Published as: van Leeuwen, J.P. and S. Fridqvist (2002) Design Knowledge Sharing through Internet Application. In: *Proceedings of the Concurrent Engineering Conference 2002*, Cranfield, UK, July 27-31, 2002.

Design Knowledge Sharing through Internet Application

J.P. van Leeuwen & S. Fridqvist
Eindhoven University of Technology, Eindhoven, The Netherlands

ABSTRACT: This paper presents the intermediate results of a research project that aims to develop and apply Design Knowledge Servers (DesKs) in the building and construction industry. This project builds on a modelling approach called Feature-Based Modelling (FBM) that is developed to provide a dynamic environment for handling design information. The Design Knowledge Servers apply this modelling approach 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.

1 SHARING DESIGN KNOWLEDGE

Sharing design knowledge is a prerequisite for collaboration in design, which is one of the key factors of successful design in building and construction. Collaboration with a great variety of disciplines and partners is necessary in almost all stages of a design project and involves the exchange of data describing the design, but also exchanging the rationale of the design.

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. This includes both generic knowledge from design theories and, for example, knowledge about construction products and materials.

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.

While digital media have already greatly improved the efficacy of collaborative design, the prevailing ways of exchanging digital design information are often semantically poor or incorrect, still leading to mistakes in interpretation, by humans and by computer-systems. An increased semantic level of design data may help to reduce faults and therefore will increase the efficiency of the collaborative design process.

2 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 design, however, incites 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 provide computer-support for design reasoning, 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. The concepts represent the body of design knowledge that was used to arrive at 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 & 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 role in the formalisation of product related design knowledge. However, this 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.

3 DESIGN KNOWLEDGE SERVERS (DesKs)

This paper describes the development in a research project that is called *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 his design knowledge with others and to use the knowledge they provide. This capability makes it possible to use DesKs to manage collaborative design projects, but also to publish design information, or to share 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). In abstracted terms, it will lead to the definition of algorithms and the development of prototype tools that allow the computer to find correspondences between the outcome of design activities and 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 4). 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 *Feature-Based Modelling in Architecture and Construction*

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.

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

3.2 *Property oriented modelling*

The FBM framework for modelling is property-oriented, meaning that it takes 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 (van Leeuwen et al., 2001). 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 is better able to follow the dynamic way that information is dealt with during design, compared to 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.

3.3 *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 the actual design. In product modelling terms: feature types define schemas for models of feature instances. 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 while still keeping their models accessible for others. 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.

3.4 *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. Not only will it be available for re-assignment, more importantly it reflects the continued significance of the spatial function in the design. 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.

3.5 *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.

3.6 *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**.

3.7 *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 DesKs technology 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 component 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.

3.8 *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.

3.9 *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)

3.10 *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.

4 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:

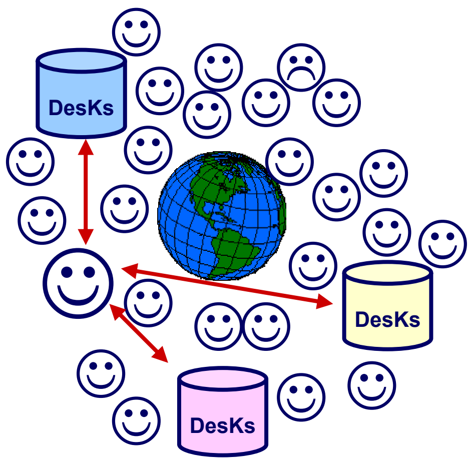
1. **DesKs WebServer** application
   This application is targeted primarily at publishing or hosting design knowledge and can be accessed only remotely. The application runs in conjunction with a common web server and provides two kinds of access. The first type of access is through HTML pages and is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

5 IMPLEMENTATION

The implementation of the FBM framework and the Design Knowledge Servers is made using Microsoft's .NET framework and the C# programming language (pronounced 'dot-net' and 'c-sharp', respectively). The functionality of the FBM framework is implemented into a core module to which the DesKs applications are connected.

The FBM 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



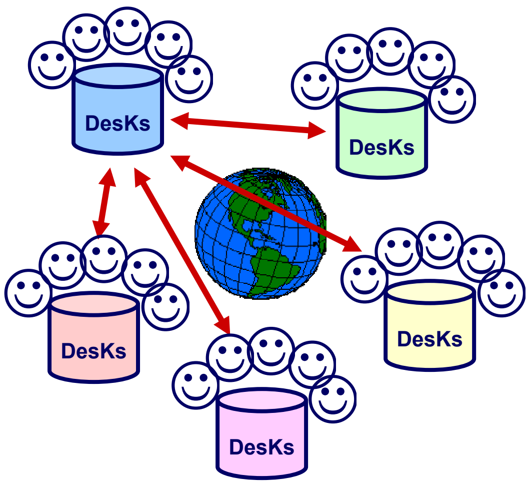


*Figure 1. On the left: DesKs WebServer application for browser or dedicated client access.
On the right: DesKs WebNode application for peer-to-peer networking (van Leeuwen, 2002).*

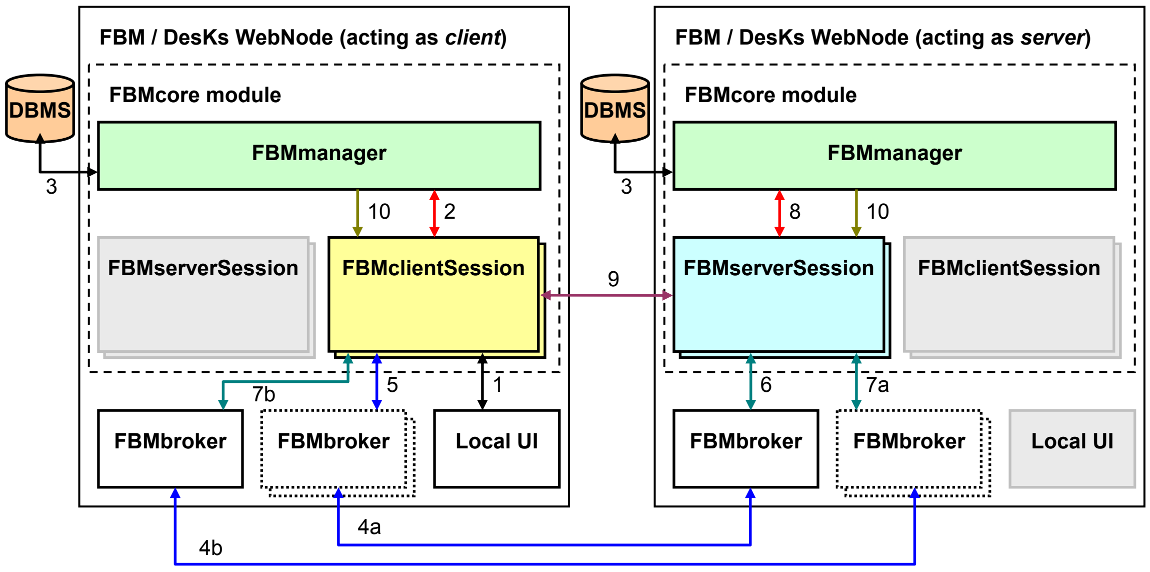


Figure 2. General architecture of the DesKs WebNode application. On the left an instance that is acting as client, on the right one that is acting as server (van Leeuwen and Fridqvist, 2002).

framework. The XML documents containing the feature instances are also validated by the XML-Schema's that contain the respective feature types.

5.1 *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 2 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-

[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.

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.

6 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 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. 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 still in an explorative phase of implementing, as is the DesKs WebNode prototype. Once both developments have become stable, the FTR functionality will be integrated into the DesKs WebNode and the DesKs WebServer 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).

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

7 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.

8 REFERENCES

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

Bakis, N. & Sun, M. 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.

de Vries, B., Jessurun, A.J., and van Wijk, J.J. 2001. 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. & van Leeuwen, J.P. 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.

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., & 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.

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

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.

URL'S

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