A Distributed Object Model for CSCW in the Construction Industry

Jos P. van Leeuwen and A. van der Zee

Eindhoven University of Technology, Design Systems Group
www.ds.arch.tue.nl
J.P.v.Leeuwen@bwk.tue.nl

ABSTRACT: Information about products for the construction industry is increasingly often provided to designers in digital ways that enable them to apply the information directly in the design process. Digital product catalogues are provided using various media and formats and several initiatives are taken by the industry and by CAD developers to integrate this kind of information into CAD systems. Generally, current practice is to distribute the information to designers, for example, by using CD-ROMs or a website where the information can be downloaded. In our research we recognise that distributing information in this manner detaches it from the business processes in the construction supply chain, which is a major disadvantage.
The project presented in this paper concerns the implementation in the Dutch construction industry of a methodology for sharing product information through a distributed object model. The methodology, which is called Concept Modelling, forms a generic basis for the support of collaborative design, but is applied in this project to the integration of information from the supply chain in the design process. Through the distributed object model, design information and product information can be integrated while the actual data objects remain at their source. This enables the supply chain to provide information of a high semantic level to designers while keeping the control over the information and maintaining the relationship of the information with their business processes.
The advantages of this approach in which information is shared, rather than exchanged, are numerous. Redundancy of information is minimised, consistency is improved, and updated information is available immediately. Moreover, design and construction processes can benefit significantly from the dynamic aspects of accessing information that is tied to business processes in the supply chain. For example, product selection during design can be based on latest information on product details, prices, production methods, and variants of products. This information can be provided to designers automatically and on demand.

KEYWORDS: Collaborative Design, Product Data Modelling, Concept Modelling, Distributed Object Model, Semantic Web.

1. INTRODUCTION

The availability of adequate product information is one of the aspects in building design that have a large effect on quality and costs of the construction process and of the final building. Design faults are often caused by incorrect or misconceived product information or by improper selection of products because of lacking information. Such mistakes in the design stage can have dramatic consequences for the construction process, when ad hoc solutions or replacements of products in the least case obstruct the process and invariably are cost-intensive, time consuming, and likely to have a negative effect on the eventual quality. If such mistakes become evident only later, while the building is already in use, the possibilities for correction are often very limited and the costs much higher. A survey by (Josephson and Hammarlund 1999) shows that 15-30% of all defect costs during production are caused by design mistakes. After construction, during maintenance, design mistakes are the cause of 40-55% of the defect costs. The same study shows that over 60% of the defect costs in construction that are caused by design mistakes can be traced to a lack of knowledge or information.

The quality and availability of product information depends largely on the form and media used to distribute this information. The following aspects determine the value of product information for design:

- Semantics (is the meaning of the information sufficiently defined and understood?)
- Validity (is the most actual information available?)
- Format (can the information be accessed and applied directly in the design context?)

- Timeliness (is the information found and available when needed?)

Current practice in the supply chain of the construction industry is to distribute product information, for example in the form of catalogues, either in paper format or in a digital format that is likewise rigid, such as CD-ROMs or documents that can be downloaded from a website. In the more advanced cases, information is produced on demand by web servers and can thus be tailored to specific requests. However, once provided, the information is no longer in control of the supplier and the consumer of the information has no guarantee of its validity.

The usability of product information in design processes also depends on how well the meaning of the information is understood by the user. Obviously, design support systems require a high level of explicit semantics to be able to interpret and process data.

The research project described in this paper is named *CoDesKs*, for Collaborative Design Knowledge services. The objective of this project is to offer a paradigm for information modelling and communication in design that on the one hand enhances the explicit semantics of information and on the other hand improves the validity and timeliness of information in a collaborative design environment.

2. DISTRIBUTING PRODUCT INFORMATION

The purpose of distributing product information is generally twofold: to communicate about merchandise and to provide details about the technical application and organisational issues concerning the product. There are many reasons why the information concerning a product can become outdated. For various reasons, such as commercial ones, there is a strong urge to innovate, with new models emerging, new materials being applied, new features added, new options, applications, technical solutions, etc. Another cause for the limited validity of product information is its relation to a specific application, for example in a particular construction project. This relation may have an organisational nature, such as contractual agreements on prices and delivery, or a technical nature, for example when the applicability of a product depends on technical aspects of the project design.

Distributing product information through catalogues, on paper or in digital format, does not support the demand for up-to-date or project-bound information. Using websites to download product data only improves the timeliness of information; it does not improve its shelf life. More advanced websites are able to produce customised information, taking project or client specific data into account, but again this does not improve the validity of the information over time after it has been provided.

The validity of information that a designer obtains from partners in a project can only be guaranteed by sharing of the information resources. This means that, rather than providing a copy of the information, the information is accessed at its source where the provider of the information has full control (and responsibility) over it.

To achieve such sharing of information, we propose a change of the paradigm 'distributed product information' from the supplier point of view to the consumer point of view. 'Distributed' no longer means 'sent to many clients' but rather 'accessed at many providers.' Sharing distributed information resources has the potential to improve business processes in many ways:

- Avoiding unsolicited communication, the traffic of information is reduced, even if there is an increased amount of wanted traffic;
- It improves the validity of information, because it remains under control of the provider;
- It increases the quality of information, since it can answer a specific request or even result from a, possibly automated, dialogue;
- It helps to integrate business processes by keeping the relationships between the processes and their output data active.

This paper first introduces the theoretical and technical features of the so-called concept-modelling paradigm that implements a distributed object model for collaborative design. It then discusses the opportunities that this technology creates for a stronger participation of the supply chain in design processes.

3. CONCEPT MODELLING

Concept Modelling is the name of a modelling paradigm that was developed in the CoDesKs project at Eindhoven University of Technology (van Leeuwen 2003). The objectives of this modelling paradigm (van Leeuwen and Fridqvist 2003) are: (a) to give the end-user (designers or other actors in the building process) authority over the schema

of models that are used for the representation of designs and products; and (b) to provide a consistent information modelling environment that supports distribution of data sources and multi-user access.

The first objective, user authority over the modelling schema, is addressed by the dynamic nature of the modelling paradigm. In principle, this is an object-oriented paradigm, but there are many features to it that increase its flexibility such that end-users have a high level of control over the exact definition (and thus semantics) of objects.

The second objective, consistent multi-user access to distributed data, is achieved mainly by the implementation of remote data access, using Internet technology, in combination with an object-level version control mechanism.

Both aspects of the modelling paradigm are discussed in more detail in the next two subsections.

3.1 User-access to modelling schemata

The concept-modelling paradigm uses the term *concept* to denote logical notions on which reasoning in design is based. This includes notions of construction elements, like floors and walls, but also non-tangible notions, like spaces and routing. Also, aspects such as colour, strength, temperature, etc., are notions that are represented by concepts. In the definition of a concept, there is no distinction between the representation of *objects* and *properties*. This distinction only becomes obvious in the application of the concept in a modelling context. The reason for this is that a concept will be viewed upon as an object in one context, but regarded a property in another. For example, a concept that represents the notion of 'usage function' in the design of a building will be used as object in early stages of reasoning about a design, but will be assigned as a property to spaces during later stages.

Concepts are defined in a formal manner, using the following five mechanisms:

- Value representation
- Interrelationships
- Prototypical versus individual concepts
- Multiple inheritance

Value representation and interrelationships

In its most basic form, a concept is a simple named entity, e.g. 'length,' that can have a *value*, e.g. the numeric value 5.4, and a unit for this value, e.g. 'm.' More complex concepts are defined through relationships to other concepts. For example, a concept named 'steel beam' would relate to concepts defining its profile, its material properties, and the concept 'length.' The different relationships that can exist between concepts are categorised into: *decomposition*, *association*, and *specification*. The latter type of relationship indicates that a particular aspect or detail of a concept is specified by another concept, like the length specifies an aspect of the beam. Decomposition relationships denote whole-part type of relationships, e.g., a steel beam is decomposed of a body and a flange. Associations indicate relationships between concepts that are in principle independent but in some way associated, for example the association between a wall and a space.

All relationships between concepts are identified using *role names*. These describe the particular role of the related concept in the context of the concept that defines the relationship.

Prototypical versus individual concepts

We can reason about design and model a design in two distinct modes. One mode is to think about design in terms of typologies. We do this when we talk about the generic properties of, for example, a type of building element. For this kind of reasoning, we can model *prototypical concepts* (also called prototype concepts, or simply *prototypes*). On the other hand, when we reason about and model a particular design case, we need to provide specific information about the case, which is modelled using *individual concepts* (or *individuals*).

While these two kinds of concepts share many features, their meaning is slightly different[1]. For example, the value of a prototype concept denotes the default, or assumed, value of such a concept. The value of an individual concept, however, denotes the particular value of that concept in the context of the particular design case.

Individual concepts are always modelled on the basis of prototype concepts; they instantiate one or more prototypes and can implement all relationships that are defined for those prototypes. This way, building elements can be modelled that integrate multiple design concepts, for example, an element that integrates the functions of both wall and furniture.

The difference between prototypes and individuals becomes particularly evident when looking at the relationships. Relationships defined between prototypical concepts could be regarded as

[1] While this approach addresses the class-instance dichotomy as discussed in (Fridqvist 2000), it does not completely eliminate the dichotomy, the way Fridqvist proposes.

the variables of concepts, while the relationships of an individual provide the actual data of those variables. There are many features of the modelling paradigm that make it very flexible and allow, for example, ad-hoc relationships between individuals that have no counterpart in the prototypes.

Multiple inheritance

The concept-modelling paradigm implements a multiple-inheritance mechanism: a prototype concept can inherit relationships from other prototype concepts. This allows a structured and layered organisation of design concepts, which is an important feature for standardisation and communication protocols. When a prototype inherits from another prototype, all relationships of the 'super-prototype' also apply to the 'sub-prototype.' Individual concepts that are based on such a sub-prototype can implement all relationships defined for the sub-prototype and its super-prototypes. Sub-prototypes can override relationships of super-prototypes, in order to make them more specific.

Figure 1 shows a network of prototype and individual concepts. It demonstrates multiple inheritance, as well as the prototyping mechanism.

3.2 Multi-user access to a distributed object model

The above-described features of the concept-modelling paradigm allow designers to formalise design knowledge and to model design cases. In practice, they would never do this in isolation: design is always a process of collaboration. Even when a particular task is not performed in direct collaboration with other individuals, a designer will always access or re-use information from external resources. There are many ways to bring together information from multiple resources. Currently the most popular approach is to use project-webs. These are websites where all collaborating partners in a project store their information, making it accessible to all. The main advantages are that such a project-web provides a central entry-point to the project information and allows centralisation of the data-management, such as security, backup maintenance, and document version control.

One major disadvantage of using project websites is that all partners need to be disciplined in keeping the information updated at the server and must refrain from sending information to each other through other routes, e.g. using email. Another major problem with project webs is that they are document-based and draw a strict line between project-specific and project-independent information. Because documents are moved away from their source to the central storage location, information that is in principle independent of projects, such as information describing the products and services of a company, automatically becomes project-specific once it is entered into the project website. As a consequence, this information is disconnected from its source and from the underlying business processes. This implies a considerable risk of inconsistencies and the usage of outdated information.

Remote data access

The CoDesKs project has incorporated the concept-modelling paradigm into an object model that

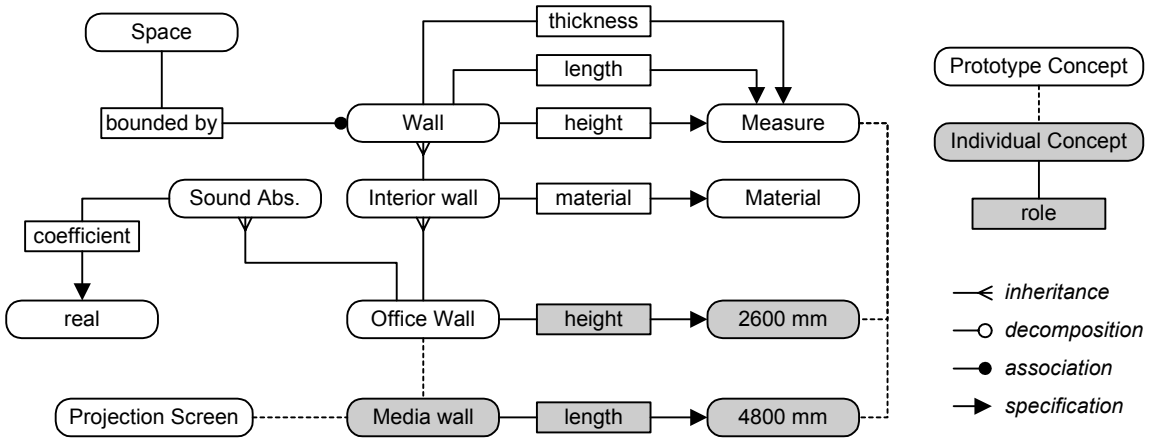

Figure 1. Example of a network of concepts. The prototype concept 'Office Wall' inherits from the 'Interior Wall' and the 'Sound Absorbing Element' concepts. It overrides the inherited 'height' relationship by fixing it to 2600 mm. The 'Media Wall' is an individual concept based on the prototype 'Office Wall' of which it uses the height; the length is added to this individual. The 'Media Wall' also implements the prototype 'Projection Screen' (no further details shown here).

offers remote access. Essentially, this offers the possibility to build applications that can access objects directly at remote resources. Rather than having to exchange information in the form of documents, such applications can share information in the form of objects.

The technology applied in this approach is standardised, HTTP and SOAP, through the implementation provided by Microsoft™ .NET Remoting facilities.

There are a number of conditions that need to be met before remote data access can be practically applied in a collaborative design context. First of all, objects, and in our case these are concepts, must be uniquely identifiable. For this purpose, concepts are organised using the notion of namespaces that are themselves identified through URI's (Uniform Resource Identifiers). This mechanism provides the capability to uniquely identify each concept and concept-relationship in a consistent and persistent manner.

A second condition for a proper organisation of remote data access is security. Obviously, data must be protected from unauthorised access, while authorised users must have sufficient rights to read or write data. In the concept-modelling paradigm, the system of access rights is more complicated because there are several levels of access that enable users to read, copy, use, inherit, or modify concepts. Access and ownership is controlled on the basis of user groups.

A third feature required from remote data access is a locking mechanism to prevent simultaneous modifications to objects by multiple users. This is implemented by way of a checkout mechanism. When a user accesses data for modification, the data is temporarily inaccessible for modification by other users. At all times, data remains accessible for operations other than modifications, such as reference or inheritance operations. The period of locking depends on the kind of modification that the user's application is performing; real-time graphical operations will take longer than non-graphical changes to data.

Finally, notification is a fourth requirement of useful remote data access. When multiple users access the same data resources, they probably like to be informed of modifications to that data. A subscription mechanism allows users to be subscribed to notifications that are sent when data is modified. Examples of such modifications are changes to the design or the release of a new type of a product to the market. To a certain extent, these notifications can be handled by the system automatically, for example to update the graphical onscreen presentation. Other notifications may require human reaction, for example to evaluate the consequences of a change in the design or to consider the application of a new product.

Object-level version control

Version control is necessary in a design system, and particularly in a collaborative design system, for a number of reasons (van Leeuwen and Fridqvist 2003). Firstly, version control is a way of recording user actions. Such a record can be used for many purposes, e.g., allowing the user to undo certain actions or enabling the user to inspect and replay the history of the design process.

Expanding on such a timeline of the design process, the second reason to provide version control is that it can be used to administrate design alternatives.

But in the context of collaborative design, version control of objects is above all important to maintain the consistency of an object model that is accessed by multiple users. Changes to objects are administered through the creation of new versions, which 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 control.

In literature, version control at the object level is described in (Cellary and Jomier 1990), who use so-called 'stamps' to identify object versions in multi-version databases; in (Bernstein 1997), proposing basic operations on versions that are identified through a succeeds relationship; in (Kimber, Newcomb, and Newcomb 1999) 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 archive 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 also has potential for, e.g., the narrative representation of designs and for computer applications used in design education and research.

In the concept-modelling paradigm, version information for objects is organised into three levels: major versions, minor versions, and revisions.

These three levels relate to the kinds of modifications that can be made to objects. A modification to an object is started by a checkout of the object, which locks the object for modifications by other users. It is concluded either by committing a new revision or by submitting a new version. Revisions are used to accumulate modifications until the user concludes that a new version is ready to be created. New versions are in principle minor versions, unless either the user or the system requires the creation of a new major version. The system will require a new major version when it cannot automatically upgrade references by other objects to the next version. This helps identify potential consistency issues in the model that require attention by the user.

This approach of storing all modifications as revisions or versions of objects helps to increase the consistency and integrity of the objects and the relationships between objects from various resources. At the same time it requires smart ways of identifying objects when making references to versions and resolving and updating these references. The object versioning mechanism implemented in the concept-modelling paradigm utilises timeline management for this purpose. The timeline of an object administrates the beginning and ending of each revision and version. Through this mechanism it is possible to identify the relevant relationships for a given concept and the concepts that form its context. An example of the timeline of concepts and relationships between concepts is shown in figure 2, which also illustrates the three levels of references required for this versioning system. Details of the implementation and implications of the object version control mechanism and the timeline management can be found in (van Leeuwen and Fridqvist 2003).

3.3 Related developments

The information modelling approach proposed in the concept-modelling paradigm bears much resemblance with technologies such as XML (W3C 2003a) and RDF (W3C 2003b) and with the development of the Semantic Web (W3C 2003c). While a thorough comparison is outside the scope of this paper, it is relevant to mention here that the concept-modelling paradigm could be regarded as a more specific form of semantic web. Where the W3C Semantic Web effort aims to standardise a very generic way of expressing semantics for the context of the world-wide web, the concept-modelling paradigm goes somewhat further in its classification of relationships between objects (comparable to the predicates in RDF). In comparison with the semantic web, the structure of prototype and individual concepts is also more restrictive. The reason for these restrictions is that we believe that the ability to make more detailed assumptions on the structure of information offers us better opportunities to develop more intelligent

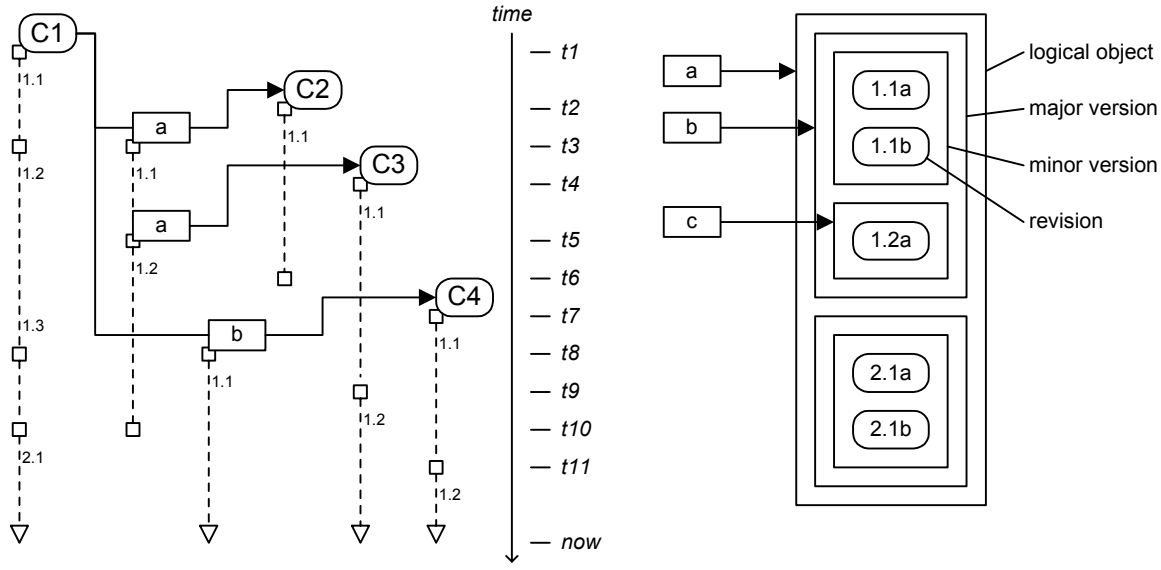

Figure 1. Left: example of a timeline of concept versions. At moment t5*, the relationship* a *from concept* C1 *to concept* C2 *is changed into a relationship to concept* C3*. Although this does not lead to a new version of concept* C1*, this change can be traced through the concept's timeline.*

The figure on the right shows the three levels of references related to the three levels of version information. Reference type a *refers to the logical object, while reference type* c *refers to the minor version. References to revisions are irrelevant (van Leeuwen and Fridqvist 2003).*

support, for example in the form of case-based reasoning tools and agent technology.

4. OPPORTUNITIES FOR THE SUPPLY CHAIN

The capabilities of defining and sharing active information and its semantics were developed in this project to support an expressive, yet formal, way of modelling designs and to support collaboration between designers. At the same time, these capabilities allow other partners in construction projects, including the supply chain, to become more actively involved in the process of collaborative design. As set out in the introduction of this paper, the availability of product information in the design process has a major influence on the quality and costs of the final design. Therefore, ability to increase their role as active participants in design processes is an exciting opportunity for product suppliers. Besides offering competitive products, the challenge is now to offer high quality information about products and information services relating these products to design projects.

The new technology to share information contents, the semantics of information, and the access to our business processes, opens up almost limitless opportunities in e-commerce. First of all, semantically well-defined information improves the process of product selection and offers a chance to better inform designers about the qualities and features of products. But the implications of this new technology go far beyond this point in improving the relationship between supplier and designer:

- Information objects from the supplier become active objects in the context of the design project. They will update themselves, or notify the designer when updated information is available.
- When enhanced with knowledge about the application of a product, information objects can react to the development of the design, for example by adjusting the features of the product in accordance with its context. This behaviour of the information object does not need to be incorporated into the design application, which is the approach followed in the development of today's CAD systems. In the distributed object model, design objects and product objects from multiple resources form an integration of knowledge from various disciplines.
- Taking this one step further, the supplier's information objects can be tied to business processes such as sales, production, and delivery. On the one hand, this allows designers and project developers to take this type of information into consideration already during design. On the other hand, it facilitates and promotes the re-usage of information models from design stages into construction or even facility management stages.

5. IMPLEMENTATION AND CURRENT DEVELOPMENTS

The concept-modelling paradigm is developed and implemented in the CoDesKs project in the form of an information-management module that takes care of all storage, access, and modification actions on the concept databases. This core module also manages the remote access, the object-based version control, and the resolution of object references. It provides an application-programming interface that can be used to develop either client applications or, e.g., web-interfaces. The concept database is currently persisted in a relational database, but interfaces on the basis of XML and RDF are planned.

The results and experience from the CoDesKs project are currently being input in the development of an industrial standard for integrated software for the Dutch architectural design market. This standardisation effort, named Het Digitale Huis (The Digital House) is a project initiated by Dutch CAD vendors and aims to market new software products based on this standard on a very short term. The suite of products that these software houses develop on the basis of this standard range from CAD software to tools for specification writing, project management, product selection, building codes checking, and facility management. Initial prototyping of the concept-modelling paradigm in this context aims at improving the module for product selection that is used in the applications for architectural design and specification writing.

In two other research projects at Eindhoven University of Technology the usability of distributed object models is investigated.

The first project concerns the development of a method for evolutionary development of design alternatives subject to a set of performance constraints and user requirements. National and local building codes are regarded primarily as constraints that a building design must satisfy (van der Zee and de Vries 2002). These constraints are derived from national standards, developed by national standardisation institutes. They are often

subject to chances. In the distributed object model approach, the standardisation institutes would maintain constraint-objects and provide remote access to them. This way, conformance-checking applications can always use the latest versions of the building codes.

The goal of the second project is to develop an application that checks if a building is designed according to the local zoning plan. During the design process, the designer must have access to the latest version of the zoning plan. Vice versa, the local government needs to have access to the most up-to-date state of the building design, also after the construction of the building. For the latter purpose, authorities would not want to rely on remote access, but always have the latest data with respect to buildings, infra structure, sewer system etc., in their possession. Working with local copies that remain a more or less active relationship with the original data at its source, is one of the future developments planned in this ongoing research.

6. REFERENCES

Bernstein, P. A. 1997. "Repositories and Object-Oriented Databases." In Dittrich, K. R. and Geppert, A. *Datenbanksysteme in Buro, Technik und Wissenschaft (Proceedings of BTW Conference)*., pp.34-46. Berlin, Springer Verlag.

Cellary, W. and Jomier, G. 1990. "Consistency of Versions in Object-Oriented Databases." *Proceedings of the 16th VLDB Conference*., pp.432-441. Brisban, AUS.

Fridqvist, S. 28 Sept. 2000. *Property-Oriented Information Systems for Design*, PhD thesis. Lund University, Sweden.

Josephson, P.-E. and Hammarlund, Y. 1999. "The causes and costs of defects in construction – a study of seven building projects." *Automation in Construction*, vol.8, pp.681-687.

Kimber, W. E., Newcomb, S., and Newcomb, P. 1999. "Version Management as Hypertext Application: Referent Tracking Documents." In Usdin, B. T. *Proceedings of Markup Technologies '99*., pp.185-198. Philadelphia, PA, USA. 7 Dec. 1999.

van der Zee, A. and de Vries, B. 2002. "Computer Aided Evolutionary Architectural Design." In Soddu, C.(ed.). *Proceedings of the 5th International Conference on Generative Art 2002*., pp.9.1-9.13. Milano, I.

van Leeuwen, J. P. 2003. "Computer Support for Collaborative Work in the Construction Industry." *Proceedings of the International Conference on Concurrent Engineering*. Funchal, PT. 26 July 2003.

van Leeuwen, J. P. and Fridqvist, S. 2003. "Object Version Control for Collaborative Design - Characteristics of the concept-modelling framework." *E-Activities and Intelligent Support in Design and the Built Environment - 9th EuropIA International Conference*. Istanbul, TR. 8 Oct. 2003.

W3C. 2003a. "W3C Extensible Markup Language (XML)." *http://www.w3.org/XML/*.

W3C. 2003b. "W3C Resource Description Framework (RDF)." *http://www.w3.org/RDF/*.

W3C. 2003c. "W3C Semantic Web." *http://www.w3.org/2001/sw/*.