Published as: van Leeuwen, Jos P. 2002. "Knowledge Sharing for Collaborative Design."
In: *Proceedings of 6th International Conference on Design and Decision Support Systems in Architecture and Urban Planning*, July 7-10, 2002, Ellecom, The Netherlands.

Knowledge Sharing for Collaborative Design

Jos P. van Leeuwen
Eindhoven University of Technology
Department of Architecture, Building, and Planning
Design Systems group
The Netherlands
www.ds.arch.tue.nl

ABSTRACT

This paper presents the preliminary results of a research project that aims to develop and apply Design Knowledge Servers (DesKs) in the building and construction industry. The paper starts with a view on where the development of computer application in design and construction might be heading. It briefly looks at the current situation concerning information modelling for design support and compares two alternative ways of developing standards in this area. After this sketch of its context, the DesKs project is introduced and its objectives, characteristics, implementations issues, and application scenarios are discussed. The paper concludes with a preview of the next steps we will attempt to take and with an explanation of why it all takes so long.

1. DESIGNING (IN) THE FUTURE

"The discipline of Design for Construction has seen considerable improvements in the twenties and thirties, mainly under influence of the recent advancements in comprehension-logic. In general terms, comprehension-logic has made it possible for computers to interpret the semantics of human-formatted information. Application of this technology in design for construction has greatly improved the way computers can be used to support this highly creative discipline. The advantages of using comprehension-logic for design are evident. Current practice modelling systems allow designers to specify the design rationale in their own terminology, using personalised learning systems to augment the computer's understanding of the individual's preferred ways of expression. Moreover, most systems today are capable of active participation in design, using reasoning-engines that are now common practice in many business processes, just like grammar checking and auto-correction tools in 20th century 'word processors[1]'. The main difference being, obviously, that those early tools functioned by application of rules, at best in combination with case-bases, since comprehension-logic was not yet available.

Another type of application of comprehension-logic is found more at the background of the design task. Routine tasks, such as searching and matching product

[1] Word processors were software packages that processed the human input of texts through keyboards where the user had to *type in* the text *letter-by-letter*. Evidently, auto-correction tools were needed to correct accidentally misspelled words. The main problem with these tools was their limited capability to comprehend the semantics of what was being typed, so that suggested corrections were not context sensitive. The early systems in fact supported only syntax checking and misspellings that still had still had a valid syntax where not detected.

specifications, optimised product selection, cost management, and production planning, are now almost completely automated. In these routine tasks, human interaction is only needed for final selection and approval and in cases of non-deterministic preferences. Other tasks are not so much auto-completed but rather computer-prepared. An example is the task of detailing a design, in which computers are able to assist only by generating detailed alternatives for general solutions created by the designer, whose expertise and interaction is often needed still for evaluation and final perfection of construction details.

What many seemed to fear around the turn of the century has, of course, not happened: computers have not taken over design and the profession of design manager has not ceased to exists, even if the term *architect* is no longer preferred. What has happened, fortunately, is that the recent innovations have made it possible for these professionals to deliver high quality design alternatives at a much lower cost and in a format that is 'ready-to-build'. The way design and construction have now been integrated almost seamlessly, together with the increased level of automation in the construction process, allows for a much more direct input from design into the construction-phases. This has dramatically reduced the costs of *individualised mass-production* in construction, close to the cost-level of conventional mass-production, and it will soon become as feasible as it is today in the transporter industry." (Unknown, 2028)

2. HOW TO GET THERE?

Before attempting to set out a path of how we can get anywhere near the innovations described in the previous section, we must take a look at the current situation. Restricting our field of view to how design information is modelled using present-day technologies, we can make the following observations:

1. Product modelling is gradually becoming current practice. However, in current practice, product modelling often means using a CAD system that can deal with application-specific, non-geometrical data in relation with, or rather based on, geometrical models.
2. Exchange of design data is either system-dependent (obliging partners to use the same tools), based on intermediate file exchange formats (e.g. an ISO-STEP format), or it uses a common implementation of standardised formal models, such as the Industry Foundation Classes (IFC's). All three approaches have in common that they are based on a predefined formal modal of concept definitions. The means to express design representations that need to be communicated are restricted to these predefinitions.
3. The schemas of the above mentioned standardised formal models are currently not targeted specifically at design support during all design stages, but rather at finalised design and construction stages, which limits their usability in supporting early design stages. The main cause of this limitation is that, except for the most trivial design cases, the concepts and data constructs used for

reasoning and modelling in final design stages and construction are not the same as those useful in earlier design stages.

4. Standards of formal models will not be able to cover the whole domain of the building and construction industry. When defining such models, there is always a trade-off between general applicability of the model versus its level of detail. Arbitrary decisions must be taken, which have a major impact on the scope of the model. Therefore, every formal model will have its limitations as to what it can represent. It will always cover only a portion of what could possibly be modelled. (Amor and Faraj, 2001).
5. Additionally, concepts used in innovative design are often determined only during design and thus cannot be standardised. Consequently, innovative designs can currently only be modelled using modelling techniques with lower level of semantics, such as geometry modellers, or by incorrect usage of standards, which leads to misinterpretation, etc. Only in large-scale and very prestigious construction projects, it is feasible to define custom data models that accurately represent the design rationale. For routine design this approach is not feasible, since the technology that allows formal definition of custom design concepts by end-users is not yet available in commercial software.
6. If the means to express design in a manner suitable for the individual designer's needs are not yet available, let alone the logic needed to make computers understand such expressions that would be full of designer-specific semantics and terminology that the designer would like to define.
7. Looking at the resources used for design, many of these are available digitally but in poorly structured formats. One of the most important resources for detailed design is product data. Although product data is increasingly often made available in digital media and accessible through Internet, it is not ready to be consumed by design support systems. Hence, intensive human effort is required still for searching for product data that matches the design specifications and for integrating this data into the design model.

Adding to the problems concerning the definition of formal models, mentioned above, there are problems related to the communication and integration of models. The often very different views of professional disciplines cannot be reconciled by attempting an integration of their formal models and communication between these disciplines is possible only "through the application of considerable human intellect and experience to understand [each other]." Reconciliation of the formalised models of different disciplines "would require greater application of artificial intelligence than is currently possible." (Amor and Fajar, 2001). The alternative to reconciliation of formal models is mapping, defining procedures to translate the concepts used in one model to those used in another. In many cases, this is simply not semantically possible or only in an incomplete manner and most mappings are non-symmetrical, meaning that mapping back cannot be done without loss of meaning.

Considering these observations, one question we can ask ourselves is whether standardisation is the way to go. A proper answer to this question can probably be

Published as: van Leeuwen, Jos P. 2002. “Knowledge Sharing for Collaborative Design.”
In: *Proceedings of 6th International Conference on Design and Decision Support Systems in Architecture and Urban Planning*, July 7-10, 2002, Ellecom, The Netherlands.

given only with hindsight, but we can attempt to predict where alternative ways of developing standards will lead us.

It is rather certain that the IFC’s, which are developed by the International Alliance for Interoperability (IAI, URL-1), will gain popularity as a de-facto standard and that an increasing number of software applications will support them. This can be concluded on the one hand simply by applying the rules of the market. Big players in the field of CAD vendors are joining this effort, taking their users with them. Other software producers will have to follow if they want to provide support for data-exchange with the bigger CAD applications. On the other hand, it must be mentioned that in the case of the IAI, the building and construction industry itself is strongly involved in the development of the standard, which obviously makes the adoption of the standard much more likely.

However, being a standardised formal model, filled with a large number of concept definitions that range from generic ones to relatively detailed ones, the IFC’s will always have limited applicability. They will never cover the industry’s need completely. The most likely areas where they will fail to support, are those where concept definition is most difficult and agreement is hard to achieve or even not yet relevant. These are the areas of early design stages and innovative design. Areas where design concepts are ambiguous, unstable, subjected to change during the process, completely new, or by default everything but standard.

The conclusion is that the development of the IFC’s will be widely supported by industry. They solve many problems and the drawbacks and limitations will be forcefully dealt with in most cases. Nevertheless, in certain stages of the design process and in certain kinds of design projects, they will be considered useless. If we want to get anywhere near the situation described in section one of this paper, standards must become more flexible in dealing with ‘non-standard’ situations.

An alternative approach to standardisation that has the potential of being much more flexible and thus supporting a much wider range of design cases, is by standardising not *what type* of concepts we describe, but the *way* we formally describe design concepts. Naturally, the ongoing standardisation efforts, such as ISO-STEP and the IFC’s, have developed and/or adopted such standards for describing concepts. The Express definition language is used in both developments, and XML is becoming a generally accepted language for data exchange. However, these definition/description languages were developed as a standard for software developers, not end-users such as architectural designers. If we want computers to be able to interpret the concept definitions that designers use, which are not fixed but a part of the act of design, we must provide them with means to formalise and work with these concept definitions in a standardised manner, before we can even start to think about the logic needed to interpret them. An important objective here is that the activity of formalising concepts fits naturally into the design activity itself, and does not force the designer to change his way of thinking because of limitations of his means of expression.

XML is a good step forward because it is a language with much flexibility to express information in a structured manner. The Express language has stronger data-awareness and contains some constructs that allow a richer definition of data structures. However, the accompanying STEP Physical File Format to describe models based on an Express-Schema lacks the flexibility found in XML documents.

3. THE LITTLE STEP WE HAVE MADE

This section describes the development in the project dubbed *DesKs*, *Design Knowledge Servers*. The project's overall aim is to develop an Internet-based environment for the support of collaborative design. More specifically, DesKs will provide a *shared* environment for designers to model design concepts in a formal manner and to use these formalised concepts in design modelling and reasoning. The project is based on the modelling paradigm of the Feature-Based Modelling (FBM) framework, as defined in (van Leeuwen, 1999). Using this framework to model design concepts both in a generic manner and for a particular design case, the DesKs project will allow the designer to share this design knowledge with others and to use the knowledge provided by others. This capability makes it possible to use DesKs for the management of collaborative design projects, but also for publication of design information, or sharing generic design knowledge in a scope beyond particular projects.

The DesKs project is strongly related with a parallel project that develops the technology of Feature Type Recognition (FTR). The FTR project is conducted by Sverker Fridqvist at Eindhoven University of Technology (Fridqvist and van Leeuwen, 2002). It will lead to the definition of algorithms and the development of prototype tools that allow the computer to compare the outcome of design activities with stored design knowledge. This technology will be useful in many applications, such as searching for building product information that is published through DesKs (see also section 3.3). Together with the facilities of the DesKs project, it forms a first small step in the direction of making computers understand what we design.

3.1 Design Knowledge Servers and Feature-Based Modelling

The DesKs project involves the development of a system that can be used to build a network of Design Knowledge Servers based on the FBM framework. Two kinds of software applications are envisioned that in principle build on the same set of functional specifications but provide different types of access to shared design knowledge. The common characteristics of these applications are described in this section, separating them in relation with two groups of requirements.

The following paragraphs describe the most important characteristics and implementation issues of the FBM framework that serve requirements of **design support**.

3.1.1 Property oriented modelling

The FBM paradigm is a property-oriented way of modelling (van Leeuwen et al., 2001). Property-oriented modelling approaches take the properties of real-world things and concepts as the basic entities of modelling and allow the modeller to compose a model of the real world (or design) by collecting properties that define the subject of modelling. The FBM framework, in principle, does not distinguish properties from objects; both are called *features*. It depends on the context in which a feature is used whether it functions as a property or a key-object in a model. This approach has been developed because it is better able to follow the dynamic way that information is dealt with during design than approaches that predefine the properties of objects. The meaning assigned to information during the various design stages is updated continuously, as the design develops. A 'spatial function' feature, for example, may at one stage in the design process be regarded a key-object in the model, while at a later stage it is assigned as a property to a 'space' feature that represents the space where the particular function is performed.

3.1.2 User-defined typologies

Design concepts are formally modelled, using the FBM framework, into so-called *feature types*. Feature types provide the templates for creating *feature instances* that represent actual design data. In product modelling terms: feature types define the schema for models of feature instance. While most formal models of design provide a fixed schema, the FBM framework allows designers to extend the schema with their own definitions of feature types: custom design concept formalisations. Designers can add to standard definitions and thus build up their own terminology and library of concepts used for design. The extendibility of the conceptual schema also serves many other purposes where new typologies must be added to standard collections, for example to represent new construction products or methods.

3.1.3 Flexibility in modelling

The property-oriented way of modelling makes the relationships between chunks of information in the model very flexible. If the space that a spatial function is assigned to is removed because it was merged with another space, the spatial function, as a property, will continue to exist in the model. The intention of removing a space object from the model does not necessarily imply that all its properties are to be removed as well, although a new assignment for the spatial function property might be required. The independence of properties from objects provides the kind of flexibility in modelling that corresponds well to the dynamic way of thinking that is characteristic for creative design.

In the FBM framework, this kind of flexibility is made possible by the fact that all feature types and feature instances have an independent existence. Usage of one type by another is always by reference to that type; similarly, instances that have other features as properties or parts do not 'own' these features but merely refer to them.

3.1.4 Ad-hoc modelling

Additional flexibility is made available at the level of feature instances, by the capability of the model to deal with ad-hoc properties and relationships between instances. The properties and relationships that can be assigned to feature instances are not restricted to those defined by the corresponding feature types. The user can add any required property or relationship to an instance without prior modification of the type. Again, this reflects very well the actual way of working that is often encountered in design: typical solutions are used in design, but adaptations or additions are required to complete a particular design case. This can be done in the FBM framework without the need to find or create an exact typology first.

3.1.5 Implementation: object-model and meta-classes

The flexible and extendible structures of feature data, described in the previous paragraphs, are made possible in the FBM framework by the implementation of a set of meta-classes. These meta-classes define a run-time object model of feature types and feature instances. The user of the framework appears to be working with feature instances as instances of feature types, but the system implements both types and instances as objects of the meta-classes. This approach provides the flexibility that is necessary to allow users to define new types at run-time and to allow instances to deviate from the types. For details about the complete set of meta-classes, their purposes and capabilities, the reader is referred to (van Leeuwen, 1999).

The next paragraphs describe the characteristics of the DesKs technology that serve requirements of **design collaboration**.

3.1.6 Version management

An important issue in collaborative activities is how to control versions of information. Keeping track of versions of information serves three objectives: to record the history of information in order to allow undo-operations; to allow changes to data without compromising references to previous versions of that data; and to make it possible to inspect and compare versions.

Current practice document management systems provide version control, but only at the document level. For collaborative design, version control is required at a finer level of detail for a combination of reasons. The number of people working with design data is large, the total collection of design data is large, documents are not always the basis for storage, and perhaps most importantly, there are strong relationships between chunks of data, within documents or crossing the scope of documents.

The FBM framework has strong support for version control of both feature types and features instances. Editing of feature data (both types and instances) takes place via a checkout-and-commit mechanism, through which users get temporary editing privileges. While data is checked out for editing, previous versions can continue to be used. After editing, data can be either submitted as a new version, or committed as a revision. Revisions of feature data are inferior to versions in the sense

that they cannot yet be actively used in modelling operations, only for further editing of the data. This reduces the number of versions and allows distinction of which submissions are of real interest and which have only an intermediate status. Only revisions of the latest version are backed up by the system.

Versions are distinguished by the combination of a major version number *M* and a minor version number *n* in the form *M.n*. New version numbers are incrementally assigned upon submission and minor version numbers are reset to zero after the submission of a new major version. Submitting a version to the system can lead to a new major version or a new minor version. Minor versions indicate backwards compatibility, which means that the version can also be used in place of previous minor versions of the same major version. For example, adding a property to a feature type leads to a new minor version because it does not compromise the functionality of the type in places where the type without that property was expected. New major versions are not backwards compatible, meaning that they cannot be used in place of any preceding versions. Modifications such as removing properties or changing the type of properties will generally lead to new major versions. Whether a submission is a new major or minor version, is determined in the first place by the user. However, the system will enforce major versions when it detects backwards incompatibility. Upgrading in instance to a more recent minor version of its type is generally possible and can probably be done automatically, although this functionality has not yet been studied in detail. An incremented revision number is assigned after each time a revision is committed or a version is submitted; the revision number uniquely identifies a revision or version of the feature data.

3.1.7 Unique Identification

Feature types and feature instances must be uniquely defined. Uniqueness is necessary not only within their direct context, but in a worldwide scope. Enforcing this scope of uniqueness guarantees the ability of sharing types and instances in any possible situation. The mechanism adopted for providing uniqueness within this wide scope is that of *namespaces*, similar to the way namespaces are used in XML. In this application of the notion of namespaces, they are related to a URI for global identification (Uniform Resource Identifier); often a URL is used for this purpose. Once the uniqueness of a namespace is established, all unique names within the namespace are globally unique as well.

Full references to feature data in the FBM framework include the identifier of the namespace, the identifier of the feature data (type or instance) and the revision number of the data, which is a unique number for each version or revision.

3.1.8 Ownership, authentication and authorisation

Each individual feature type or feature instance is *owned* by an identifiable user. Users are identified by their email address and are authenticated using a password. Each initial access to an application of the FBM framework will require authentication. Users have full access rights to the features they own and can grant anonymous access or access rights restricted to other users or groups of users. Authorisation will take place automatically upon each access. Namespaces have

owners as well and can have restricted access. Access rights set for individual features in a namespace impose restrictions further to those that are set for the namespace as a whole.

Groups of users can be defined to represent teams in collaboration projects or to specify other kinds of group access to certain data. User can acquire access rights through membership of a group, but higher individual rights will not be restricted by such membership.

While the authorisation mechanism is still under development, the following levels of access rights are currently distinguished, listed in incremental order:

- Copy (read but only for copy, not for reference)
- Read (read but not instantiate)
- Instantiate (relevant for types only)
- Modify (change contents but not add)
- Add (add contents)
- Write (includes delete and rename)
- Ownership (includes the right to set access rights and to transfer ownership)

3.1.9 Access in a distributed environment

A previous implementation of the FBM framework supported access to remote data by offering the capability to download feature data from URL's. This approach only supported read-access to the remote data and thus solved only a small aspect of the collaboration problem. It did not support real-time collaboration in any way.

The current implementation of the framework supports direct remote access to data. Together with the mechanisms for authorisation and checking out data, this provides the means to collaborate in a distributed environment; distributed not only in terms of distributed users, but also in terms of distributed data. Users can access remote data as if it were local data, albeit that they are subject to the authorisation settings of the remote system.

Having the option to distribute data, project managers can now decide to leave the physical ownership of data where it belongs: with the experts that are responsible for it.

3.2 Applications and implementation

The FBM framework provides a two-layered object-model of feature types and feature instances, through the implementation of a third layer of meta-classes. The framework has been implemented such that access to remote data is possible as long as the data is available online. The DesKs project aims to deliver two applications (see Figure 1) that provide the framework's functionality to support collaborative design in two different ways:

1. **DesKs WebServer** application
   This application is targeted primarily at publishing or hosting design knowledge and can be accessed only remotely. The application runs in conjunction with a common web server and provides two kinds of access. The

first type of access is through HTML pages and is available for web browsers. Although it is probably possible to develop a web interface that provides almost full modelling functionality, this is not an objective of this type of access. The web interface will be restricted to browsing the feature data in the namespaces that are available through the server; adding or modifying data will not be made possible through this interface.
The second type of access is through Web Services. Web Services provide enhanced functionality by allowing clients to execute procedures at the server. A dedicated client will be necessary to access Web Services. However, the protocol of using Web Services is platform independent, meaning that clients can be made available for a wide range of operating systems.
The development of the DesKs WebServer application has not yet been started but is programmed for prototyping after testing of the DesKs WebNode application has been concluded satisfactorily.

2. **DesKs WebNode** application
   The WebNode application is targeted for usage by local users as well as remote users. Part of this application will function as a node in a peer-to-peer network. Each node can have multiple local users and each node has access to multiple other nodes in the network. Users can access the local data on the node, but remote data on other nodes as well. This part of the application is not based on web server technology, but it does use the common protocols HTTP and SOAP.
   Another part of this application functions as dedicated client to the DesKs WebServers. This enables the application to search and retrieve data from these servers and to work actively with the data on the servers.

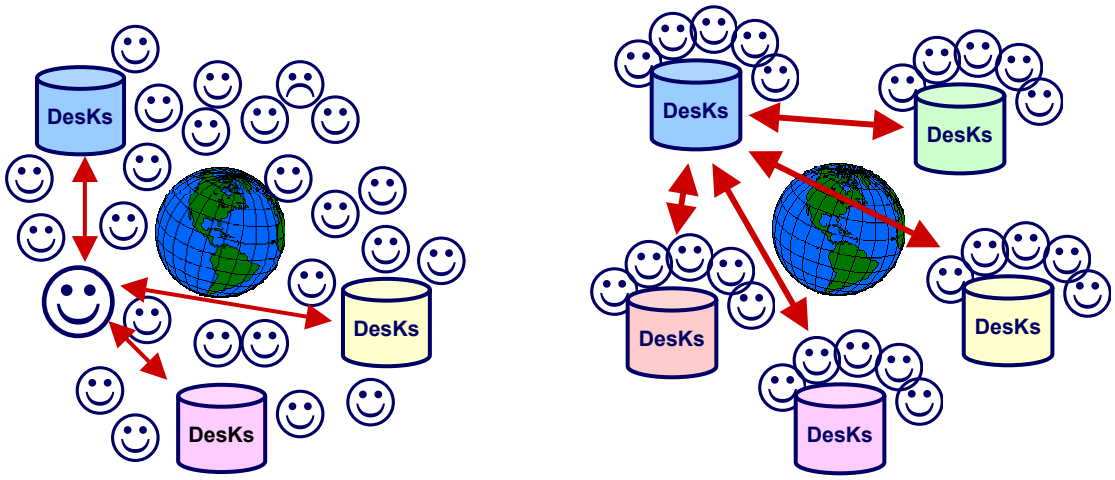

Figure 1. On the left: DesKs WebServer application for browser or dedicated client access. On the right: DesKs WebNode application for peer-to-peer networking.

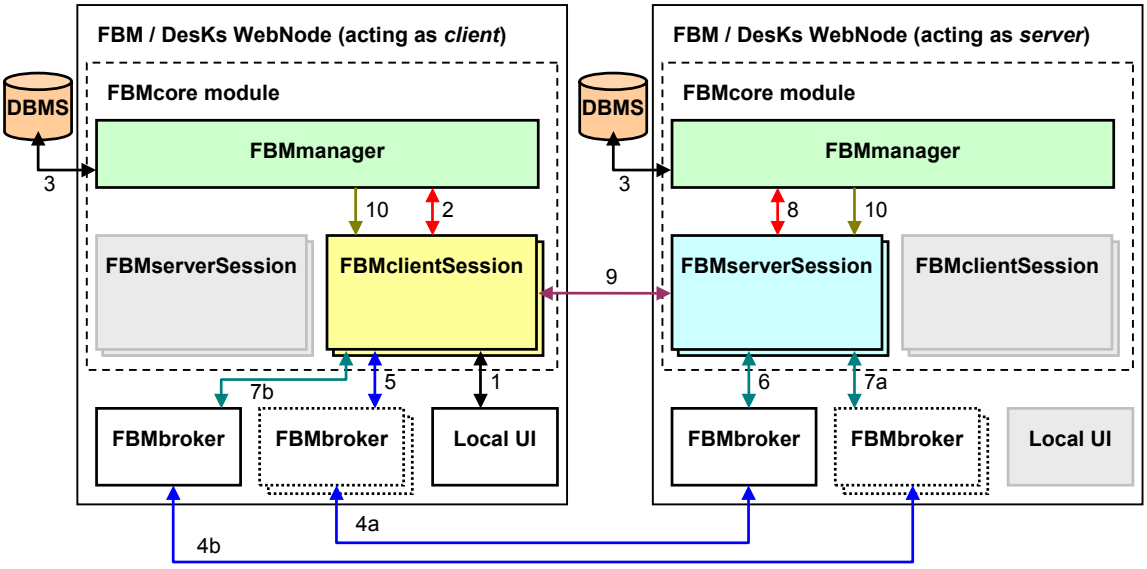

Figure 2. General architecture of the DesKs WebNode application. On the left an instance that is acting as client, on the right one that is acting as server (van Leeuwen and Fridqvist, 2002).

3.2.1 Implementation: .NET remoting

This section briefly addresses some aspects of the implementation of the DesKs technology. Both the FBM framework and the DesKs functionality are implemented using Microsoft's .NET framework and the C# programming language. The framework's object-model is implemented in the so-called FBMcore module. The object-model forms the programming interface to this module for the development of applications such as the DesKs WebNode and the DesKs WebServer.

The central part of the FBMcore module is the FBMmanager. This is a single object in each instance of the core module, which is responsible for the persistence and consistency of the feature data. In the current implementation, data is persisted into a relational database. However, for purposes of interfacing with other applications, XML import and export is foreseen. Feature types will be streamed from and to XML-Schema files, and feature instances will be streamed from and to XML documents. Both the schemas and the documents are validated by a generic FBM XML-Schema, while XML documents containing feature instance can also be validated using the Schemas of their respective feature types.

The remote access functionality of the DesKs applications is implemented using the .NET *remoting* facilities. Remoting provides access to objects on a remote server, as if these objects resided in the memory of the client.

Figure 2 shows a simplified overview of the architecture of the DesKs WebNode application, with the FBMcore module in place. Two running instances of this application are shown; on the left an instance that acts as a client; on the right one that acts as a server. In reality, each instance of the WebNode application can play both roles simultaneously and for multiple local and remote users. Local users work with local data through a *client session* on the local *manager* (lines 1, 2, and 3). Remote users need to establish a connection to a server (line 4a), which is done through a so-called *broker* object. The client creates a *proxy* (dotted rectangles) for the server's broker, which then allows it to instruct that broker (line 5), for example to log onto a *server session* on the server's manager (lines 6 and 8). The server establishes a reverse connection to each client (line 4b) for the purpose of sending notifications and instructions. In this multi-user environment, users need to be notified of modifications to data. This is initiated by the manager that is responsible for the modified data (lines 10) and communicated directly to the local users (line 1) and via the server sessions to remote users (lines 7a and 7b). Once the connections between client and server have been established, the broker is no longer necessary for each communication. Proxies exist on the client for the server-side objects, including sessions, namespaces, and feature objects, which allow the client to have direct access to these objects at the server (line 9).

More details about this architecture and the ways of communication can be found in (van Leeuwen and Fridqvist, 2002).

3.3 Usage scenarios

The DesKs technology can be used in a variety of scenarios. It is developed mainly for the support of collaborative design. Participants in a collaborative design project can maintain their own WebServer or WebNode applications to publish or share design knowledge related to the particular project. Additionally, generic design knowledge can be made available in a wide area network, ready for use in design projects.

Being able to publish generic design knowledge in a formal format that can be used remotely reveals a number of new possibilities of which publication of construction product data is perhaps one of the most feasible. Information concerning new products or modifications of products can be made available instantly in a ready-to-design format. Accessing the latest version of information concerning products is now automatic.

DesKs technology also makes it possible to offer remote design services. Using a DesKs network, suppliers of design services can publish portfolios or generic solutions, which can be accessed remotely by their clients. Clients can acquire access rights to re-use existing solutions or to retrieve custom-made designs. Alternatively, clients can share their design problem with the suppliers of design knowledge, giving them access rights to the 'problem' server, so that design services can be delivered remotely at that server.

4. NEXT LITTLE STEPS

The FTR project, mentioned before in this paper, investigates and develops the potentials of matching models and features types. This is one area of application of artificial intelligence in the context of the DesKs technology. FTR will open the way for many applications of Case-Based Reasoning. It can aid the designer in finding suitable products for a design or in finding similar design problems and the solutions that exist for those problems. It helps the designer to re-use his own or other designers’ knowledge, by matching the current state of a design model to the body of design knowledge that is formally represented by feature types.

The FTR project is currently in its first phase of testing (Fridqvist and van Leeuwen, 2002). Once these methods for feature type recognition are stable, they will be integrated in the DesKs applications.

Integration of the FBM modelling approach in other lines of research on design support systems may well enhance both the DesKs environment and these various research activities. Two examples of candidates for integration are the E3DAD project that studies associative reasoning in design, where the computer supports designers by providing information that can be associated to the current design activities (Segers et al., 2001), and the DDDoolz project that builds a VR interface to three-dimensional sketch operations (de Vries et al., 2001).

5. WHY IT TAKES SO LONG…

Why does it take so long before we can publish results like the invented quotation at the beginning of this paper?
There are two main reasons:

Much increased levels of Artificial Intelligence will be required to arrive at something that we could call ‘comprehension-logic’. Since many difficulties in developing such technology can be foreseen, it will probably take much longer even before computers can start to show behaviour that one could call ‘understanding’. Many of these difficulties are related to the incoherent ways we human beings think and express ourselves, such as the languages we use.

Perhaps a cliché, but true nevertheless: the construction industry will have to adopt new technologies like this and will need to accept major changes in their business processes. Formalisation of knowledge will have many consequences, for example, on the way we set up design and construction teams and make contracts. Issues such as ownership, copyright, and the responsibility for the application of design knowledge need to be formalised as well.

ACKNOWLEDGEMENTS

The DesKs research project has been carried out with financial support from Eindhoven University of Technology and the Portuguese Fundação para a Ciência e a Tecnologia. It was hosted at the Instituto Superior Técnico, Lisbon, in collaboration

with Prof. João Bento. The project has strong relations with the parallel project on Feature Type Recognition, which is under development by Sverker Fridqvist at Eindhoven University of Technology.

REFERENCES

Amor, R. and I. Faraj (2001) "Misconceptions about Integrated Project Databases." *Electronic Journal of Information Technology in Construction* (www.ITcon.org), Vol. 6 (2001) pg. 57-67.

B. de Vries, A.J. Jessurun, and J.J. van Wijk. "Interactive 3D Modeling in the Inception Phase of Architectural Design." *Eurographics Short Presentations*, Manchester, United Kingdom, September 4 – 7, 2001, pp.265 – 271.

Fridqvist, S. and J.P. van Leeuwen (2002) "Feature Type Recognition – implementation of a recognizing feature manager." *Proceedings of the CIB W78 conference Distributing Knowledge in Building*, June 12-14, 2002, Aarhus, Denmark.

Segers, N.M., B. de Vries, H. H. Achten, H.J.P. Timmermans (2001) "Towards Computer-Aided Support of Associative Reasoning in the Early Phase of Architectural Design." *Proceedings of CAADRIA 2001*, April 19 – 21, 2001, Sydney, Australia.

van Leeuwen, J.P. (1999) *Modelling Architectural Design Information by Features*, PhD. Thesis, June 1999, Eindhoven University of Technology, Eindhoven, NL.

van Leeuwen, J.P., Hendricx, A., and Fridqvist, S. (2001) "Towards Dynamic Information Modelling in Architectural Design." *Proceedings of the CIB-W78 International Conference IT in Construction in Africa 2001*, CSIR, Division of Building and Construction Technology, May 29 – June 1, 2001, Mpumalanga, South Africa, pp 19.1-14.

van Leeuwen, J.P. and S. Fridqvist (2002) "On the Management of Sharing Design Knowledge." *Proceedings of the CIB W78 conference Distributing Knowledge in Building*, June 12-14, 2002, Aarhus, Denmark.

Unknown, Y.E.T. (2028) "A Survey of Enhanced Comprehension Logic in Design and Engineering." To be published in: *Proceedings of the 19th International Conference on Design and Decision Support Systems*, Amsterdam, July 9 – 12, 2028.

URL'S

URL-1: http://iaiweb.lbl.gov/

URL-2: http://www.designknowledge.info/