiousOrganization
                        protonu:EducationalOrganization
                            protonu:School
                            protonu:University
                        protonu:Charity
                        protonu:InternationalOrganization
                        protonu:SportOrganization
                            news:SportsLeague
                            protonu:SportClub
                                protonu:SoccerClub
                            protonu:SportsFederation
                        protonu:Team
                        protonu:StockExchange
                        protonu:Division
                        protonu:PoliticalEntity
                            protonu:PoliticalParty
                            protonu:Parliament
                        protonu:ResearchOrganization
                            protonu:Institute
                            protonu:University
                        protonu:CommercialOrganization
                            protonu:Company
                                protonu:MediaCompany
                                    news:RadioStation
                                    protonu:PublishingCompany
                                    protonu:NewsAgency
                                    protonu:TVCompany
                                protonu:Airline
                                protonu:PublicCompany
                                protonu:InsuranceCompany
                                protonu:Telecom
                                protonu:SportClub
                                    protonu:SoccerClub
                                protonu:Bank
protons:LexicalResource
    protons:Alias
protons:EntitySource
    protons:Recognized
    protons:Trusted
```

APPENDIX B

Semantic Types Mapping for OpenCalais

```
<?xml version="1.0" encoding="UTF-8"?>
<map>
        <mapping>
                <key>Anniversary</key>
                <value>min-up:Anniversary</value>
        </mapping>
        <mapping>
                <key>City</key>
                <value>pup:City</value>
        </mapping>
        <mapping>
                <key>Company</key>
                <value>pup:Company</value>
        </mapping>
        <mapping>
                <key>Continent</key>
                <value>pup:Continent</value>
        </mapping>
        <mapping>
                <key>Country</key>
                <value>pup:Country</value>
        </mapping>
        <mapping>
                <key>Currency</key>
                <value>pup:Currency</value>
        </mapping>
        <mapping>
                <key>EntertainmentAwardEvent</key>
                <value>min-up:EntertainmentAwardEvent</value>
```

```
        </mapping>
        <mapping>
                <key>Facility</key>
                <value>pup:Facility</value>
        </mapping>
        <mapping>
                <key>Holiday</key>
                <value>min-up:Holiday</value>
        </mapping>
        <mapping>
                <key>IndustryTerm</key>
                <value>min-up:IndustryTerm</value>
        </mapping>
        <mapping>
                <key>MarketIndex</key>
                <value>min-up:MarketIndex</value>
        </mapping>
        <mapping>
                <key>MedicalCondition</key>
                <value>min-up:MedicalCondition</value>
        </mapping>
        <mapping>
                <key>MedicalTreatment</key>
                <value>min-up:MedicalTreatment</value>
        </mapping>
        <mapping>
                <key>Movie</key>
                <value>pup:Movie</value>
        </mapping>
        <mapping>
                <key>MusicAlbum</key>
                <value>pup:Album</value>
        </mapping>
        <mapping>
                <key>MusicGroup</key>
                <value>min-up:MusicGroup</value>
        </mapping>
        <mapping>
                <key>NaturalDisaster</key>
                <value>min-up:NaturalDisaster</value>
        </mapping>
        <mapping>
                <key>NaturalFeature</key>
                <value>min-up:NaturalFeature</value>
        </mapping>
        <mapping>
                <key>OperatingSystem</key>
                <value>min-up:OperatingSystem</value>
        </mapping>
        <mapping>
```

```
            <key>Organization</key>
            <value>ptop:Organization</value>
        </mapping>
        <mapping>
            <key>Person</key>
            <value>ptop:Person</value>
        </mapping>
        <mapping>
            <key>PoliticalEvent</key>
            <value>min-up:PoliticalEvent</value>
        </mapping>
        <mapping>
            <key>Product</key>
            <value>ptop:Product</value>
        </mapping>
        <mapping>
            <key>ProgrammingLanguage</key>
            <value>min-up:ProgrammingLanguage</value>
        </mapping>
        <mapping>
            <key>ProvinceOrState</key>
            <value>pup:Province</value>
        </mapping>
        <mapping>
            <key>PublishedMedium</key>
            <value>pup:PublishedMaterial</value>
        </mapping>
        <mapping>
            <key>RadioProgram</key>
            <value>min-up:RadioProgram</value>
        </mapping>
        <mapping>
            <key>RadioStation</key>
            <value>pup:RadioStation</value>
        </mapping>
        <mapping>
            <key>Region</key>
            <value>pup:GlobalRegion</value>
        </mapping>
        <mapping>
            <key>SportsEvent</key>
            <value>pup:SportEvent</value>
        </mapping>
        <mapping>
            <key>SportsGame</key>
            <value>pup:Sport</value>
        </mapping>
        <mapping>
            <key>SportsLeague</key>
            <value>min-up:SportsLeague</value>
```

```
        </mapping>
        <mapping>
                <key>Technology</key>
                <value>min-up:Technology</value>
        </mapping>
        <mapping>
                <key>TVShow</key>
                <value>min-up:TVProgram</value>
        </mapping>
        <mapping>
                <key>TVStation</key>
                <value>pup:TVChannel</value>
        </mapping>
        <mapping>
                <key>SocialTag</key>
                <value>min-up:Misc</value>
        </mapping>
        <mapping>
                <key>SportsLeague</key>
                <value>min-up:SportsLeague</value>
        </mapping>
</map>
```

APPENDIX C

Questionnaire on Usefulness and Usability of IntrospectiveViews

IntrospectiveViews:
a Graphical User Interface for Scrutinizing Semantically-Enriched User Interest Models
Evaluation Form

Rate your expertise in user modeling and adaptation:	(don't know anything about)	1 2 3 4 5	(expert)
Rate your expertise in information visualization:	(don't know anything about)	1 2 3 4 5	(expert)
Rate your expertise in usability:	(don't know anything about)	1 2 3 4 5	(expert)

	#	Tasks / GUI Implementation of Tasks	Rate Importance 1. not at all important 2. not important 3. somewhat important 4. important 5. very important	Rate Usability 1. not at all usable 2. not usable 3. somewhat usable 4. usable 5. very usable
	O.1.	**Get an overview of the entire collection of user interests**	1 2 3 4 5	
	O.1.1.	All user interests are represented as terms (e.g. Germany, Berlin, Angela Merkel) and displayed as string labels on a circular surface		1 2 3 4 5
	O.1.2.	*Navigator* window displays a zoomed out view of the entire collection of user interests		1 2 3 4 5
	O.2.	**Get an overview of interest groups available in the user model**	1 2 3 4 5	
iew	O.2.1.	Interest groups are represented as rings, which are differently distanced from the circle center. Proximity to the center denotes higher interest.		1 2 3 4 5

	O.3.	**View fuzziness/uncertainty between interest groups**	1 2 3 4 5	
	O.3.1.	Fuzziness/uncertainty between interest groups is represented by the gradient color filling out the border areas of rings		1 2 3 4 5
	O.4.	**Get an overview of available types in the user model (e.g. countries, people, companies)**	1 2 3 4 5	
	O.4.1.	Terms can be grouped into circular sectors according to their type. Such sectors are distinguished by different shades of gray color and identified by a label containing type's name.		1 2 3 4 5
Zoom	**Z.1.**	**Zoom**	1 2 3 4 5	
	Z.1.1.	Zoom with zoom slider		1 2 3 4 5
	Z.1.2.	Zoom with mouse wheel		1 2 3 4 5
Filter	**F.1.**	**Filter terms by type (e.g. by Company, Person, Country, etc.)**	1 2 3 4 5	
	F.1.1.	Terms can be filtered based on their type by selecting/deselecting the corresponding checkboxes in *Types* window		1 2 3 4 5
	F.2.	**Filter terms by interest group**	1 2 3 4 5	
	F.2.1.	Terms can be filtered based on the interest degree by selecting/deselecting the corresponding checkboxes in *Interest Groups* window		1 2 3 4 5
Details-on-Demand	**D.1.**	**View semantically related terms**	1 2 3 4 5	
	D.1.1.	By clicking on a term, the semantic relations of this term to other terms will be shown as lines connecting the terms (try it with Bavaria or Germany)		1 2 3 4 5
	D.2.	**View terms with the same interest degree**	1 2 3 4 5	
	D.2.1.	By clicking on a term, the interface will display an orbit of the selected term and highlight all terms located on this orbit		1 2 3 4 5
	D.3.	**View textual description of terms**	1 2 3 4 5	
	D.3.1.	Displaying an excerpt from Wikipedia in the *InfoBox*, which is activated by clicking on a term with right mouse button (e.g. make right mouse click on Angela Merkel)*		1 2 3 4 5
	D.4.	**View graphical depiction of terms**	1 2 3 4 5	
	D.4.1.	Displaying a picture in the *InfoBox*, which is activated by clicking on a term with right mouse button (e.g. make right mouse click on Angela Merkel)*		1 2 3 4 5
	D.5.	**Get information about evidence of interest**	1 2 3 4 5	
	D.5.1	Displaying log-based / inference-based / manual update history in the *InfoBox*, which is activated by clicking on a term with right mouse button (e.g. make right mouse click on Angela Merkel)*		1 2 3 4 5
	D.6.	**View resources (containing a certain term) that the user has not viewed yet**	1 2 3 4 5	
	D.6.1.	Displaying a list of hyperlinks to the new resources in the *InfoBox*, which is activated by clicking on a term with right mouse button (e.g. make right mouse click on Angela Merkel)*		1 2 3 4 5
	D.7.	**View resources (containing a certain term) that the user has already viewed**	1 2 3 4 5	
	D.7.1.	Displaying a list of hyperlinks to the viewed resources in the *InfoBox*, which is activated by clicking on a term with right mouse button (e.g. make right mouse click on Angela Merkel)*		1 2 3 4 5
	D.8.	**View adaptation rules for term types (e.g. what type of resources are recommended about countries, how, and when)**	1 2 3 4 5	

Relate	**R.1.**	**Relate terms to their frequency in the browsing log (how often the term is present in the resources that user viewed in the past)**	1 2 3 4 5	
	R.1.1.	Term frequency is encoded in font size of labels		1 2 3 4 5
	R.2.	**Relate term types to their size (e.g. view how many companies are in the user model)**	1 2 3 4 5	
	R.2.1.	Terms can be grouped according to their type and displayed within circular sectors. The span of circular sectors denotes the type's size (number of terms of the type).		1 2 3 4 5
	R.3.	**Relate interest groups to their importance (overall interest value and number of terms)**	1 2 3 4 5	
	R.3.1.	Importance of interest group is encoded into the ring's width		1 2 3 4 5
	R.4.	**Relate the individual user interest model to the group interest model**	1 2 3 4 5	
History	**H.1.**	**View snapshot of user model in a certain time frame in the past**	1 2 3 4 5	
	H.1.1.	Date slider allows to obtain a view on a state of the model in a certain time in the past*		1 2 3 4 5
Extract	**E.1.**	**Export user interests**	1 2 3 4 5	
	E.1.1.	A certain set of user interests (all, displayed, selected) can be saved in a file on disk using *Export* window *		1 2 3 4 5
Modify	**M.1.**	**Change interest degree of a term**	1 2 3 4 5	
	M.1.1.	By dragging a label closer to or further from the circle center, user can increase or decrease interest degree		1 2 3 4 5
	M.2.	**Propagating interest change of semantically related terms**	1 2 3 4 5	
	M.2.1.	While dragging a label closer to of further from the circle center, interest degree of its related terms is changed too and their positions on the circle is adjusted accordingly (try it with Bavaria)		1 2 3 4 5
	M.2.1.	While dragging a label closer to of further from the circle center, interest degree of its related terms is changed too and their positions on the circle is adjusted accordingly (try it with Bavaria)		1 2 3 4 5
	M.3.	**Add new terms into the user interest model**	1 2 3 4 5	
	M.3.1.	New terms can be added into the user model by typing a term in an autocomplete combobox in *Add New Term* window and clicking *Add* button		1 2 3 4 5
	M.4.	**Block terms from being tracked and used for recommendation**	1 2 3 4 5	
	M.4.1.	Dragging terms onto the recycle bin will mark them as blocked. Such terms will not be tracked by the system and will not be used for any recommendation until user takes them back from the bin.*		1 2 3 4 5
	M.5.	**Modify adaptation rules for term types (e.g. change the type and representation of resources that can be recommended about countries)**	1 2 3 4 5	
	M.6.	**Modify color scheme (e.g. switch from hot-and-cold metaphor to traffic light metaphor)**	1 2 3 4 5	

* Implemented as a mockup