Intelligent link-management for the support of integration in building life cycle

H. Willenbacher, R. Hübler; Bauhaus University Weimar, 99421, Germany
([heiko.willenbacher | reinhard.hübler]@informatik.uni-weimar.de)

Summary

The processes in the life cycle of buildings are characterised by highly distinct teamwork. The integration of all the distributed working participants, by providing an environment, which especially supports the communication and collaboration between the actors, is a fundamental step to improve the efficiency of the involved processes and to reduce the total costs.

In this article, a link based modelling approach and its "intelligent" link management is introduced (1). This approach realises an integration environment based on a special building model that acts as a decision support system. The link-based modelling is characterised by the definition and specialisation of links between partial models. These intelligent managed links enable a very flexible and task specific data access and exchange between all the different views and partial models of the participants.

1 Motivation

The processes in the life cycle of buildings are characterised by an increasing teamwork between a large number of different actors and their applications. The integration of all the distributed working participants is a necessary step to improve the efficiency of the involved processes and to reduce the total costs. Integration especially means the realisation of an efficient and error-free data exchange between the different applications. Furthermore, integration has to provide an environment, which enables computer aided communication and collaboration between the participants.

Both aspects are topics of current research work at diverse institutions but there is no satisfactory and complete solution particularly regarding the entire building life cycle (Willenbacher 2002).

The approach presented in this article is focussed on the data exchange and the notification of data changes. This link based modelling approach and its link management realise an integration environment based on a special kind of building model to support and to improve the data access and exchange between all the special data models of the participants.

2 Basics

The participants in the building life cycle have to solve different tasks and problems. Therefore, they need a lot of information (by interpreting data). Most of these data is created in preliminary processes. So many additional information about the data (like syntax and semantic of data, how it is structured and formatted, where are the data and how to get it, who is the owner, …) is necessary for the correct access and use of these data (Eastman 1999).

It is generally accepted, that a building model (a description of a building) is an adequate method to support collaboration and communication (Eastman 1999, Hauschild 2003, Willenbacher 2002). It represents and delivers the entirety of data required during the building lifecycle. A complete and commonly accepted and used building model answers most of the questions mentioned above. The participants "speak one language", described by the building model and thus the data exchange is no problem in principle.

2.1 Approaches for complexity reduction

Thereby the problem in fact is the definition of the building model that means the description of the required data for the entire building lifecycle. According to the common view, it's impossible to specify a global building model by reasons of complexity (Junge and Liebich 1997). Therefore, methods are necessary that reduce the complexity.

2.1.1 Object oriented modelling

The complexity can be reduced by abstraction mechanisms like:

- intension/extension (classification)
- generalisation /specialisation (inheritance)
- composition / decomposition (aggregation).

The object oriented modelling paradigm (oo-paradigm) provides these mechanisms. Hence, the decision deduces to use it as the conceptual basis for our building model (figure 1).

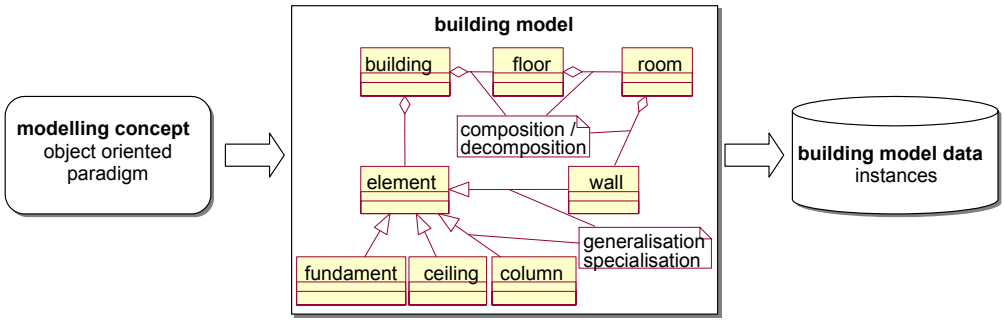

figure 1: object oriented modelling approach

2.1.2 Partial models

The second step to reduce complexity is characterised by the decomposition of the whole building model in partial models. These partial models are defined as follows in the context of this paper: *A partial model specifies a domain specific closed set of object descriptions (classes) and the relations between them as a subset of the building model.*

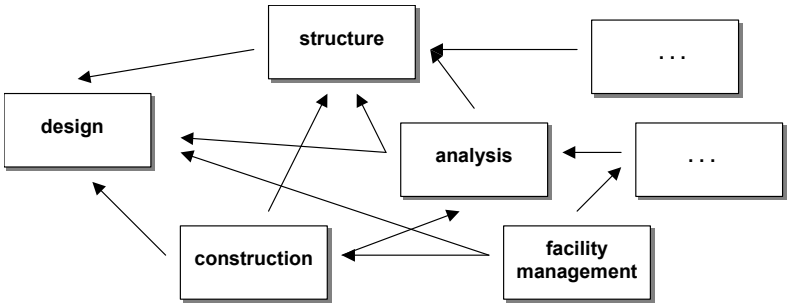

figure 2: relations between partial models reflect information needs

The partial models as a whole establish the building model. However, this pure union of partial models does not reflect the entirety of data in the sense of a "virtual building". Therefore, the existing relations between the data of the partial models have to be modelled too. Thus, the building model is made up of partial models and the relation structure, which is defined explicitly between the partial models (figure 2).

2.1.3 Dynamic of building models

It is to the specific of buildings and the multitude of diverse participants and their different views to the entirety of data that a total prediction of occurring data requirements is impossible,

in spite of the mentioned mechanisms for complexity. Hence, the de facto existing dynamic of the planning processes and of the building life cycle has to be considered, within the scope of data modelling and management.

Referring to this, two aspects of dynamic can be identified. The first conclusion is related to the set of involved partial models. This set is not determined at the beginning of a project and varies in project lifetime.

The second and more significant aspect is related to the content and structure of the partial models directly. In this context dynamic means the possibility of further development and modifiability of the partial models (Steinmann 1997). According to this, the following features have to be provided: (figure 3):

- specialisations and decompositions of classes
- hierarchical reorganisations within the models
- renaming of classes and attributes
- creation and deletion of classes and attributes.

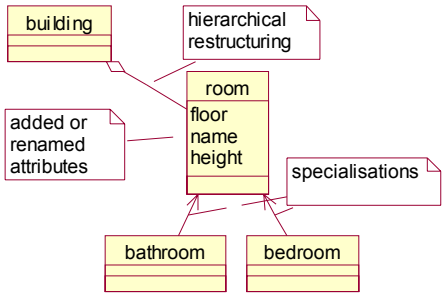

figure 3: examples of dynamic modifications

The outcome of these assumptions is the following definition of a building model: *The digital building model consists of a dynamic varying set of currently involved dynamic modifiable partial models including the existing dynamic adaptable relation structure between these models*.

The first mentioned aspect of the varying number of involved partial models has to be reflected by the dynamic definition and management of relations (afterwards called “links”) between the partial models. The dynamic model development and adaptation is realised by the inclusion and management of a conceptual modelling level. This level of conceptual modelling is handled by a so-called dynamic object oriented model management systems (MMS)(Hauschild 2003, Willenbacher 2002). The MMS provide an API (application programming interface) for manipulating and analysing the structure and the data of partial models (domain models).

2.2 Resulting requirements

The introduced building model approach is a very good basis for descriptive model oriented integration in building life cycle. But there are a some problems resulting especially from the dynamic of the partial models like the followings:

- no standardisation (standardised partial models are wanted if existing, but not claimed)
    - mostly no or only little knowledge about partial models
    - difficulties to understand and to analyse partial models
- models (structures, classes, names) change over time
    - difficulties to provide algorithms
    - generic data access and presentation (MMS – API)
    - links between partial model elements are probably not up to date

3 Solutions

Comprising the preceding section can be stated that the mentioned modelling approach is very flexible and every participant can model his domain data according to his requirements, but there are some problems in using the data from other domains. The link based modelling approach and the link management system are developed to accomplish these problems in data exchange by providing:

- support for the definition and specification of links between partial models,
- methods and tools to facilitate the analysis of models and links,
- an automatic change notification,
- and thus, a maximum of flexibility and visibility of the data exchange.

3.1 Link based modelling approach

The handling of the coherence of the partial models and thus the support of the data exchange between the participants and their specific models represent the main goal of this approach. The coherence of the partial models results from process and domain spanning character of the data and information, which is necessary for decision support. Links are specified following the data requirements between the partial models. According to this, these links establish a dynamic modifiable central co-ordination and navigation level.

Links relate source entities from partial models with target entities from other partial models (see example in figure 4).

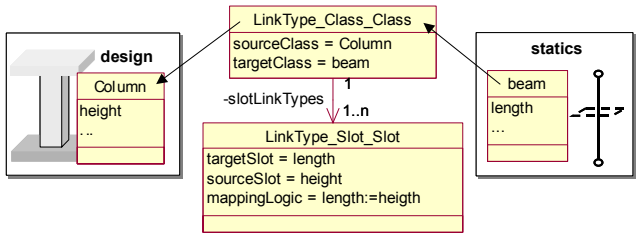

figure 4: example of a link-type between entities

The definition of link-types and the specification of links are based on a framework, which follows the oo-paradigm too. Thereby link-types define the relations between partial models on the class level while the links specify these link-types on the instance level (figure 5).

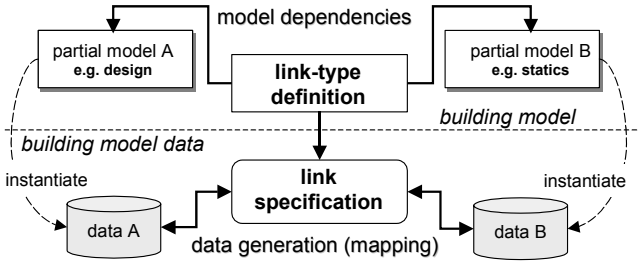

figure 5 : link-type definition and link specification

3.1.1 Link-types and links

The tasks and goals of the links can be subdivided in three categories. The links of the first category, the generating links, are used for direct manipulation of data in partial models. The

second category contains informing links, which represent informing relations in the sense of notifications and change management. The propagating links form the third category.

Generating links

The generating links solely support the data exchange (access) between partial models. Generating links can be executed to set data in a target model according to a mapping logic specified in the link-type (figure 4).

Creating this kind of links is a two-step process. First, a link-type has to be defined according to a matching predefined template provided by the framework (figure 6). While defining a link-type it's stated, what and how many classes (link-types between classes) or slots (attributes – link-types between slots) are in relation and what is the mapping logic.

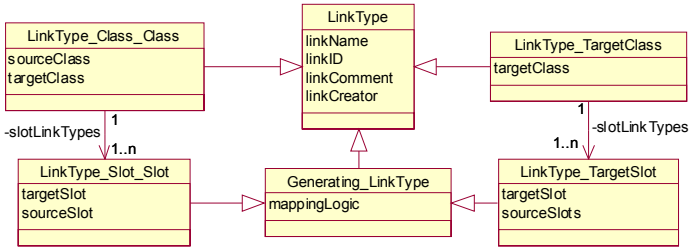

figure 6: section of the link-types class framework

The second step is to specify a link according to the defined link-type. This specification is realised by instantiation of link-types and the adequate assignment of concrete data from the

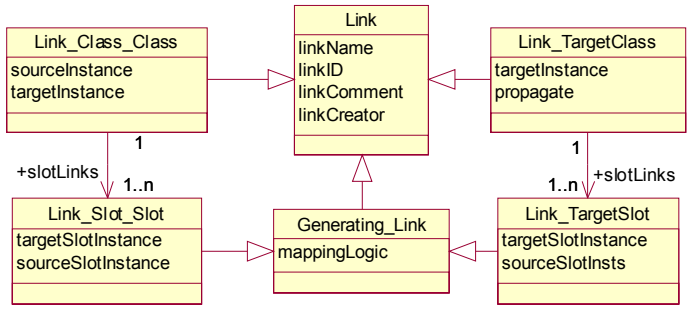

figure 7: section of the generating links framework

partial models. Hereby the correlated entities are instances and slot instances (values). So it's possible to automatically create links for different instances, which all have the same mapping logic.

Informing links

The informing links are made to inform interested participants about changes of elements in partial models. In this regard, the following events, which partly result from the use of the dynamic building modelling approach, have to be covered through special link-types:

- existence (registration, deregistration) of partial models in the total system
- existence (creation, deletion) of classes, instances and attributes in partial models
- changes on values of attributes.

It's not necessary to define a link-type to specify an informing link. The required link-types are predefined by the framework (figure 8).

The informing links are characterized by special properties. An informing link relates one source entity, which is observed (s. section 3.2) and a number of target entities, which are informed when the source is changed. A source or target entity can be of any entity type that is supported by the model management system (MMS). These types are schema, package, class, instance, relation, relation instance, slot and slot instance.

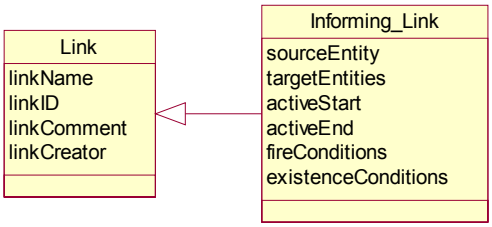

figure 8: informing link

The other properties of an informing link are used to minimize unwanted information about changes on entities. *Fire conditions* determine when a change of an entity is relevant, e.g. if a specified maximum value is exceeded. *Existence conditions* as well as *active start* and *active end* affect the activity of a link. If it's inactive, no change events are triggered.

Propagating links

The purpose of propagating links is characterised by the direct transfer of values between already existing model elements. This transfer is only executed if a relevant change on a source entity occurs. The propagating links are automatically created based on generating links, if

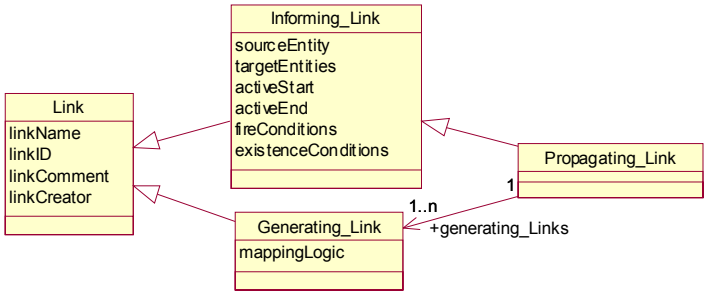

figure 9: properties of a propagating link

desired. Thereby the source elements (instance or slot instance) of the generating link become observed source entities of the propagating link. Furthermore, the propagating link has references to generating links that are executed in case of propagation (figure 9).

3.2 Link management system

The link management system provides the infrastructure for managing the links between the partial models.

3.2.1 Distributed system architecture

The model architecture that has to be supported consists of a set of partial models. A hybrid system architecture is qualified for managing this compound of partial models (Willenbacher and Hübler 2000). Thus, the partial models are distributed and separate managed models. There is a need for a central co-ordinating layer that supports the communication and co-operation between the distributed partial models.

The data exchange between the partial models is realised by specified links according to the suggested link based approach. The management and processing of these links are characterised by following criteria:

- spatially separated
- temporal variant
- structural differences in the concepts and languages
- different access and management functionality.

The agent technology and especially multi-agent systems are predestined to support such systems (Weiss 1999, Willenbacher and Hübler 2001). These criteria resulting from the processual character of integrated planning and the dynamic modelling approach set the initial point for an agent-based link management. The reason for this follows from the specific properties of software agents (Jennings and Wooldridge 1998) like:

- autonomy
- pro-activeness / reactivity
- social ability
- mobility

Thereby agents are qualified to support the work of every individual participant (autonomy) as well as the necessary communication and co-operation (social ability).

The suggested system architecture consists of distributed partial models based on a multi-agent system. Thus, every MMS (s. section 2.1.3) is wrapped by different types of agents that realise the co-operation and communication between the partial models. The "local" agents at the partial models are co-ordinated by some other "central" agents, which form the "central link management" system (figure 10).

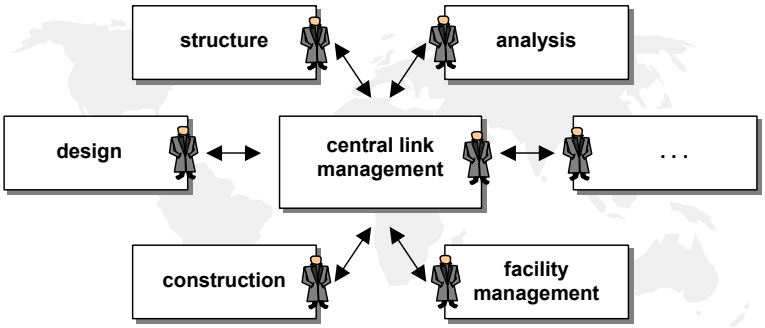

figure 10: agent system architecture

The agents communicate via sending and receiving messages. The agents have to use the same language within the messages to understand each other. Therefore, an ontology is used to ensure this common language. It defines the needed concepts in this context. The concepts are mainly the used link-types, links and entities (classes, packages, instances, slots, slot instances, etc.).

3.2.2 Link management

As it was mentioned (in section 2.2), it's very difficult for the participants to create link-types and links because of missing knowledge about the dynamic partial models. Therefore, the link management system provides efficient methods and tools for creating link-types and links. Especially for creating link-types for generating links, a rule-based mechanism generates

suggestions how a source class should be mapped to a target class. Thereby, the slots and relations of the source class are analysed by the rule-based system. Rules, which can be defined or extended by the participants, control this analysis. In this way, the rule-based system generates link-types between the source and the target class and their slots (Willenbacher and Hübler and Koch 2003). After this, these link-types can be used to create links, which relate instances and their slot values.

The link-types and the links are created in a very dynamical manner, too. That implies the problem of specifying algorithms for link-types. It's often necessary to define a special mapping logic for the value of a target slot in the form of an algorithm. Regarding the dynamic of the source and target models, it's not always possible to define these algorithms at the beginning of the building life cycle. So there has to be the possibility to define these algorithms during runtime. Therefore, the link based management system uses an interpreting language system that provides the facility to define and to execute algorithms at runtime.

Furthermore, the dynamic of the models and the links requires a very flexible way to store and to access the link-types and links. The above mentioned model management system is predestined to store these data, too. So there exists another partial model reflecting the link-types, links and the related entities (cp. figure 6-9). The structure of this model is identical to the ontology used for agent communication. This results in a very efficient approach, because the same link-based possibilities can be used to update and to inform the links themselves. So if a source entity is deleted the corresponding link-type and or link can be deleted, too if desired. In addition this way of saving link-types and links offers the possibility for the users to edit these data as they would edit there own data in there own partial models, no other browsers or edit tools are necessary.

3.3 Prototypical implementation

The current version of the prototype of the link management system offers a lot of tools and dialogs to support link creation and execution. Only some special aspects are illustrated in this paper.

3.3.1 Link-type definition

The link-based management provides a way to define and to execute algorithms during runtime.

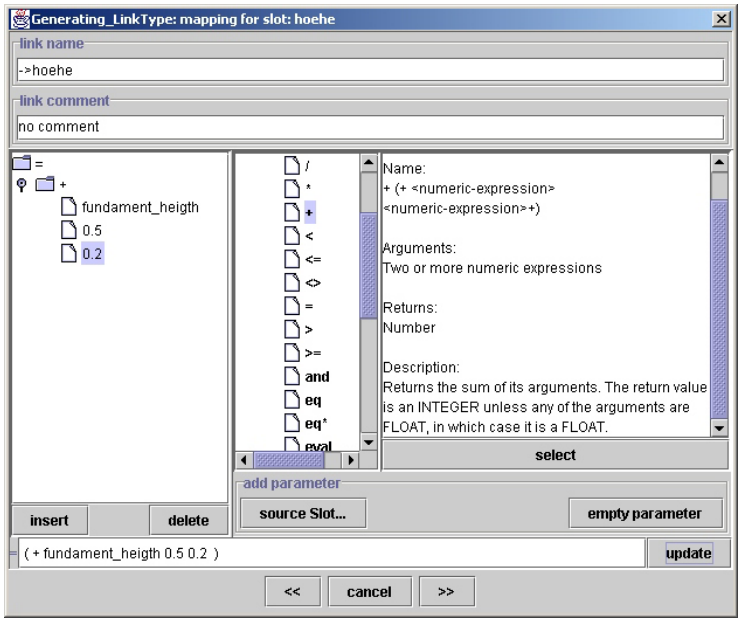

figure 11: dialog for mapping logic definition

The JESS (2) interpreting language is used to realise this feature. Participants in the building life cycle normally will not be able to write such lisp-like JESS code to define the mapping logic. A graphical programming tool is provided by the link management system to make this task easier (figure 11). With the help of this tool, it's possible to create the mapping logic by selecting terms from a list of possible functions and programming language expressions. There is an explanation of every function, too. The syntax of the JESS language is very easy. Together with the provided functions, every participant should be able to define the needed algorithms. In addition, this tool allows the participant to access slots from any class in any partial model. Therefore, it's possible to calculate a target slot value (when an according link is executed) from many different source slot values from very different classes (instances) and partial models.

3.3.2 Informing link specification

The task of an informing link is to inform about a change of an observed entity. However, not every change at any point in time is interesting for the users. Therefore, while defining an informing link some link attributes can be set to reduce the triggering of unwanted events. Besides the temporal activation through the *active start* and *active end* attributes the most important one in this context is the *fire conditions* attribute. The *fire conditions* can be defined with a special dialog (figure 12). These conditions are later used when the entity is observed. During observation, a change event is only triggered if the specified conditions are complied. The test whether the conditions are complied or not is realised by the JESS system, too. Therefore, the conditions are transformed to rules, which the rules engine of the JESS system is running against. The change event is triggered if the rule processing delivers true.

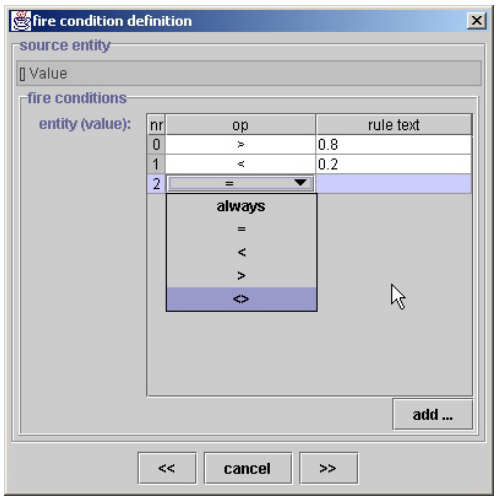

figure 12: fire condtions definition

3.3.3 Link specification and execution

Generating links are created by instantiating a generating link-type. This task is supported by some other dialogs that allow the user to create many links according to a selected link-type at once. Therefore, all instances of the source class of the link-type are listed. The user now ticks all the instances for that a link should be specified (figure 13). Thereby the user states whether a propagating link should be created, too.

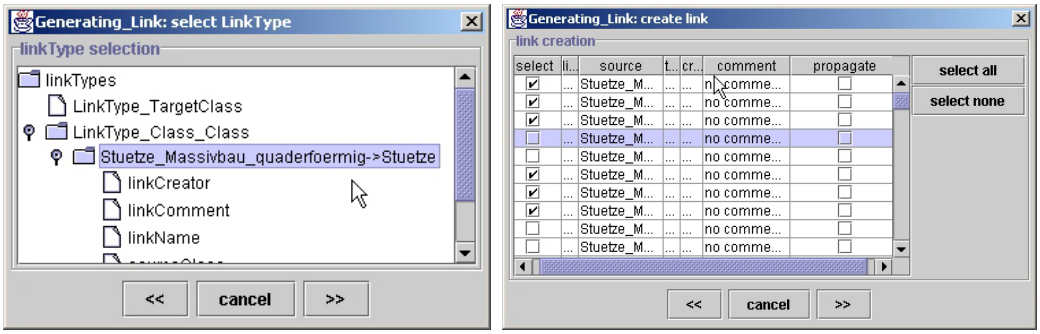

figure 13: link selection and creation

Link execution is organised in the same way. The user selects all the links that should be executed. Propagating links are created when the generating links are executed. This order is necessary because a propagating link can only exist between already existing entities.

Informing and propagating links are not executed by the user directly. These links are performed when relevant changes on their source entities occur.

3.3.4 Agent-based link management and entity observation

The agent based link management and observation is responsible for recognition of these changes. One partial model is observed by a local domain observer agent. This agent observes relevant entities of the partial model. Source entities of informing or propagating links are relevant entities. The central link management agent sends links to the local domain agent, which then registers observers on the source entities of these links. The observers are realised by CORBA-notification services and part of the MMS implementation (Hauschild 2003, Hauschild and Hübler 2003). The local domain observer agent recognise all changes on observed entities. Based on the fire conditions the agent has to decide whether a relevant event occurred and which links are affected. In case of informing links, the target entities have to be notified. Thereby messages are sent to user or domain agents of other domains. Affected propagating links are executed and the target values are set.

The agent system is realised by using the JADE framework (3). The ontology is defined by PROTÉGÉ (4). The bean generator plug-in (5) for PROTÉGÉ is used to transform the PROTÉGÉ structure into an ontology suitable for JADE.

4 Conclusions

The link based modelling approach in combination with the dynamical object oriented model management represents an adequate approach to support the data and information exchange between the different participants and their applications especially under consideration of the dynamic aspects of the building life cycle.

The acceptance of the approach depends on the convenience of link creation and link management. Therefore, some further research work is necessary especially in the field of more intelligent link specification support and user interfaces for efficient object selection. The realisation of a navigation layer that supports the search for and the selection of objects is the next important step in this context.

The agent technology is very suitable for the management of this dynamic and distributed approach, which results from the specific properties of agents and multi-agent systems. The social ability with the massage-oriented communication has to be emphasised especially in this context, because this is the basis for a very flexible behaviour of the total system.

The agent-based link management and the rule-based link processing in an interpreting manner realise the needed functionality to react on dynamic information requirements in building life cycle. Furthermore, the approach offers the opportunity for participants to put as much knowledge and intelligence as possible into the data exchange, hence it fits to their special needs.

There are lot of difficult problems within this approach but up to now it possibly seems to be the only way to manage the complexity of a building model and it’s a good starting point for further investigations.

5 Endnotes

(1) This project is part of the collaborative research center 524 “materials and structures in revitalisation of buildings”, which is sponsored by Deutsche Forschungsgemeinschaft.

(2) JAVA Expert System Shell http://herzberg.ca.sandia.gov/jess/ (2004-04-15)

(3) Java Agent Development Framework http://sharon.cselt.it/projects/jade/ (2004-04-15)

(4) PROTÉGÉ http://protege.stanford.edu/index.html (2004-04-15)

(5) bean generator http://gaper.swi.psy.uva.nl/beangenerator/content/main.php (2004-04-15)

6 References

Eastman, C. M. (1999). Building Product Models: Computer Environments Supporting Design and Construction. CRC Press LLC

Hauschild, Th. (2003). CSCW Applikationen in der Bauwerksplanung auf Basis einer integrierten Bauwerksmodellverwaltung, doctoral thesis, bauhaus-university weimar

Hauschild, Th. and Hübler, R. (2003).Techniken der Verwaltung dynamischer digitaler Bauwerksmodelle für Revitalisierungsvorhaben IKM 2003, Weimar

Jennings, N. R. and Wooldridge, M. J. (1998). Agent Technology Foundations, Applications, and Markets. In: UNICOM Seminars Ltd/Springer-Verlag Berlin Heidelberg

Junge, R. and Liebich, T. (1997). Product Data Model for Interoperability in an distributed Environment. In: Junge, R. (ed.): Proceedings of the 7th International Conference on Computer Aided Architectural Design Futures. Kluwer Academic Publishers

Steinmann, F. (1997). Modellbildung und computergestütztes Modellieren in frühen Phasen des architektonischen Entwurfes, doctoral thesis, bauhaus-university weimar

Weiss, G. (1999) (ed.). Multiagent Systems A Modern Approach to Distributed Artificial Intelligence. In: The MIT Press Cambridge, Massachusetts London, England

Willenbacher, H. (2002). Interaktive verknüpfungsbasierte Bauwerksmodellierung als Integrationsplattform für den Bauwerkslebenszyklus, doctoral thesis, bauhaus-university weimar

Willenbacher, H. and Hübler, R. (2000) Relationen zwischen Domänenmodellen - Ansatz zur Schaffung einer integrierenden computergestützten Bauplanungsumgebung. IKM 2000, Weimar

Willenbacher, H. and Hübler R. (2001) Agentenbasierte Bauwerksmodellverwaltung In: Romberg, R. and Schulz, M. (ed.). Forum Bauinformatik 2001; VDI Verlag GmbH, München

Willenbacher, H. and Hübler, R. and Koch, C. (2003) Intelligentes Verknüpfungsmanagement am Beispiel einer 3D_Visualisierung. IKM 2003, Weimar