Available online at www.sciencedirect.com

ScienceDirect

Computers in Industry 57 (2006) 809–816

COMPUTERS IN INDUSTRY

www.elsevier.com/locate/compind

An information model for collaboration in the construction Industry

J.P. van Leeuwen [*], S. Fridqvist

Eindhoven University of Technology, Department of Architecture, Building, and Planning, Design Systems Group, P.O. Box 513-VRT 9.11, 5600 MB Eindhoven, The Netherlands

Accepted 27 April 2006
Available online 23 June 2006

Abstract

Collaborative work is an essential ingredient for success in the construction industry. With the advancements of capabilities of information technologies and communication infrastructures, the effective utilisation of these technologies has become very important and strongly affects business processes that have long followed traditional paths. This article describes the main characteristics of the concept-modelling framework that is developed in the DesKs project. Concept modelling gives end-users access to the schema of design models and provides a high level of flexibility for modelling. To support collaborative work, it provides remote data access and allows users to share resources that, instead of being exchanged or stored centrally, remain active at their source in tight relation with business processes. The main technical aspects of the concept-modelling framework are discussed. Object version control and timeline management of revisions of objects are used to increase the integrity between objects that are accessed and edited by multiple users across a network.

Keywords: Collaborative design; Concept modelling; Object version management

1. Collaborative design

Construction projects typically are projects in which a large number of participants have to work together on the design and production of a complex product that is one-of-a-kind. Many of these participants do not work together on a regular basis; teams in construction projects are often organised on a project-basis. Yet, collaboration in the design process of such projects is generally regarded to be the critical factor of success. Collaborative design is a term that denotes more than just co-operation. In co-operation, participants work together to achieve mutual benefits but do not necessarily have a common goal. They will retain their own resources, sharing only the minimum required for the co-operation. In collaboration, however, the participants are committed to a common mission and are willing to share the knowledge that is necessary to fulfil that mission [1,2].

1.1. State-of-the-art in CSCW

Current practice of computer support for collaborative work (CSCW) in the construction industry mainly utilises tools such as centralised project databases, systems for workflow management (WFM) [3–5], and electronic document management (EDM) applied in local or wide area network environments. Although beneficial to the industry, this kind of support has important limitations. Centralisation of project data aims to bring together all data that concerns a project. However, the boundary between project-related data and project-independent data is not clearly defined. Hence, centralised databases are never complete. More importantly, centralised project data becomes isolated from business processes that are not centralised.

Tools for workflow and document management are generally based on documents as organising entities. Although documents may be a good means for human beings to communicate, they are not a logical means to organise and store information. Consistency of information is often compromised when multiple documents describe different aspects (e.g. shape and material) of one artefact, or when one aspect (e.g. a particular measure) is redundantly repeated in several documents.

[*] Corresponding author. Tel.: +31 402472288; fax: +31 402450328.
E-mail address: J.P.v.Leeuwen@bwk.tue.nl (J.P. van Leeuwen).
URL: http://www.ds.arch.tue.nl

0166-3615/$ – see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.compind.2006.04.011

Research and development of product modelling technology involves the implementation of object-oriented approaches for the description of products throughout their life cycle [6,7]. The general methodology applied in product model development is to predefine schemata of object classes that represent the common ground for a particular domain. International standards of schemata are being defined for many disciplines, including various domains within the building and construction industry [8–12]. Using the schemata, designs can be described by populating object models with properties and relationships that are defined in the object classes. Communication takes place either by exchanging these models as documents or by placing them in centralised databases.

1.2. Identified problems

The notion of standardising object classes for modelling designs is currently based on the assumption that a satisfying classification of high-level objects can be agreed upon by all actors within the construction industry. Product modelling developments for this industry manifest the following problems in relation with collaborative design:

1. *Inadequate standards*: Object classes are generally targeted at production stages. This renders the schemata unacceptable for early design stages, because the concepts used in early stages differ from those in later stages. Using the more final concepts in early design would imply or enforce many decisions in the early stages that designers do not want to take at that point. Neither are the production-centred classes particularly suitable for the maintenance phase.
2. *Inaccessible schemata*: The schemata are predefined and not accessible for changes by end-users. Again, this makes the schemata hard to use in the early stages of design, when standard concepts may not fit the particularities of the design, or state of the design, that designers need to express.
3. *Inflexible standards*: Standardisation concentrates on the definition of classes for real-world objects with all their properties and interrelationships. Typically, the schemata contain classes for different kinds of walls, floor slabs, windows, doors, heating components, and so forth. This enforces a classification of products, which does not necessarily serve the needs of the supply chain, for example, when new products are developed or when multiple functions are combined into single products.
4. *Monolithic standards*: To support communication, product modelling schemas need to be standardised. Unfortunately, the efforts to create standards such as the IFCs [39] seem to result in large schemas intended to cover all needs for a branch of industry. Such large schemas are hard to accomplish to begin with, and will become hard to maintain as technology and practices change.
5. *Exchange rather than sharing*: Exchanging documents or using centralised project databases for the communication separates data from its source and isolates it from the business processes. This leads to redundancy and potentially to inconsistency and outdated data. In any case, it does not contribute satisfactorily to a tighter integration of business processes from partners in a collaboration project.
6. *No design support integration*: The problems identified above render product modelling an ineffective technology for design support in the construction industry. As a result, many R&D efforts that aim to support specific design tasks, such as case-based reasoning, simulation, and evaluation systems, cannot make use of the richness of integrated information that could potentially be delivered by product modelling. This seriously obstructs the path for integration of design support systems with computer support for collaborative work.

2. Concept modelling

Concept modelling is a technology that provides:

- user access to the definition of schemata;
- property-oriented modelling;
- a distributed object model for sharing rather than exchanging information;
- support for a layered approach to modelling, which resolves the problems of monolithic standards.

Concept modelling is a dynamic form of product modelling that was initially described by van Leeuwen[1] [13]. Concept modelling supports designers by giving them access to the schema, the conceptual level of the product model. This allows designers to describe design concepts in a formal manner by defining extensions to the schema. Such design concepts may represent tangible objects as well as more abstract notions such as functions or properties. Concept modelling uses Concepts to build conceptual schemas (also known as ontologies) while specific models representing individual designs are built with Individuals. The meaning or interpretation of Individuals is given through reference to Concepts. In other words, the Concepts defined in a schema can be used to instantiate Individuals that represent information concerning a particular design. The internal structure of both Concepts and Individuals is created through components that define the Concept or Individual by connecting it to other Concepts or Individuals in specific ways, for example, to denote a part-of relation.

While being property-oriented, i.e. supporting modelling of properties as individual objects, the concept-modelling approach does not distinguish between objects and properties; both are defined as concepts with relationships to other concepts.

In principle, concept modelling is an object-oriented approach, but there are two important aspects that distinguish it. Firstly, relationships can be added to an Individual, disregarding the definition of the Concept from which it was instantiated, to make it represent a specific design case.

[1] In this work, concept modelling was called feature based modelling, but a change of terminology was made in 2002 to avoid confusion with form features. The terms Feature Type and Feature Instance were changed into Concept and Individual.

Secondly, the connection between an Individual and a Concept creates a strongly typed modelling environment. However, since this connection can be modified, it is also dynamic, supporting the conceptual evolution that is characteristic of design processes. Such 'change of concept' could be triggered, for example, by a search algorithm that has found a better match for the particular Individual's properties. Concept modelling is designed to provide flexibility to end-users, such that they can determine what concepts to use in modelling and how to deal with non-typical situations in the model [14].

Concept modelling supports a layered approach to schema modelling, whereby standards can be broken down into smaller entities that are comparatively easy to construct and maintain. The current monolithic approaches seek to include everything into one schema, both vertically and horizontally. The layered approach allows schemas to be broken down vertically, i.e. basic modelling concepts can be modelled apart from the higher-level concepts that build upon the basic ones. The layered approach also works horizontally, by separating relatively unrelated branch-specific parts. Thus, the layered approach allows schemas to be compartmentalized, so that parts of the schema can be developed and maintained independently of other parts. One benefit would be that basic layers, whereupon much relies, could achieve very long life-spans, while simultaneously allowing highly specific top-level layers to be changed as needed with little trouble, thanks to the small size of the involved group of users. Another benefit is that communication between schema-wise incompatible systems might still be partially successful based on a lower, common level schema. This, obviously, would support collaboration, but also alleviate some of the troubles of schema mapping.

Research in the Design Systems group at Eindhoven University of Technology has resulted in the development of a technological framework for concept modelling. The work has been implemented in the form of an application-programming interface (API) [15]. Prototype testing of the API has successfully demonstrated the following functionality:

- *Object data management for concept modeling*: The API makes available a core object model that can be used to describe both design concepts and individual designs. Data are organised using namespace functionality similar to that in XML.
- *Object-based version control and timeline management*: The API implements version control and maintains a timeline for each object (concepts and individuals). This serves multiple purposes, including improved consistency and reliable multi-user access.
- *User management and authentication*: The API is prepared for multi-user environments and provides functionality for ownership and role-assignment per object.

Current research investigates the rationale and implementation of:

- *Concept recognition*: This is a kind of pattern-matching approach that enables users to find concepts that suit a particular network of individuals. An example application of this technology is to search for products whose concept description matches the required properties specified by a designer.
- *Remote object sharing*: The core model transparently deals with remote objects in a network of systems that are based on the API. This implementation makes use of the standard HTTP and SOAP protocols.

2.1. Related research

Concept modelling has been developed as a theory and later implemented in a framework over the past several years [16,17,13,18]. It was inspired by the technology of feature modelling and how this technology can be used in conceptual design stages; examples are the work at the Design Automation Laboratory of Arizona State University [19] and at the Technical University of Delft [20–22].

Internationally, the paradigms of schema evolution and model flexibility have been recognised as essential innovations, answering to restrictions that standardisation efforts fail to address. Similar research has been conducted by Eastman at UCLA and later at Georgia Institute of Technology on evolution of schemata [23,24]; at Lund University of Technology on property-oriented modelling [25,26]; and at Deakin University on design knowledge management [27].

Parallel to this work, XML has emerged as a technology that addresses the same issues of extensibility and flexibility in modelling and communicating information [28]. Hence, it is fruitfully utilised in the concept-modelling developments. In a simplified view, the concept-modelling paradigm could be compared to an XML Schema that specifies a limited set of attributes to elements, which enables us to provide certain reasoning mechanisms that support the interpretation of the information.

3. Distributed object management

Two aspects of the current state of the implementation of the concept-modelling framework are discussed in this section. Both pertain to the management of distributed objects in a network of design and engineering information.

3.1. Object-based access control

Controlled and authenticated access to shared information resources is a prerequisite for computer supported collaborative design. This involves defining various levels of access, in order to control if users are authorised to perform the requested operations on information. In the concept-modelling framework, access-levels are used to govern reading, copying, using, referring to, and editing information. Editing is controlled by a checkout-and-commit mechanism that works on single objects. Users have to check out an object, marking it as being under revision, before they can make changes to it. Once the changes are made, the object is committed back to the source and stored as a new version or revision (see Section 3.2). The checkout mechanism can be applied to function automatically or manually in software applications based on the framework. This is related to three modes of editing that are distinguished:

- *Instantaneous editing* is required when changes made by one user should instantly be visible to other users. This mode of working is applied, for example, in virtual workspaces when users collaborate synchronously on a design and need to see and communicate about each other's modifications in real time. During such an operation, the changes to the coordinates are instantly and continuously made available to all users.
- *Intermittent editing* is sufficient when users do not need to have instant updates of modifications in synchronous collaboration sessions. The changes are made available to others only when the user has committed them.
- *Off-line editing* is relevant when network facilities are not permanently available. Objects remain checked out for a longer period and changes are committed only the next time a user is online.

The implementation of the concept-modelling framework uses remote data access and ensures that multiple accesses to an object actually address one single object.

3.2. Object-based version management

The concept-modelling approach structures and organises information on the basis of objects, rather than documents. Hence, version control is necessary on the level of objects. In the remainder of this paper 'object' is used to denote all objects for which version information is maintained: *Concepts*, *Individuals*, and *components* of Concepts as well as Individuals.

3.2.1. Why object versions?

Maintaining versions of objects representing a design is interesting for the purpose of documenting alternatives of that design. Additionally, in the context of collaborative design, version management of objects is important to maintain the consistency of an object model that is accessed by multiple users. Changes to objects will be administered through the creation of versions and revisions. This ensures that the state of objects recorded in previous versions will remain available. References between objects can make use of the version information of objects, so that the data consistency is not compromised when new versions are created. Semantic consistency is, of course, not ensured by the implementation of object version management.

In literature, version control at the object level is described by Cellary and Jomier [29], who use so-called 'stamps' to identify object versions in multi-version databases; by Bernstein [30], proposing basic operations on versions that are identified through a succeeds relationship; and by Kimber et al. [31] who describe referent tracking documents as a means to control version information through hyperlink management.

Administering versions and revisions of objects provides a means to record the changes to objects. In combination with authenticated access, it is possible to trace the changes of objects to the users who made those changes. Having a record of the history of each object also facilitates the browsing and restoring of previous states of a design model. This has potential for quality control, but also for design education and research, e.g., to present a narrative of design processes.

3.2.2. Levels of versions

Version information for objects in the concept-modelling framework is structured in three levels. In the top two levels, an integer number is used to identify versions: one for major versions and another for minor versions. Numbering starts at 1 and minor version numbering is restarted within each major version. New major versions may be initiated either by the user when he regards the changes significant enough for a new major version, or by the system when the changes are such that consistency problems are likely to arise in other places of the model. For example, a new major version is created by the system when a component is removed, because existing references to the concept may rely on the presence of the component.

The third level of version information is for revisions and time management. When an object is checked out for editing, it will remain under revision until it is submitted again as a new version. Also, new objects are initially under revision until they are submitted. Revisions are identified by their creation time. The revision information is also maintained for versions of objects, so the timestamp is available for each object version as well.

In the concept-modelling framework, either committing a revision or submitting a version concludes an editing activity. If this is done manually or automatically depends on the implementation in the application that is based on the framework. An implication is that, once committed or submitted, revisions and versions are fixed and can no longer be changed. Changes on objects will always lead to the creation of new revisions or versions. On the one hand, this helps to ensure consistency in the model. On the other hand, it calls for smart ways of referencing objects, such that up-to-date information is used when referring to an object. This is discussed further in Section 3.2.4.

3.2.3. Timeline management

Versions and revisions of objects have timestamps that designate their lifetime. Because each version of an object is also a revision, we will refer to 'revision' in this text to indicate both. Each revision of an object always has a 'valid from' timestamp, indicating the moment this revision was created. When a revision becomes outdated, either because the object was deleted or because a newer revision was created, this revision will also get a 'valid to' timestamp. This concludes the lifetime of the particular revision. Subsequent revisions together form the lifetime of an object. Normally, the 'valid from' timestamp of a revision corresponds to the 'valid to' timestamp of its predecessor. An exception is when a previously deleted object is *revived*. In that case, the timeline of revisions will show a gap. Fig. 1 shows the graphical notation that is used for the representation of timelines of objects. Blocks indicate the beginning and ending of a particular revision's lifespan; an arrowhead denotes 'no ending time' meaning that the revision is the current one.

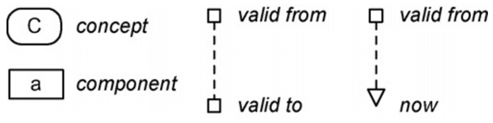

Fig. 1. Elements of the graphical notation of revision timelines.

Using the examples in the following figures, we will examine the functioning of the timeline of objects. Fig. 2 shows a Concept *C*1 that was created at time *t*1 and a Concept *C*2 that was created at time *t*2. Component *a* that refers to *C*2, was created and added to *C*1 at time *t*3. The addition of this component to *C*1 signifies a new minor version of *C*1. At point *t*5, a new version of component *a* was created (for example, because its cardinality was enlarged).

Note that this does not result in a new version of Concept *C*1 that owns it. The deletion of component *b*, however, results in a new major version of *C*1 at point *t*6.

The timeline management of objects makes it possible for the system to find the correct references at any particular moment in time. A change to Concept *C*2, as shown at point *t*9 in the timeline, is thus automatically taken into account when the reference from *C*1 through its component *a* is followed at the current moment in time, indicated as *now*. The mechanism that deals with this follows the timelines of related components and Concepts to their most recent revisions that are alive at a given moment in time. When we want to examine the state of Version 1.3 of *C*1, this mechanism would look up the ‘valid to’ timestamp of *C*1’s Version 1.3 and subsequently find the component *b* Version 1.1 and component *a* Version 1.2 whose ‘valid from’ and ‘valid to’ times straddle this timestamp. Following component *a*, Version 1.1 of Concept *C*2 would be found as the version that is relevant for *C*1’s Version 1.3.

Looking at the latest revision of Concept *C*1 this way (*now*), the reference to Concept *C*2 by component *a* will be followed to the latest Version 2.1 of *C*2.

A more complex example is shown in Fig. 3 where a new version of component *a* was created at point *t*5 by changing its reference from *C*2 to *C*3. *C*2 was then deleted at point *t*6. Component *a* itself was deleted at point *t*10, leading to a new major version of Concept *C*1.

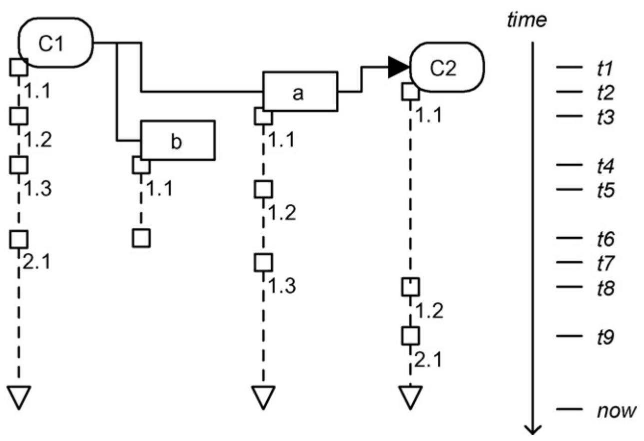

Fig. 2. Example of a timeline of a structure of Concepts.

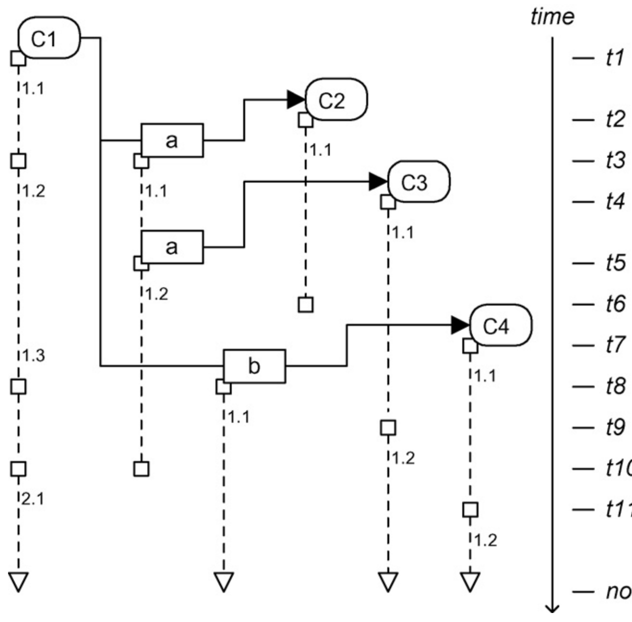

Fig. 3. More complex example of a Concept’s timeline. Component *a* first changes its reference and is later removed altogether, leading to a new major version for Concept *C*1.

3.2.4. Using the version control mechanism

References in the concept-modelling framework are made with an indication of the level of version information that should be included in the reference. The levels used in references are minor, major, and logical, as shown in Fig. 4. Making a reference to an object without any version information signifies a reference to the logical object (see component *a* in Fig. 4). Such a reference will always point to the most recent revision of the referred object at the given moment in time. By including version information, the reference can be restricted to either a particular major version or a particular minor version. When a major version is referenced, the latest minor version within the major version is used. References at the level of revisions are not relevant, since

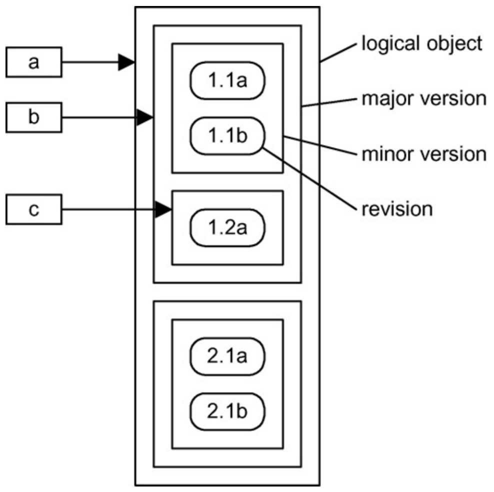

Fig. 4. Four version-levels of detail exist: logical object, major version, minor version, and revision. References to objects can be made to the first three of these levels.

the level of revisions is intended for editing purposes only and cannot be used for making references.

Looking again at Fig. 2, the reference from component *a* to Concept *C*2 at the moment *now* can result either in the retrieval of Version 1.2, for example in case the reference was made to the major Version 1, or in the retrieval of Version 2.1 if the reference was made to the logical object *C*2.

Because revisions and versions, once submitted to the system, cannot be removed anymore, the term ‘deletion’ gets a special meaning. When an object is deleted, its latest revision is marked as ended by setting its ‘valid to’ timestamp. References to the existing versions can still be made, but in the de-referencing mechanism their timeline will be taken into account.

One of the advantages of having version control on the level of objects is that ‘undo’ operations can be performed at the object level as well. ‘Undo’ in this context means to re-establish a previous revision. This does not lead to a factual revival of the particular revision, but to the creation of a new revision that has the state of the previous one. Strictly speaking, ‘undo’ is thus not supported, but the re-establishing of any earlier state of an object is, which is in fact a richer mechanism.

3.2.5. Subscription and notification

In a collaborative design situation, changes to objects made by one user are often of interest to other users. To get informed of such changes, a user can subscribe to notifications issued by an object. If the subscription request was accepted, the notification is handled autonomously by the system and may lead to an automatic update of references or even an automatic upgrade of object versions. The right to subscribe to an object is one of the access rights that the owner of an object can grant to other users, which is a necessary restrictive mechanism built into the system to be able to limit the amount of communication.

4. Benefits and potential

The concept-modelling framework proposes to model design information using a distributed object model. This model provides controlled, multi-user access to both conceptual and instantiated information that is structured in a very flexible manner. It integrates information that remains at the source and in this manner provides a means to integrate business processes. The advantages of this approach include:

- integration of business processes through data sharing;
- enhanced consistency and reduced redundancy;
- control of information remaining with the owner;
- potential to connect a large variety of data sources;
- authenticated and authorised access control in combination with version management.

Although the concept-modelling framework is developed from the requirements identified in the construction industry, its principles and functionality are generic to product design. The potential of this technology therefore reaches many engineering disciplines and, for example, the discipline of industrial design.

5. Research agenda for CSCW

From the current state of the work on the concept-modelling framework we defined a research agenda for the further development of CSCW. Although we have defined this agenda based on the concept-modelling framework, we expect it to have general significance for the construction industry.

With the capabilities of direct access to remote data, be it through the concept-modelling framework or through other web services, the industry will show an increasing need for design and engineering software that can transparently deal with remote data. Having access to shared or exchanged documents that are made available through networks will no longer be sufficient when distributed object models become the prevalent means to structure and manage information.

Although a technology such as the concept-modelling framework and the more generic technology of web services provide a means to technically design such ‘remoting-enhanced’ software, the impact on the working methods will be dramatic and the actually supported design and engineering processes may well need to be rethought. Fundamental research, not only from a software engineering point of view, but from within the construction industry, will be required to address this issue.

As was seen with the advent of using digital media to exchange design and engineering information, the standardisation of communication protocols will be essential for a successful uptake in the industry. While institutional and de-facto standards have appeared for the exchange of product data in documents, a similar standardisation will be necessary for the communication between applications that utilise distributed object models. The access of end-users to the schemata of such models, as is provided in the concept-modelling framework, increases the complexity of the required protocols. However, this perceived complexity should not lead to the conclusion that standardisation at this level is not feasible. Standardisation at this level will be necessary to achieve open-ended solutions that will be acceptable by the industry as justifiable investments.

Some specific areas of design and engineering support will be further developed using the technology in the concept-modelling framework. Initial research results have been published on the implementation of case-based reasoning techniques that utilise the concept-modelling approach [32]. Enabling case-based reasoning tools to access structured, remote data in a transparent manner will increase their capabilities and the scope of the reasoning mechanisms significantly.

Building on results from ongoing research at Eindhoven University of Technology on multi-agent systems [33–36], enhanced approaches to support design and planning processes with autonomous agents representing specific domain knowledge will be investigated. These agents can benefit from the flexibility of the concept-modelling framework and the accessibility of remote data through the framework.

Other forms of creativity support that are currently under development in the Design Systems group will be able to benefit from the capabilities of the concept-modelling approach. The work by van der Zee and de Vries [37] on genetic algorithms aims to generate innovative solutions by combination of existing successful cases. The work by Heylighen and Segers [38] focuses on using linguistic relationships between concepts. Different terminology used for similar concepts potentially forms a limitation to the concept-recognition algorithm. Linguistic relations such as synonyms, hyponyms, etc., can be used to address this limitation by expanding the search space. Integration of this work with the concept-modelling paradigm is expected to lead to mutual benefits.

References

[1] T. Kvan, Collaborative design: what is it? Automation in Construction 9 (2000) 409–415.

[2] T. Kvan, L. Candy, Designing collaborative environments for strategic knowledge in design, Knowledge-Based Systems 13 (2000) 429–438.

[3] G.L.M. Augenbroe, S.R. Lockley, Project management issues in remote CAD outsourcing, in: M.A. Lacasse, D.J. Vanier (Eds.), Durability of Building Materials and Components 8, Institute for Research in Construction, Ottawa, Canada, 1999, pp. 2559–2568.

[4] C.M. Eastman, Managing integrity in design information flows, Computer Aided Design 28 (1996) 551–565.

[5] Z. Turk, Communication workflow approach to CIC, in: R. Fruchter, F. Pena-Mora, K. Rodis (Eds.), Computing in Civil and Building Engineering, 2000, 1094–1101.

[6] C.M. Eastman, Building Product Models: Computer Environments Supporting Design and Construction, CRC Press LLC, Boca Raton, FL, USA, 1999.

[7] G.L.M. Augenbroe (Ed.), COMBINE 2 Final Report, CEC Report HOU2-CT92-0196, Delft University of Technology, Delft, 1995.

[8] ISO-10303. ISO PAS 12006, Building Construction—Organization of Information about Construction Works—Part 3: Framework for Object-Oriented Information Exchange, 2000.

[9] A. Kiviniemi, IAI and IFC—state-of-the-art, in: M.A. Lacasse, D.J. Vanier (Eds.), Information Technology in Construction, vol. 4, Vancouver, Canada, 1999.

[10] K. Woestenenk, Implementing the LexiCon for practical use, in: G. Gudnason (Ed.), Construction Information Technology, Reykjavik, Iceland, 2000.

[11] M. Böhms, F. Tolman, Building and construction eXtensible mark-up language (bcXML), in: G. Coetzee, F. Boshoff (Eds.), IT in Construction in Africa, Mpumalunga, South Africa, 30 May–1 June, 2001.

[12] F. Tolman, J. Stephens, R. Steinmann, R. van Rees, M. Böhms, A. Zarli, bcXML, an XML vocabulary for building and construction, in: Symposium Report on the 2nd Worldwide ECCE Symposium. Information and Communication Technology in the Practice of Building and Civil Engineering, RIL—Association of Finish Civil Engineers, Espoo, Finland, 6–8 June, 2001.

[13] J.P. van Leeuwen, Modelling architectural design information by features, Ph.D. Thesis, Eindhoven University of Technology, Department of Architecture, Building, and Planning, Eindhoven, The Netherlands, 1999.

[14] J.P. van Leeuwen, S. Fridqvist, Design knowledge sharing through internet application, in: Proceedings of CE 2002, International Conference on Concurrent Engineering, Cranfield, UK, 27–31 July, 2002.

[15] J.P. van Leeuwen, S. Fridqvist, Supporting collaborative design by type recognition and knowledge sharing, Electronic Journal of Information Technology in Construction (2002) 8.

[16] J.P. van Leeuwen, H. Wagter, R.M. Oxman, Information modelling for design support—a feature-based approach, in: Proceedings of the 3rd Conference on Design and Decision Support Systems in Architecture and Urban Planning, Spa, Belgium, (1996), pp. 304–325.

[17] J.P. van Leeuwen, H. Wagter, Architectural design-by-features, in: Proceedings of CAAD Futures'97, TU München, München, Germany, 1997, pp. 97–115.

[18] J.P. van Leeuwen, A. Hendricx, S. Fridqvist, Towards dynamic information modelling in architectural design, in: Construction Information Technology—Proceedings of CIB W78, Mpumalanga, South Africa, 30 May–1 June, 2001.

[19] J.J. Shah, M. Mäntylä, Parametric and Feature-Based CAD/CAM, Wiley & Sons, New York, 1995.

[20] W.F. Bronsvoort, F.W. Jansen, Feature modelling and conversion—key concepts to concurrent engineering, Computers in Industry 21 (1993) 61–86.

[21] W.F. Bronsvoort, F.W. Jansen, Multi-view feature modelling for design and assembly, in: J.J. Shah, M. Mäntylä, D.S. Nau (Eds.), Advances in Feature Based Manufacturing, Amsterdam, Elsevier Science, The Netherlands, 1994, pp. 315–330.

[22] W. van Holland, W.F. Bronsvoort, F.W. Jansen, Feature modelling for assembly, in: W. Straszer, F. Wahl (Eds.), Graphics and Robotics., Springer-Verlag, Berlin, Germany, 1995, pp. 131–148.

[23] C.M. Eastman, Information exchange architectures for building models, in: M.A. Lacasse, D.J. Vanier (Eds.), Durability of Building Materials and Components 8, Institute for Research in Construction, Ottawa, Canada, 1999, pp. 2139–2156.

[24] C.M. Eastman, T.S. Jeng, A database supporting evolutionary product model development for design, Automation in Construction 8 (1999) 305–324.

[25] S. Fridqvist, Property-oriented information systems for design, Ph.D. Thesis, Lund Institute of Technology, Department of Construction and Architecture, Lund, Sweden, 2000.

[26] A. Ekholm, Principles for classification of properties of construction objects, in: K. Agger, P. Christiansson, R. Howard (Eds.), Distributing Knowledge in Building—CIB W78 Conference Proceedings, Aarhus, Denmark, 2002.

[27] S. Datta, Managing design knowledge with mixed-initiative dialogue, in: H.J.P. Timmermans, B. De Vries (Eds.), Design and Decision Support Systems in Architecture, Eindhoven University of Technology, Eindhoven, The Netherlands, 2002.

[28] W3C-XML, Extensible Markup Language (XML) 1.0, second ed., World Wide Web Consortium Recommendation, 6 October 2000, http://www.w3.org/TR/REC-xml, 2000.

[29] W. Cellary, G. Jomier, Consistency of versions in object-oriented databases, in: Proceedings of the 16th VLDB Conference, Brisbane, Australia, (1990), pp. 432–441.

[30] P.A. Bernstein, Repositories and object-oriented databases, in: Dittrich, Geppert (Eds.), Datenbanksysteme in Buro, Technik und Wissenschaft (Proceedings of BTW Conference), Springer-Verlag, Berlin, Germany, 1997.

[31] W.E. Kimber, S. Newcomb, P. Newcomb, Version management as hypertext application: referent tracking documents, in: B.T. Usdin (Ed.), Proceedings of Markup Technologies'99, Philadelphia, PA, USA, December 7–9, (1999), pp. 185–198.

[32] S. Fridqvist, J.P. van Leeuwen, Feature type recognition—implementation of a recognizing feature manager, in: Distributing Knowledge in Building—Proceedings of CIB W78, Aarhus, Denmark, 2002.

[33] T.A. Arentze, H.J.P. Timmermans, A multi-agent model of negotiation processes between multiple actors in urban developments: a framework and results of numerical experiments, Environment and Planning B: Planning and Design 30 (2003) 391–410.

[34] J. Dijkstra, H.J.P. Timmermans, Towards a multi-agent model for visualizing simulated user behavior to support the assessment of design performance, Automation in Construction 11 (2002) 135–145.

[35] H.H. Achten, A.J. Jessurun, An agent framework for recognition of graphic units in drawings, in: K. Koszewski, S. Wrona (Eds.), Proceedings of the 20th International Conference on Education and Research in Computer Aided Architectural Design in Europe, Warsaw University of Technology, Warsaw, Poland, 2002, pp. 246–253.

[36] J. Beetz, J.P. van Leeuwen, B. De Vries, Towards a multi agent system for the support of collaborative design, in: J.P. van Leeuwen, H.J.P. Timmermans (Eds.), Proceedings of the Seventh International Conference on Design & Decision Support Systems in Architecture and Urban Planning, Eindhoven, The Netherlands, 2–5 July, 2004.

[37] A. van der Zee, B. De Vries, Computer Aided Evolutionary Architectural Design. Fifth International Conference on Generative Art, Milan, Italy, 2002.

[38] A. Heylighen, N.M. Segers, An architectural Shift + F7—supporting concept development through design cases, in: Proceedings of the ARCC/EAAE 2002—International Conference on Architectural Research, Montreal, Canada, 2002.

[39] http://www.iai-international.org/.

J.P. van Leeuwen is associate professor in collaborative design at the Design Systems Group in the Department of Architecture, Building, and Planning at Eindhoven University of Technology. He has a background in architectural engineering, commercial software development, and scientific research in building information technology. His research interests include digital design, collaborative design, design information modelling, product modelling, and design knowledge representation. He is co-ordinator of a Master of Science programme on design & decision support systems, supervisor of MSc students and co-supervisor of PhD candidates on collaborative design. He holds an MSc and PhD degree from Eindhoven University of Technology, The Netherlands.

S. Fridqvist holds an Architect's degree and a PhD from Lund Institute of Technology, Sweden, where he conducted research and development in the area of computer aided architectural design for some 15 years. He was affiliated as post-doctoral researcher with Eindhoven University of Technology, The Netherlands, for two years. His research interests focus on information systems for product representation, specifically on what requirements the design process poses on such systems, and on ways to augment the design and production process with semantically rich product representations.