Integration of Constraints into Digital Building Models for Cooperative Planning Processes

A. Borrmann[2], Th. Hauschild[1], R. Hübler[1]
1) Bauhaus University Weimar, 99421 Weimar, Germany
2) now: Technical University of Munich, 80290 Munich, Germany.
borrmann@bv.tum.de, {thomas.hauschild|reinhard.huebler}@informatik.uni-weimar.de

Summary
The uniqueness and the long life cycle of buildings imply a dynamically modifiable building model. The technological foundation for the management of digital building models, a dynamic model management system (MMS), developed by our research group, allows to explicitly access and to modify the object model of the stored planning data. In this paper, the integration of constraints in digital building models will be shown. Constraints are conditions, which apply to the instances of domain model classes, and are defined by the user at runtime of the information system. For the expression of constraints, the Constraint Modelling Language (CML) has been developed and will be described in this paper. CML is a powerful, intuitively usable object-oriented language, which allows the expression of constraints at a high semantic level. A constrained-enabled MMS can verify, whether an instance fulfils the applying constraints. To ensure flexibility, the evaluation of constraints is not implicitly performed by the systems, but explicitly initiated by the user. A classification of constraint types and example usage scenarios are given.

1 Introduction
The planning of building is a process, which is characterised by complex tasks and a high degree of synchronous and asynchronous, local and geographical distributed collaboration of professionals from a number of special subjects. Digital building models, i.e. computer internal building representations, are a necessary precondition for computer support of these complex building planning and erection processes. These models form a base for the integration of building information and for the adequate supply of necessary and relevant information for the current design task and represent all available structural information at building planning projects (Figure 1). The high complexity of the building models and the request for models, which are able to accompany most phases of the building life cycle, imply exceptionally demands for their management.

In contrast to other engineering disciplines, the product is unique and not part of a series production. It is impossible to make a test specimen; most usability simulations have to be carried out based on the digital building model. In comparison to other engineering disciplines, there are few standardised and many singular building components. These circumstances result in further high demands to the flexibility of the building model and its management component, too.

Many planning projects are carried out in Virtual Enterprises, which exist for the period of the planning and construction processes. Some characteristics of Virtual Enterprises are temporary relationships between participants, dynamic involvement and retiring of members and limited chances to promote a common information infrastructure. A design environment for building planning processes has to meet the requirements for a support of planning activities in Virtual Enterprises.

The investigation of these modelling methods for building data and model management techniques are the subject of the research project “Digital Building Model for Inventory

Information", which is part of the Collaborative Research Centre 524 "Materials and Constructions for Building Revivification" at the Bauhaus – Universitity Weimar, Germany.

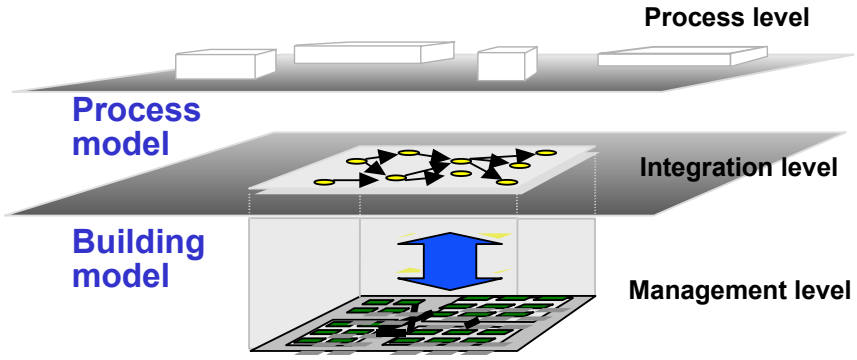

Figure 1: Process integration in building planning processes

2 Management of Dynamic Desciptive Building Models

For the reflection of the structural information the application of the object-oriented paradigm has proven to be an adequate foundation. The domain knowledge of the planning professionals can be represented in class taxonomies through abstraction processes. For descriptive models, these taxonomies mainly consist of classes with attributes and structures of inheritance, aggregation and association. The concerning building is represented in a specific project model by instances of taxonomy classes.

This way it is possible to manage generalized knowledge about building objects and to use it as a base for planning activities for present building projects. There is a necessity of dynamic in domain taxonomies regarding their structure and content (Steinmann 1997). Reasons are the very long building and model life cycles, changes in regulations and rules and the demand for systems, which are adjustable to special needs, specific structures or private know-how, for example.

2.1 Model management systems

Model management systems are the technological base for storage and retrieval of digital building models. The logical structure and the kind of managing the real model architecture have been specified before. This included above all the handling of domain knowledge taxonomies as well as concrete project data. The model management system has also to pay attention to the collaborative and distributed character of the planning processes and has to ensure the physical access to the planning releases of other co-workers without collisions. Other requirements are the support of information versioning, the guarantee of the physical model consistency and the provision for tools for the generation of digital model signatures, for example. In addition to the information handled by means of object-orientation, there is data like texts, digitised drawings, CAD files, photographs of damage areas, etc., which can hardly be formalised. This data is stored on the design support system and referenced by special attributes of classes and instances. An efficient retrieval and context dependend provision of the information is possible that way (Figure 2).

There are several approaches for the realisation of runtime dynamic model management kernels. On the one hand, it is possible to new-implement all necessary features of the kernel thoroughly by particular program modules of the system, on the other hand, the implementation can exploit relevant properties of several object-oriented languages. Besides, the realisation could be founded on suitable tools or libraries (Hauschild and Hübler 2002).

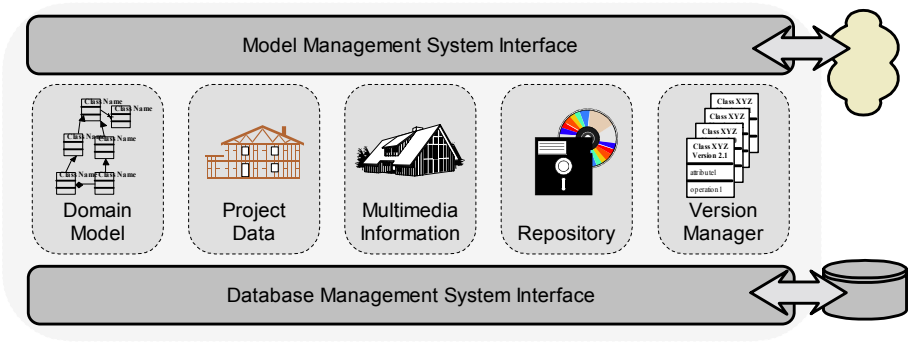

Figure 2: Model Management System kernel structure

The second realisation way was chosen due to experiences of very high implementation expenses and very complex optimisation tasks in preceding research efforts, which utilised the first way. All information in the model management kernel can be locally and remote accessed by means of a static CORBA interface. This interface encapsulates the meta level modelling functionality and is a further developed version of the AKO–interface (Ranglack et al. 1997) (Figure 3), which defines all necessary functionalities for the management of taxonomies and instance sets.

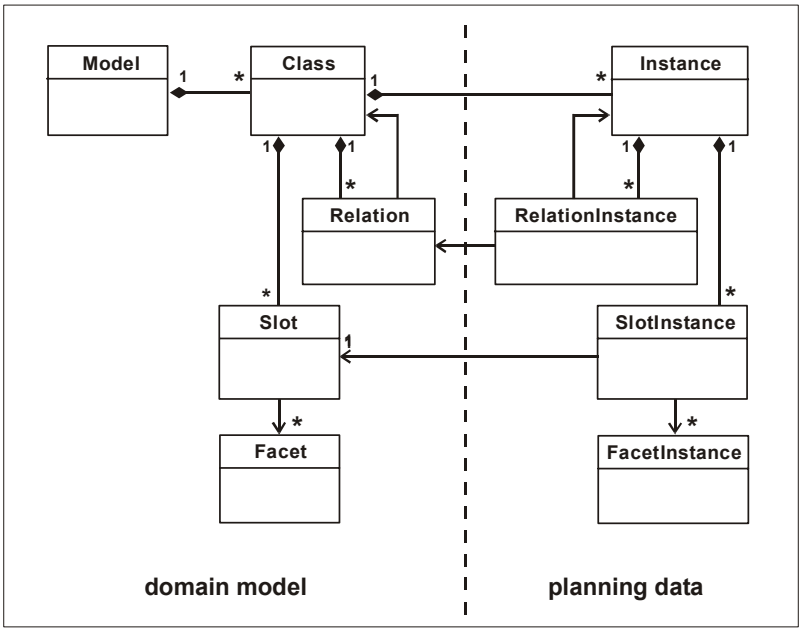

Figure 3: The AKO meta-model

2.1 System architecture

The synchronous and asynchronous collaboration of revivification planning processes has to be carried out on the technical foundation of wide area networks. This implies high robustness demands on the system components regarding communication delays and communication media failures. The usability of the design support system must not be adversely affected by this kind of problems. However, the system architecture is based on the assumption, that all persons with the same model view perform their planning activities non-distributed. Our approach to the system architecture of the design support environment consists of the base components central project server, domain model server and domain client The development of a dedicated system architecture is motivated by the failure of conventional multi- tier architectures to provide an proper support of planning activities in Virtual Enterprises (Hauschild 2003).

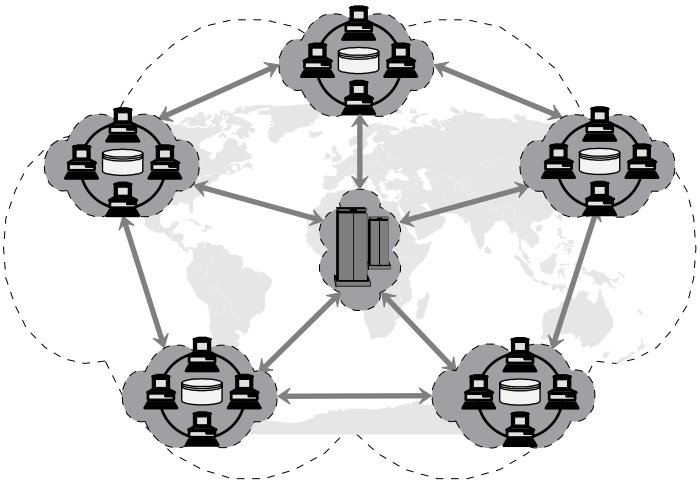

Figure 4: Distributed planning support system configuration

In this system architecture, the central object server is responsible for the management of the global project information, i.e. necessary parameters of the technical project infrastructure as well as authentication and access authorization data.

The domain model servers perform the management of the taxonomies and instances of a partial model. Therefore, its most essential component is the model management kernel, which performs this task. Furthermore, this component is responsible for measures to ensure the physical model consistency for precautions against problems caused by synchronous model access and for a local access control (Hauschild and Hübler 2000).

The domain clients, who reside on the workstations of all persons involved in the planning process and communicate with the proper domain model server, perform the Computer Supported Co-operative Work (CSCW)-based presentation and interaction tasks. Its main task however, is to act as an interface between model management and domain specific applications in order to allow the user to communicate with the shared building information by means of her or his usual applications.

In order to associate sets of building model objects to planning stages and activities objects it is feasible to assign model objects tuples of hierarchic structured description strings. These description strings can be used in modelinformation queries, which supply object sets following given search criteria. For that purpose, references to parts of the hierarchy trees are linked by Boolean expressions. That way, a connection of the model management kernels to process management support tools becomes viable.

In order to implement building model based applications for synchronous co-operation support with a responsive user interface, it is necessary to provide a mechanism for the propagation of changes of the model information. To accomplish this demand, an architecture for the announcement of building model changes based on the Observer Pattern was developed and on the technological foundation of the CORBA Typed Notification Service (The Object Management Group 2000) implemented.

3 Constraints in dynamic building models

3.1 Overview

During the planning process, many design decisions are made on the base of constraints. They arise from the numerous boundary conditions the planned building has to satisfy. In order to

support the decision making process, it is desirable to let the information system handle and evaluate these constraints.

In the scope of this paper, the term *Constraint* is used for rules, which apply to instances of classes of the domain model. They enhance the digital building model with additional semantics, and their use can help to satisfy the high requirements on information systems used in cooperative planning processes. Constraints are described by using identifiers from the domain model, like names of classes, attributes, and association roles, and are defined by the user at runtime of the information system. (Borrmann 2003)

The use of constraints in the proposed way, allows the user to successively complete the domain knowledge managed by the computer. A dynamic MMS with support for the definition and evaluation of constraints can therefore be seen as a highly customisable system, fitting perfectly to the needs of building planning processes, which are typically characterised by a high degree of uniqueness due to the individuality of buildings. Furthermore, constraints can also be used to describe fuzzy or uncertain data.

One of the main advantages of integrating constraints in digital building models is that the design rules (and therefore the domain knowledge) are not exclusively available in some specific design or engineering application, but managed by the central model management system and may thereby easily be adapted to the specific needs of a particular project and, what might even more important, exchanged between different project partners.

Basically, we distinguish two classes of constraints: To the first class belong constraints, which are used to describe domain knowledge that applies to all or almost all projects. Such constraints might be derived from legal norms, reflect state-of-the-art knowledge in this domain or discipline, or the individual expertise of a certain engineer. On the other hand, constraints can be used to describe design rules that apply to the current project only. These might be wishes of the client or domain specific requirements. Used in this way, constraints can describe the *nominal condition* of the planning. An evaluation of such constraints then shows the difference between nominal and *actual condition*. Project-specific constraints are subject to much higher dynamics than those that form part of the comparatively static domain knowledge.

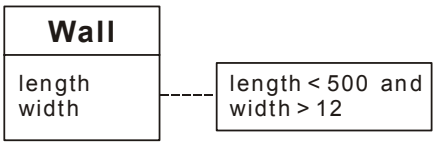

Figure 5: A constraint restricting the possible values of two attributes

In the simplest case, a constraint just restricts the range of value of an attribute, as shown in Figure 5. But constraints can as well describe dependencies between different attributes of one (Figure 6) or of different classes (Figure 7). That permits the existence of redundant data in the information base: Such constraints allow domain models to be well suited to the needs of the planers and their applications, because the information system is able to manage the logical consistency and, to a certain degree, the coherence of the data. Furthermore, the constraint-based modelling of dependencies between attributes forms a suitable base for the implementation of generic algorithms for the versioning of planning data. The most notable advantage over conventional approaches is to have the user define those dependencies dynamically, and that the dependencies are not one- but multidirectional.

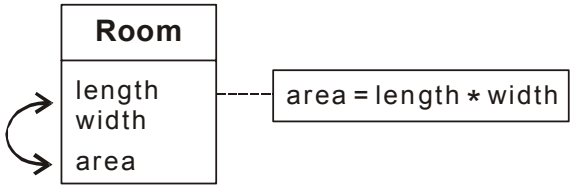

Figure 6: A constraint modelling dependencies between different attributes of one class

In order to provide the necessary flexibility during planning, the information system has to tolerate temporary inconsistencies between the planning data and the formulated constraints. But it allows the user to detect these inconsistencies, keep track of them, and eliminate them at the most appropriate time. By using a constraint-enabled model management system in this way, severe planning errors can be prevented, and the usually high costs associated with them can be avoided.

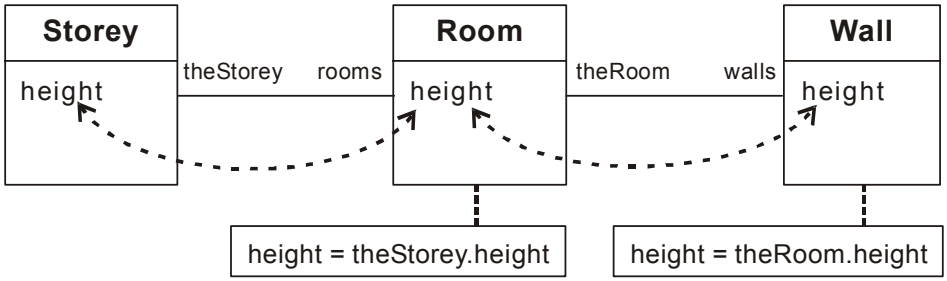

Figure 7: Constraints modelling dependencies between attributes of different classes

3.2 The constraint language CML

In order to allow the user to express constraints, an intuitively usable formal language, called CML (Constraint Modelling Language) was developed. CML is a dialect of the OCL (Object Constraint Language) which forms part of the widely used industry standard UML (Unified Modelling Language). While OCL is used for the formal specification of software systems, CML was tailored to the specific needs of defining consistency and design conditions in dynamic building models. (The Object Management Group 2003, Warmer and Kleppe 1999)

The syntax of CML is identically to that of OCL, but the language extent has been reduced. The language constructs of OCL that are used to define pre- or post-conditions could be neglected, because descriptive building models do not provide the capability to model object behaviour, like operations or methods. So, in terms of the OCL specification, all OCL constraints are invariants.

The CML allows read access to attributes and their facets, navigation along associations, the use of arithmetic, logic and comparison operators and if-then-else branching. The navigation along an association with a cardinality bigger than one results in a collection of objects. There is a set of predefined operations on collections, which iterate over all members of the collection. For instance, the operation *forAll()* tests, whether a specified condition is satisfied by all elements (Figure 8).

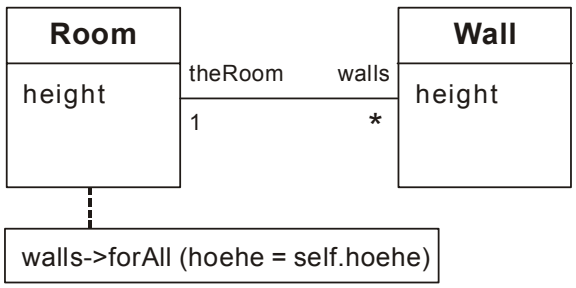

Figure 8: The use of collection operators in CML

For CML a checker and an interpreter were developed. Whereas the checker validates a certain CML expression when it is assigned, i.e. if the used names of roles and attributes exist in the domain model and if the types are compatible, the interpreter evaluates it for the present data of an instance.

3.3 Management of constraints by the MMS

In order to integrate the constraint concept in building models the AKO meta-model (Figure 3) was extended by the class *Constraint* (Figure 9). An object of type Constraint has an unique name and encapsulates a CML expression. It may be either associated with a single instance, or with a class; where in the latter case it applies to all instances of that class. In analogy to the distinction between the two types of constraints given above, it can be stated that project-independent constraints will usually be associated with classes, whereas constraints that represent project-specific design knowledge will mostly be associated with single instances.

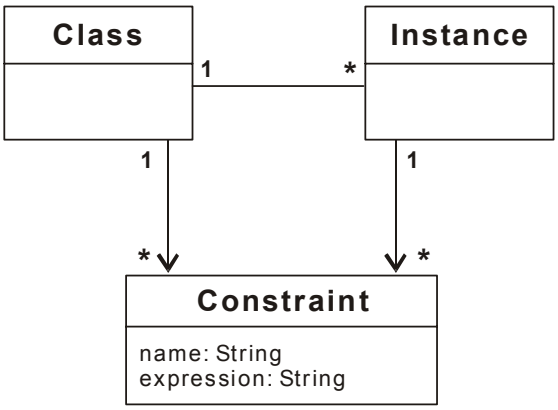

Figure 9: The use of collection operators in CML

Constraint objects are managed by the MMS together with the domain model and its instances. The CML checker and the CML interpreter are also located in the address space of the model management server. Thereby, the overhead of remote procedure calls can be avoided and a high performance is achieved, as needed for the fast evaluation of big instance populations.

When assigning a CML expression to a constraint object it will be syntactically and semantically checked. If the check is successfully passed, the syntax tree built up by parsing the expression will be stored together with the constraint object. So, for interpreting the expression at later times no second parsing is necessary, which again ensures a high performance for the evaluation of constraints for a high number of instances.

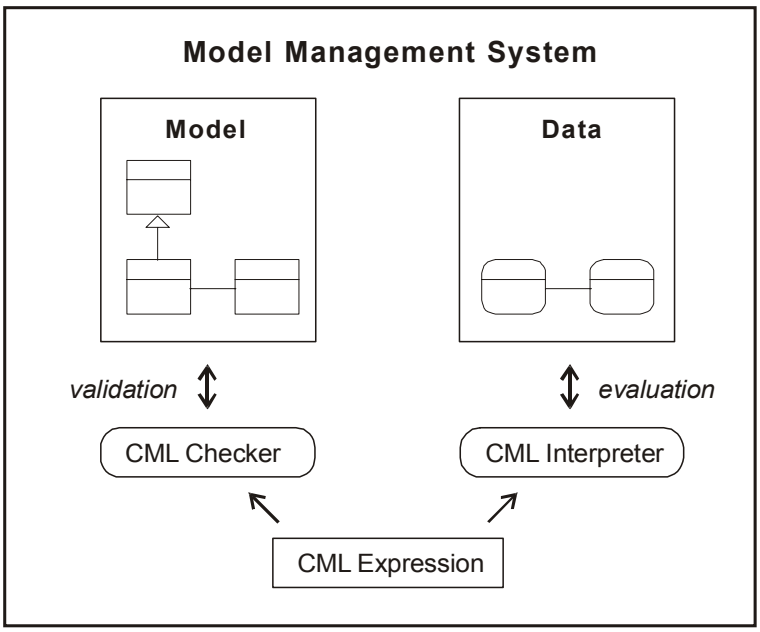

Figure 10: Work of checker and interpreter

3.4 Evaluation process

The evaluation of CML expression for a certain instance is done by the CML interpreter. It substitutes the identifiers of attributes in the expression with the corresponding data of that instance. The evaluation of constraints is performed at user-defined times and will be invoked from a MMS client. It can be performed for single instances, or along aggregations for complete instance populations.

Constraint evaluations possess particular significance when merging partitioned data sets after their separated editing by different users in offline mode, and when creating a revision of the planning data. The result of a constraint evaluation is always a Boolean value, which indicates whether the constraint is fulfilled or broken. If the system determines a broken constraint, it gives detailed information about the affected instances. This information provides a base to identify the cause and manually resolve the inconsistencies by the user.

4 Future Work

Recently, inconsistencies can be detected by the information system, but have to be resolved manually. Mainly, if the evolved constraints span a complex net of dependencies, this might be difficult task. In that case, the design application should assist the user. In order to provide such a functionality, techniques from Artificial Intelligence like “Constrained-based Reasoning” and its specialisation “Spatial Reasoning ” can be applied. (Güsgen 1992, O'Sullivan 2002)

By doing so, the system can also detect conflicts between certain constraints, i.e. it can find out if there is a solution to the constraint problem at all. The syntax trees already managed by the system provide a suitable base for the application of propagation algorithms.

5 Conclusion

In this paper, the integration of constraints in digital building models was shown. The approach described here uses a formal language for expressing constraint conditions. The developed interpreter can be used to evaluate constraint expressions for instances of domain model classes. This functionality can be used as a powerful tool to insure the consistency of planning data.

6 Acknowledgements

This work is supported by the ‚Deutsche Forschungsgemeinschaft' (German Research Foundation) within the scope of the DFG Collaborative Research Centre 524 "Materials and Constructions for Building Revivification".

7 References

Borrmann, A. (2003); Constraint-basierte Konsistenzprüfungen in dynamischen Bauwerks-modellierumgebungen. Diploma thesis. Bauhaus-University. Weimar.

Hauschild, T. and Hübler, R. (2000); Aspekte der verteilten Bauwerksmodellierung in kooperativen Entwurfsumgebungen auf Basis dynamischer Objektstrukturen. Proceedings of the Internationale Konferenz für Anwendungen der Informatik und Mathematik in den Ingenieurwissenschaften IKM 2000. Weimar.

Hauschild, T. and Hübler, R. (2002); Distributed, Collaborative Management of Building Models for Revivication Projects. IX. International Conference on Computing in Civil and Building Engineering (ICCCBE). Taipeh.

Hauschild, T. (2003); Computer Supported Cooperative Work- Applikationen in der Bauwerks-planung auf Basis einer integrierten Bauwerksmodellverwaltung. PhD-thesis. Bauhaus-University. Weimar.

Güsgen, H. W. (1992); A perspective of constraint-based reasoning. Springer Verlag. Berlin.

The Object Management Group (2000); Notification Service Specification, Version 1.0, OMG Document 00-06-20, Framingham MA.

The Object Management Group (2003); The Unified Modeling Language Specification, Version 1.5. Chapter 6: The Object Constraint Language Specification. OMG Document 03-03-13. Framingham, MA.

O'Sullivan, B. (2002); Constraint aided conceptual design. Professional Engineering Publications. London.

Ranglack, D., Kolbe, P. and Steinmann, F.(1997); Eine Schnittstelle für dynamische Objektstrukturen für Entwurfsanwendungen. Proceedings of the Internationale Konferenz für Anwendungen der Informatik und Mathematik in den Ingenieurwissenschaften IKM 1997. Weimar.

Steinmann, F. (1997); Modellbildung und computergestütztes Modellieren in frühen Phasen des architektonischen Entwurfes. PhD-thesis. Bauhaus-University. Weimar.

Warmer, J. B. and Kleppe, A. G. (1999); The object constraint language: precise modelling with UML. Addison Wesley Longman. Reading, MA.