Published as: van Leeuwen, Jos P., and S. Fridqvist. 2002. “On the Management of Sharing Design Knowledge.”
In: *Distributing Knowledge in Building*, Proceedings of CIB W78 2002, Aarhus, Denmark: 12 – 14 June 2002.

Title: **On the Management of Sharing Design Knowledge**

Authors: Jos P. van Leeuwen and Sverker Fridqvist
Institution: Eindhoven University of Technology,
Department of Architecture, Building, and Planning
Eindhoven, The Netherlands
E-mails: J.P.v.Leeuwen@tue.nl
S.Fridqvist@tue.nl

Abstract: *This paper presents the approach and results of a research project that develops an environment for sharing design knowledge. The project implements a design modelling approach that allows designers to capture design concepts into formally defined typologies. With these, designers can build flexible design models and have full control over definitions and structures used to represent design concepts.*

The presented research project uses this dynamic modelling approach in the development of Design Knowledge Servers that function in a network to provide a distributed multi-user environment for sharing design knowledge. Such a network serves the requirements of collaborative design and is useful for other purposes, such as publication of formalised product datasheets.

The paper briefly provides the background of the knowledge modelling approach underlying this project and describes the design and implementation issues of the Design Knowledge Servers and the services they provide.

Keywords: *Collaborative Design, Design Information Modelling, Design Knowledge Servers, Case-Based Reasoning, Internet Technology*

Sharing Design Knowledge for Collaborative Design

Collaboration in design is one of the key factors of successful design in building and construction. It is necessary in virtually all stages of design and involves a great variety of disciplines and partners in the project. This includes the multiple design partners who each bring in their own domain-specific knowledge, the client and users of the future building, contractors and suppliers in the building and construction industry, and prescribing and regulating authorities.

Probably the most important aspect of collaboration is sharing knowledge. *Design knowledge* is the totality of on the one hand information about a design, including the data that describes the design and the contextual meaning of this data, and on the other hand information about how a design is achieved and evaluated. We can distinguish design knowledge into the kind of knowledge that is particular to a design project and the kind of knowledge that is used in projects but is of a generic nature.

A third kind of design knowledge is design cases. Knowledge from previous designs is continuously built up, consciously or unconsciously, in the minds of designers and applied, again consciously or unconsciously, in new design projects.

Digital media have greatly improved the efficacy of collaborative design, by reducing the effort and faultiness of communicating design information. Despite that, prevailing ways of exchanging digital design information are often semantically poor or semantically incorrect and lead to mistakes in interpretation, by humans and by computer-systems. However, digital media promise more efficient means to express and communicate design knowledge. An increased semantic level of design data may help to reduce faults and therefore will increase the efficiency of the collaborative design process.

Formalised Design Knowledge

One way to increase the semantic level of design data is to develop semantically more detailed and more explicit standards for data exchange. The Industry Foundation Classes (URL 1), under development by the International Alliance for Interoperability, have a great potential to become the de facto standard for data exchange exactly because they will bring the data exchange to a higher semantic level.

Innovative designs, however, incite the need for expressing novel concepts that cannot be described using the standards. In these cases, designers may feel limited by standards since they only provide means to express design intentions on a generic level that cannot catch much of the specific intentions. To fully support design, modelling tools need to allow design concepts to be defined that exactly represent the rationale of the design (van Leeuwen, 1999). This would allow a design support system to be tailored for a particular designer.

To formally define design concepts is a form of knowledge modelling, since the designer expresses not only the actual design case, but also the concepts used in the design. The concepts represent the body of design knowledge that was used to come to the particular design solution, and they can be reused for other designs.

Formalised building product information is another way to enhance the semantic levels of design data. Although product information is increasingly often made available digitally, the format is mostly ill structured, such as a web page with text and images that can only be interpreted by human readers. To search this kind of format is very time consuming but can be greatly enhanced if the information structure allows automated searches (Bakis and Sun, 2000). Once product information is made available in a structured way that allows computers to interpret the content, it can form a much more valuable source for design support systems in providing intelligent feedback and suggestions to the designer (Augenbroe, 1998). Again, standardised models will play an important, but limited role in the formalisation of product related design knowledge. The role is limited for two reasons: firstly, standards, in the way they are currently developed, cannot be expected to both reach sufficient level of detail and to remain sufficiently generic for the required general applicability that they are developed for. Secondly, new products, materials, and construction methods will continuously appear, which will require additions to any but the most generic standards.

Dynamic Feature-Based Modelling

The system for sharing design knowledge that is described in this paper, is based on a dynamic approach to product modelling that is inspired by *Feature-Based Modelling* (FBM) in mechanical and structural engineering. The theory for this approach in a generic architectural context, called the *FBM framework*, has been described in (van Leeuwen, 1999) and can be characterised as a property-oriented modelling approach (van Leeuwen et al., 2001). Following are the main issues addressed by the FBM framework:

1. **Property oriented**
   Rather than defining object-classes to represent building components that embed many properties, this approach takes the properties themselves as a starting point for modelling and allows the designer to compose the object-representations from properties. In fact, the FBM framework has no need to distinguish objects from properties, calling them both features. It allows a feature to be used with different roles: as a key modelling object in one situation and as a property of another feature in another situation. For example, a feature ‘Spatial function’ can at one stage in the design process be a key object and at a later stage be changed into a property of the ‘Space’ feature where this function is performed.
2. **Flexibility in modelling**
   Properties have an independent existence, independent of the object that carries the property, and they can be shared by multiple objects. The existence of a property such as ‘load-bearing’ is not dependent of the building component that performs this task. This not only allows the designer to describe the design rationale in a natural way, it also better supports the process of design, because it can deal with the often inconsistent and ‘composing’ nature of design. The load-bearing building component might be removed from the model while the property ‘load-bearing’ is still required and needs to be transferred to another component that might be added to the model at a later stage.
3. **Ad-hoc modelling**
   The FBM framework does not constrict the modelling of properties of features and relationships between them to those that are defined in feature types. Features in the model can be changed as needed, without prior adjustment of feature types. For instance, if the context of a building design requires a ‘door’ to have the property ‘fire-resistance’, but this property is not defined for the door-type, then the designer still can add the property to the model, without having to modify

the door-type. This approach reflects the actual situation that is often encountered in design, where specific features are added to typical design solutions, or where components are used in unforeseen ways.

4. **Designer-defined typologies**
   In addition to this flexibility of building relationships between objects and between objects and properties, designers can also define new typologies, i.e. new feature types. This capability allows designers to formalise design concepts that result from a particular design case, or that represent general aspects of their design methodology. These custom-defined feature types build up a library of concepts that represent the designer’s specific knowledge. The extendibility of the conceptual model also serves the need to define new typologies, e.g., for new construction products or construction methods.

The FBM framework includes simple data types and complex data structures, but also constraints and procedural types. Although the latter two have not been implemented fully yet, they do form an integral part of the framework. The framework implements the property-oriented system and its flexibility and extendibility by defining an object model with a set of meta-classes for both feature types and feature instances. The system’s end-user appears to deal with feature instances as objects that are instantiated from the feature types, but in fact, both feature instances and feature types are objects that are instantiated from the meta-classes. This way, the system provides run-time extendibility of the schema, while keeping all objects consistent with the meta-classes and maintaining the relationships between instances and types. In addition, the ad-hoc modelling capability, which allows deviations of feature instances from their types, is provided through the meta-classes.

When a feature model is changed as mentioned in item 3 above, the feature instances may no longer match their original types. A process called *Feature Type Recognition* (FTR) can rectify this by finding matches between patterns in the feature model and existing feature types and suggest new assignments. It can also assist the designer by suggesting replacements in or additions to the model based on the knowledge available from these feature types. Feature Type Recognition is under development in a parallel project that is described in (Fridqvist and van Leeuwen, 2002).

Design Knowledge Servers (DesKs)

The functionality of the feature-based modelling approach is used to build so-called *Design Knowledge Servers* (DesKs). The DesKs project manifests an Internet-based client-server technology that is to be applied in a number of scenarios of sharing distributed design knowledge.

Targeted Applications and Usage Scenarios

Design Knowledge Servers function as a network of interrelated servers that provide managed access to distributed data. Such a network can be used inside and outside the scope of a design project. Inside a project-scope, each node in the network represents a repository of design knowledge that falls under the responsibility of the owner of the node, e.g. a structural engineer or an architect.

Outside the scope of any particular design project, the DesKs can be used to provide access to ‘general’ design knowledge and public design information, such as product data, information about construction services, design methods, public evaluation tools, etc.

Below, four example scenarios are described, in which the DesKs can be used to share design knowledge.

Collaborative Design Projects

The Design Knowledge Servers can be used in this scenario to develop design solutions by gradually building up formal definitions for design concepts (specific feature types) in combination with the application of more standard design concepts (generic feature types). The project’s design database, containing both feature types and feature instances, does not need to be centralised but can be distributed over the network of systems managed by the collaborating partners. This way, the participants maintain their ownership and responsibility of the knowledge and data involved in their design task. Other participants are granted access to design data based on their role in the design team, on contracts, and on shared design tasks and responsibilities. Knowledge that is external to the design project, in the form of modelling standards or automated code-checking procedures, can be included in the DesKs network if the communication and data exchange protocols are supported.

Design Knowledge Commerce

In this scenario, there is a contract between a party needing a design solution to a particular problem and a party able to provide the solution. The parties have been brought together through a search session provided by one or multiple DesKs, where the searching party has input the design problem into a search algorithm and the providing party had made the design solution, or information about how to achieve a solution, available on a server in the network. Based on the contract between the parties, rights are assigned to the acquiring party to use the provided design knowledge.

This commercial scenario involves many complications in terms of copyrights, multiple uses of the provided knowledge, re-use of knowledge provided by third parties, composite solutions, etc. These complications need to be addressed before this scenario can actually function in a commercial setting. However, a similar scenario can be envisioned in the context of, for example, designers’ associations. Designers subscribe to the provided services, acquiring the right to use all knowledge available while feeding the association’s servers with their own knowledge.

(Historical) Design Reviews

The ability for design critics and architectural historians to make their observations of design rationales in historic or new buildings available in a formalised manner would allow them to share this knowledge with the architectural design discipline. This would offer a tool for very direct feedback into today’s design practice. It would also support comparisons of design approaches and alternative design solutions.

Product Databank

Providing product data in a ready-to-use format is a very feasible application area for computer support in design. The construction industry has already seen several successful developments in this area. For example, the Dutch building-documentation system (NBD) provides this kind of information in a digital format, although this format often is too loosely defined to be utilised directly in computer supported design tasks. The international developments on standardisation of Part Libraries (ISO 13584) aims to provide a more detailed definition of how part information is to be provided. The part Libraries deal with structural, descriptive, and procedural knowledge in electronic catalogues to serve the selection, evaluation, and representation of parts in engineering tasks (Pierra, 1997).

The scenario of applying DesKs for providing product data to the building design community follows a similar path: manufacturers and suppliers can provide their own specific typologies and samples of products and materials in feature types and instances. Product information made available through Design Knowledge Servers can be used directly in the design process.

DesKs System Design

Targeting the application scenarios above, a system has been designed that provides the FBM framework functionality through an object-model that supports remote access to distributed data sources. The FBM framework implementation will be used in the development of two applications:

1. The **DesKs WebServer application** allows publication of design knowledge through a common web server and makes the design knowledge available in two ways: as HTML web pages and as a Web Service. HTML access to the knowledge is limited and provides only a browsing type of digestion of the provided information. The Web Service access is more enhanced and allows client applications to consume search, storage, and retrieval services.
2. The **DesKs WebNode application** is a modelling tool that provides all the FBM capabilities of formalising concepts and dynamically modelling designs. The application also provides dedicated client functionality to access the Web Services provided by the DesKs WebServer application. This enables the application to search and retrieve remote design knowledge and to actively work with distributed design data.
   In addition, the WebNode application provides peer-to-peer functionality that enables it to share its data with other WebNode instances in a wide area network. This part of the application is not based on common web server technology, but does use the common protocols HTTP and SOAP.

Object-model and meta-layer

As mentioned above, the FBM framework is implemented as an object-model that provides a meta-layer of classes defining the kinds of feature types and the kinds of feature instances that can be modelled. The

run-time system thus contains three layers of data: the bottom layer that defines the actual design data in terms of feature instances, the middle layer that defines the typologies used for creating the instances, and the meta-layer at the top that defines the ‘syntax’ for the lower two layers.

The meta-layer includes classes for simple data, such as strings and floating point numbers, and complex data structures. Complex data structures are defined as features that contain *components*. At the type-level, a component of a complex feature type is a reference to another feature type with information about the role and occurrence of that type in the context of the complex type. At the instance-level, a component references one or several other instances. The instance-level component can either be instantiated from a component defined by the type of the instance, or it can be a ‘custom’ component that is added ad-hoc to the feature instance.

Components are specified by a role-name and a role-type. The role-type distinguishes associations between features, decompositions, and specifications; the latter being a particular kind of association that tells us that one feature specifies a characteristic of another. These role-types add semantic meaning to the model that is used in the reasoning support procedures that are under development in the parallel project on FTR described in (Fridqvist and van Leeuwen, 2002).

Version management

A very important aspect of collaborative activities is information version control. Implementing versioning has three objectives: to support undo operations, to allow changes to data without compromising references to previous versions of that data, and to make it possible to compare versions. Current practice document management systems provide good support for version control at the document level. In collaborative design, however, version control is needed at a finer level of detail, because of the large number of participants working on a design and the strong relationships between objects within one or separate documents.

In the FBM framework, design information is organised by way of *namespaces* rather than documents (see next section). Documents can be generated upon selections of objects from one or more namespaces, but documents are not the basis for structuring the information. The FBM framework has a strong support for version control, at the level of both feature types and feature instances. Through a *checkout* and *commit* mechanism, users can acquire temporary editing privileges for features. During editing, previous versions can continue to be used by other users. Users can decide to submit a new *version* to the system, or to commit a *revision*, which is a not-finalised version that cannot yet be referenced.

Versions are identified by a major version number and a minor version number. Minor version numbers indicate backwards-compatible changes, while major version numbers indicate non backwards-compatible changes. Whether a modification is backward compatible or not is in the first place decided by the user, but checked by the system; certain modifications, such as changing the role-type of a component, will always lead to new major versions, while other modifications, such as adding a component, may lead to new major versions if the user decides so.

Identification and namespaces

In the original definition of the FBM framework (van Leeuwen, 1999), feature types were located in *feature type libraries*, while feature instances that represented a design case were collected in *feature models*. Libraries and models were defined as single storage locations (e.g. a single database or single data file). In the current implementation of the FBM framework, libraries and models are replaced by *namespaces*, in accordance with the usage of namespaces in XML. Namespaces have the advantage that data can be distributed over multiple resources (although a mechanism for retrieval of all parts must be available) and namespaces can be related to a URI reference (Uniform Resource Identifier), such as a URL, giving globally unique identification to all names that are unique within the namespace.

The notion of namespaces is applied in the current FBM framework to identify collections of features, both types and instances, that together form a particular body of design knowledge, or that represent a particular design case.

Ownership, authentication and authorisation

Each individual feature is *owned* by an identifiable user of the system. Users are identified by their email address and are authenticated using a password. Each server from which a user requires services will ask for user authentication unless the required services are allowed for anonymous access. Owners have full

access to their own features and can grant selective access rights to other users or groups of users. Similarly, namespaces have an owner that can grant access rights to the namespace. Access rights set for individual features in a namespace impose restrictions further to those that are set for the namespace as a whole.

For the purpose of collaboration, teams of users can be formed. Access rights can be acquired through membership of a team, but restricted team access rights do not suppress individual access rights. Users can be member of multiple teams with various roles within teams and within the context of namespaces.

Implementation

The implementation of the FBM framework and the Design Knowledge Servers is made using Microsoft’s .NET framework and the C# programming language. The functionality of the FBM framework is implemented into a core module to which the DesKs applications are connected.

The framework’s object-model forms the programming interface to the core module for the development of applications. Internally, the object-model is persisted into a relational database (current testing uses MS-SQL server). For communication with other applications, an important feature of the core module is the import/export capability from and to XML documents. Feature types can be streamed from and to XML-Schema’s, and feature instances can be streamed from and to XML documents. Both schema’s and documents are validated by a generic XML-Schema that represents the syntax of the FBM framework. The XML documents containing the feature instances are also validated by the XML-Schema’s that contain the respective feature types.

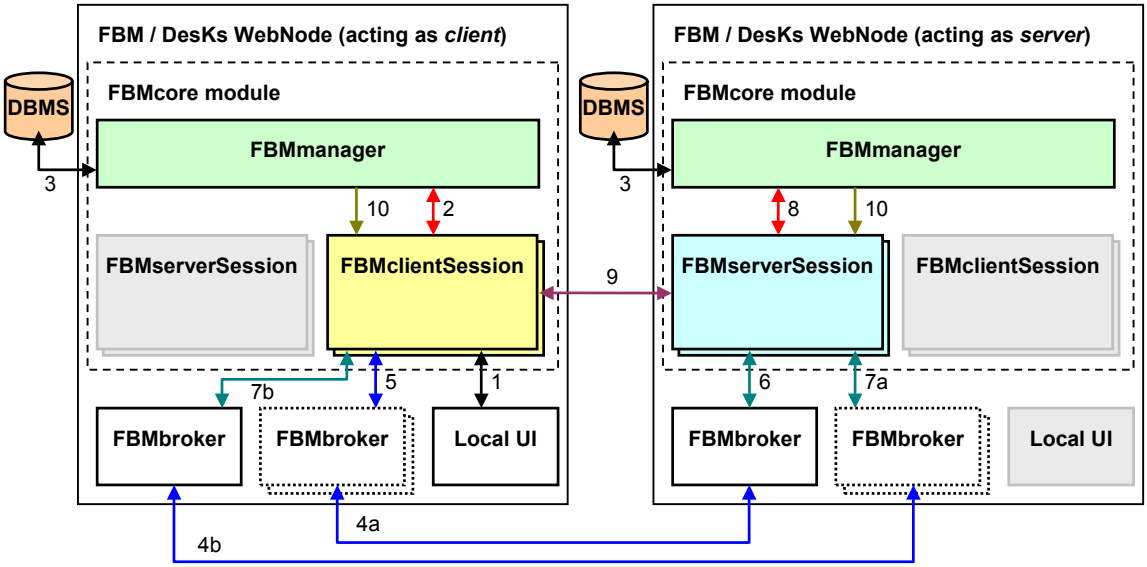

Figure 1. General architecture of the DesKs WebNode application. On the left an instance that is acting as client, on the right one that is acting as server.

Networking DesKs

The peer-to-peer functionality of the DesKs WebNode application is built using the .NET framework’s *remoting* facilities. Simply put, remoting allows objects on a server to be accessed by remote clients, as if they resided in the local memory of the client. A WebNode client can have open connections with multiple servers, which allows the simultaneous utilisation and combination of feature data from various resources. Vice versa, servers allow multi-user access and provide functionality for sharing sessions on a server. Sharing sessions allows teams to work together with the same set of namespaces and features retrieved from servers. A subscription mechanism notifies client applications about changes at the server, to avoid problems with outdated information at the remote clients.

Figure 1 shows a general picture of the architecture of the DesKs system as it is currently implemented in the prototype WebNode application. The figure also indicates, in a simplified manner, the communication lines between the various parts of the system, as listed below.

1. The FBMcore module is prepared for multi-user access, either local or remote. Each local user of the application communicates with a private client-session object.
2. Client-session objects communicate with the single manager object in the application, which provides the object-model of the FBM framework, including objects for namespaces, feature types, and feature instances.
3. The manager instantly persists all modifications through an OLE-DB interface.
4. The initial contact (4a) with remote servers is made through a broker object that exists at the remote server and for which a proxy is created at the client-side (proxies have dotted outlines in the figure). Each instance of the application runs a single broker object, but can maintain proxies for multiple server-brokers. This connection is two-way: the server establishes a connection back to the client (4b) using a proxy for the client’s broker object. This second connection is used for communications initiated by the server[1] (see also item 7 below).

5/6. Once the connection with the server is established, the client-session can retrieve information about the server via the broker-proxy (5). For each remote client, the broker at the server creates a server-side session (6) with which the client can communicate as if the server-side session were a client-session to locally managed data (see also item 9).

7. When a client establishes a connection to a server, the server creates a server-side proxy for the broker of the client in order to push data back to the client on its own initiative. Via the server-side proxy (7a), the server can get access to the client-session (7b). This way the server can notify the client to retrieve updates for feature data to which the remote user has subscribed.
8. The server-sessions communicate with the server-side manager in the same way as client-sessions communicate with the local manager (see item 2).
9. After the brokers have been used to establish the connections between client and server, and once the client has connected to its server-side session and vice versa, proxies will exist on both sides for the remote session objects. The broker is no longer needed for communication that is initiated by the client. Remote feature retrieval and editing activities related to remote features will now be executed directly between client-session and server-session, using proxies for remote sessions and for feature data that resides in the server-session.
10. Users need to be notified of modifications of data made by other users. This is done through a subscription mechanism that registers users’ interests in certain data on a per session basis. Upon notification by the manager, client-sessions inform the local UI and server-sessions inform remote client-sessions of the modification events.

Conclusions and Future Work

This paper described how Design Knowledge Servers (DesKs) can be used in a wide area network environment to support collaborative design. Using DesKs technology designers can take advantage of the dynamic product modelling approach of the Feature-Based Modelling (FBM) framework, a property-oriented modelling approach in which designers can formalise design concepts and use these formalisations in flexible design modelling. The DesKs technology can be applied in various scenarios, including collaborative design projects, commercially delivering online design services, building up case-bases of contemporary and historical designs, and publishing construction product information in a format with a high level of semantics.

The prototypes developed for the FBMcore module and the DesKs WebNode application have given us sufficient proof of the feasibility of developing this kind of client-server system for the support of collaborative design. The feasibility of applying the DesKs technology into the practice of collaborative design has not yet been proven sufficiently and requires significant test cases with experts from industry. These are planned to be conducted in collaboration with industry partners once a sufficiently user-friendly UI has been developed for the prototype systems. Future work also includes collaboration with the supply

[1] The reverse communication from server to client could also have been implemented using remote event handling. However, experiments have demonstrated that remote event handling does not function properly in all WAN configurations.

chain in the construction industry to investigate the feasibility of using the DesKs technology in formalising the information resources in the supply chain.

After testing the current prototype applications, the DesKs WebServer application will be developed to provide the FBM framework as Web Services through common web server technology. The DesKs WebNode application is designed to integrate the outcomes of this research project with the results of the Feature Type Recognition (FTR) research project, described in (Fridqvist and van Leeuwen, 2002). This involves using the matching algorithms in FTR to search WebServers and WebNodes for design concepts and design solutions that are useful for solving the designer’s actual design problem.

The project’s website is publicly accessible and will be used to distribute the prototypes for public testing and feedback (URL 2).

Acknowledgements

This 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.

References

Augenbroe, G. (1998) *Building Product Information Technology*. Executive white paper, Construction Research Center, Georgia Institute of Technology, 1998.

Bakis, N. and M. Sun (2000). Intelligent broker for collaborative search and retrieval of construction information on the WWW. In: Gudnason, G. (ed.), *Proceedings of CIT2000, International Conference on Construction Information Technology*, June 28-30, 2000, Reykjavik, Iceland.

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

Jain, S. and G. Augenbroe (2000). The Role of Electronic Product Data Catalogues in Design Management. In: *Proceedings of the CIB W96 Conference on Design Management in the Architectural and Engineering Office*, May 19-20, 2000, Atlanta, Georgia, USA.

van Leeuwen, J.P. (1999) *Modelling architectural design information by features*. PhD thesis, Eindhoven University of Technology, The Netherlands.

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

Pierra, G. (1997). Intelligent electronic component catalogues for engineering and manufacturing. In: *Proceedings of the International Symposium on Global Engineering Networking GEN’97*, April 23-24, 1997, Antwerp, Belgium.

URL’s

1. http://www.iai-international.org
2. http://www.designknowledge.info