KNOWLEDGE-BASED PRODUCT DATA SERVER FOR CONCURRENT ENGINEERING

Knowledge-based product data server

P. KATRANUSCHKOV and R. J. SCHERER
Institute of Applied Informatics in Civil Engineering, Dresden Univ. of Technology, Dresden, Germany

Durability of Building Materials and Components 8. (1999) *Edited by M.A. Lacasse and D.J. Vanier*. Institute for Research in Construction, Ottawa ON, K1A 0R6, Canada, pp. 2189-2198.
© **National Research Council Canada 1999**

Abstract
Many requirements to product data management for concurrent engineering can be most efficiently satisfied through the application of advanced client/server architectures together with appropriate knowledge-based methods. This paper outlines the basic design considerations for a knowledge-based product data server and suggests a development approach on the example of a prototype server implementation realised in the domain of architecture, engineering and construction (AEC).

Keywords: concurrent engineering, knowledge-based system, product data management

1 Introduction

State-of-the-art product data management (PDM) and CAE software for AEC involve sophisticated problem solving algorithms as well as high-level database management functionality. With the emerging IFC project model (IAI, 1997), supported by most established CAD vendors, the use of a standardised building model as basis for application interoperability and co-operative work is rapidly taking shape. Nevertheless, IT environments in AEC are still limited to data exchange on the basis of e.g. STEP physical files (ISO 10303-21 1994), or to data sharing using one global product model implemented on top of a general-purpose relational or object-oriented database system. Applications making use of product data technology methods are mainly realised on the basis of the object-oriented programming paradigm, typically in the C++ programming language. However, as outlined in table 1 below, the general demands of concurrent engineering methodology (Prasad 1992; Scherer 1998) impose several additional requirements to product data management in building construction beyond integration and simple data exchange which can be tackled more efficiently with the application of advanced knowledge-based methods in product data management server software.

Table 1: Concurrent engineering issues requiring knowledge-based solutions

Basic Concurrent Engineering issues	Related PDM features	Appropriate knowledge-based solution methods
Collaborative work of physically distributed teams	• Distributed client/server architectures • Comprehensive data access and retrieval functions, providing just-in-time information • Continuous product model evolution	• Advanced agent-based communication paradigm • Search engines Search agents • Dynamic object evolution and classification methods
Co-operative problem solving	• Information sharing • Inter-model-operability	• Knowledge-based model mapping methods
Concurrency	• Multi-user access to product data • Availability of local view models together with methods for effective model transformations • Consistency checking • Change and version management	• Multi-agent architectures • Knowledge-based model mapping methods • Rule-based agents • Model matching and merging methods
Simulation, monitoring and forecasting	• Advanced simulation tools	• Agent technology • Probabilistic reasoning • Cased-based reasoning • Learning algorithms etc.

Many of the above features would be embedded in advanced client applications armoured with deeper engineering knowledge than the scope of a PDM server would allow, but there are also several issues that require inter-discipline problem solving and are therefore better suited for server-side solutions. Tasks that would typically be dedicated to advanced server-side tools are e.g. model mapping and matching (Khedro et al. 1994; Scherer and Katranuschkov 1997), building code services (Han et al. 1999), information searches in the full project space (Scherer 1998b) etc. Such tasks are too complex to be satisfied on the basis of one single IT paradigm. Therefore, what is needed is a *hybrid approach* combining traditional object-oriented methods with advanced AI problem solving and representation techniques within a flexible client-server architecture. Moreover, the architecture of the PDM server itself has to be chosen in such way that a uniform interface to client software is achieved, conforming to a commonly acknowledged modelling methodology, such as STEP (ISO 10303-1).

Based on these objectives a PDM server has been developed and implemented for the IFC project model, release 1.5 (IAI 1997). The prototype environment includes client applications based on Autodesk's Architectural Desktop, a finite element structural analysis program, a foundation design expert system, a knowledge-based construction simulation system for geotechnical works and a facility management information system (Scherer 1998a; Katranuschkov and Hyvärinen 1998).

2 Client/Server environment architecture

Client/Server solutions based on TCP/IP and making full use of the Internet are doubtlessly most suitable for distributed project development, typical for the virtual enterprise in building construction.

There are quite a few different architectures that can be envisioned for a concurrent engineering environment based on the client/server communication paradigm. On the client side a variety of tools have to be supported, as shown on the left side of fig. 1. On the server side, the following realisation alternatives can be envisioned:

1. **All-in-one PDM server**
   This is, conceptually, the simplest possible server design. It has the benefits, but also the drawbacks of not being dependent on any other services. In order to satisfy fully the requirements of concurrent engineering, such server implementation must embed several features which go far beyond product data management functionality, such as process and workflow management, actor authentication, secure communication etc. Such features make the server implementation very complex and consequently extremely difficult to achieve in practice. On the other hand, if such implementation concentrates on typical PDM tasks, it can still be useful even without the support of all required issues, provided that an appropriate work discipline is maintained within the project team.

2. **PDM Server – DBMS**
   This alternative is very similar to the above, the main difference being in the use of a comprehensive commercial DBMS interfaced through a standardised data manipulation language like SQL, OOSQL, or through ODBC requests.

3. **Information Logistics Server – PDM Server Interface – PDM Server – DBMS**
   In this approach, full concurrent engineering support is achieved with the help of an Information Logistics System (ILS) used as front-end server. The ILS encompasses: (1) an Object Request Broker that can identify among several possible servers the one responsible for answering an incoming client request (remote procedure call) and then forward the request to that server, (2) realisation of secure synchronous and asynchronous communication, and (3) identification and authentication of the issuing actor and/or software application (Wasserfuhr und Scherer 1998). With such architecture, all requirements to IT supported concurrent engineering can be satisfied, if a set of dedicated servers for comprehensive process, document and product management are available.

4. **ILS – PDM Server Interface – PDM Agents – PDM Server – DBMS**
   This is an enhancement of the architectures described above, allowing also intelligent server-side components to be integrated as automated software agents into the system. Such intelligent agents can accomplish more sophisticated server-side tasks like automated change recognition, notification of actors for possible conflicts, performing of certain well-defined automated services like cost estimation for prefabricated elements, automated model mapping depending on the state of the data etc. Therefore, in our opinion this alternative offers greatest advantages for concurrent engineering work in AEC projects.

The last alternative, shown schematically on fig. 1 below, has been actually used in the developed prototype, described in more detail in the following sections. Due to the prototype state of the development the application of a commercial DBMS was avoided for convenience.

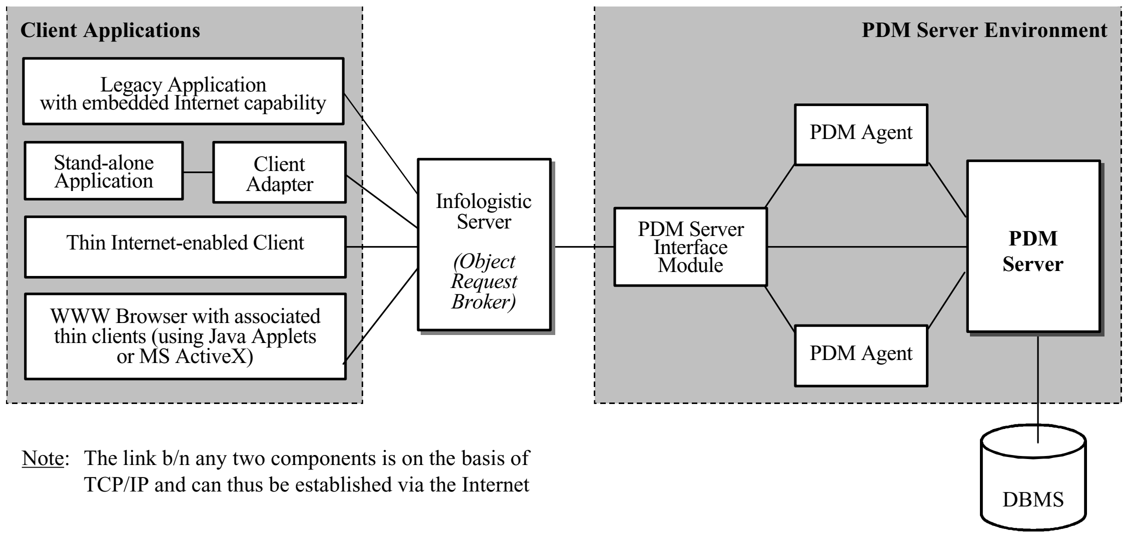
Client Applications
Legacy Application with embedded Internet capability
Stand-alone Application
Client Adapter
Thin Internet-enabled Client
WWW Browser with associated thin clients (using Java Applets or MS ActiveX)
Infologistic Server
(Object Request Broker)
PDM Server Environment
PDM Server Interface Module
PDM Agent
PDM Agent
PDM Server
DBMS
Note: The link b/n any two components is on the basis of TCP/IP and can thus be established via the Internet

Fig. 1: Advanced multi-tier client/server environment configuration

3 Knowledge-based product data representation

The realisation of PDM functionality is nowadays most commonly achieved on the basis of sets of loosely connected tools, or, at the high-end, with the help of integrated PDM servers using the object-oriented programming paradigm. However, according to our opinion, object-oriented methods alone are not powerful enough for the implementation of many advanced tasks required for efficient concurrent engineering support. Therefore we propose the use of a *hybrid approach*. In this approach the underlying modelling framework of the environment is based on the ISO STEP methodology, enhanced through the use of EXPRESS-C (ISO/TC184 1994) as a common modelling language for all supported data models. To enable coherent use of both the object-oriented paradigm based on the EXPRESS-C specification and knowledge-based methods applied in several product data services, such as model transformations, model matching, complex queries and assertions on the basis of generic templates etc., the representation of the product data models in the PDM server is done according to the *frame-based modelling paradigm* (Russel and Norvig 1995).

The suggested frame-based representation parallels the conceptual model schemata specified on the basis of EXPRESS-C. *Frames* can incorporate all object-oriented features of EXPRESS-C entities, such as inheritance, state (attributes) and behaviour (operations), but allow also to vary dynamically the number and type of their attributes (important for object evolution methods), to define different properties of the attributes themselves (important for dynamic object classification rules), as well as to associate rules to different frame structures. *Rules* are defined separate from the data and methods captured within frames, and can be used for run-time binding of variables to modelling object instances. This enables their use for information retrieval purposes in applications that do have knowledge of the model schema, but not necessarily of the particular model instantiations. Rules are triggered through the execution of ASSERT and QUERY functions from the body of the object-oriented server methods. Thus, rule-based reasoning can be seamlessly integrated in the ”normal” object-oriented requests of a client application to the PDM server, as shown on the simplified example in fig. 2.

TC_Model extension in EXPRESS-C:

```
ENTITY TC_MODEL SUBTYPE OF (IfcRoot);
...                          -- attribute specification
(* versionStatus: StatusEnum; -- inherited from IfcRoot *)
Operations                   -- EXPRESS-C method specification
   match (compVersion: TC_MODEL);
...
END_ENTITY;
```

When a modified version of an instantiated product model **M1** is checked in at the PDM server and its status is set to ”proposed”, e.g. by executing a remote procedure call from a client application like:

```
    M1.setAttribute (Name:”Status”, Value:”proposed”) *)
```

the following rule will be triggered automatically, allowing a separate agent process to determine all changes done to the model and to notify subsequently the affected actors:

```
IF (the ‘versionStatus of ?m is ‘proposed)  ; ?m is automatically bound to
   (the ‘projectID of ?m is ?id)            ;  the model instance M1.
   (?n is in ‘TC_MODEL)                     ;  Gets all ‘active’ instantiated
   ((the ‘projectID of ?n) = ?id)           ;  model versions with the same
   (the ‘versionStatus of ?n is ‘active)    ;  projectID on the server.
THEN
   (the ‘versionStatus of ?n is ‘frozen)
   (exec.operation ‘match (oid:?m inParams:(?n)))
```

*) Note: The syntax of client requests is simplified here and in all subsequent examples for convenience.

Fig. 2: Triggering a rule-based method from a client application

In this example, `exec.operation` is used to invoke the object-oriented method ”match”, thus allowing to combine rule-based and object-oriented techniques.

`match (oid: ?m inParams: (?n))` is the method that will be applied consecutively to model **M1** and any other model $\mathbf{N_i}$ satisfying the above rule. However, it will in turn trigger forward and/or backward chaining rules during its execution, switching seamlessly between object-oriented and rule-based processing techniques.

The actual implemented rules in the PDM server for the above example are somewhat more complicated, but without any principal difference. Similar rules are applied for other possible model states, allowing the PDM server agents to react automatically to various changes in the enviornment.

4 Architecture of the knowledge-based product data server

The outlined representation paradigm allows the development of a server architecture that very much resembles that of a typical knowledge-based system as shown in fig 3. However, in contrast to expert systems, designed to assist engineers in the solution of specific problem solving tasks, the goals of the knowledge-based methods in the PDM server are to react to general-purpose, yet complex queries and assertions, and to adapt dynamically to a variety of changes in the product models resulting from the concurrent, simultaneous work of the separate actors in a project team.

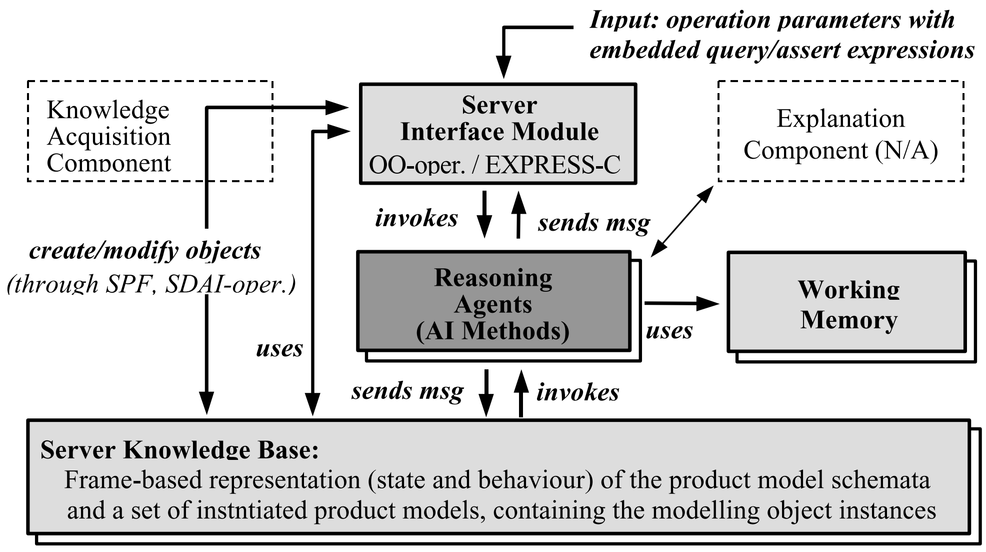
Input: operation parameters with embedded query/assert expressions
Knowledge Acquisition Component
Server Interface Module
OO-oper. / EXPRESS-C
Explanation Component (N/A)
invokes
sends msg
create/modify objects
(through SPF, SDAI-oper.)
Reasoning Agents (AI Methods)
Working Memory
uses
uses
sends msg
invokes
Server Knowledge Base:
Frame-based representation (state and behaviour) of the product model schemata and a set of instntiated product models, containing the modelling object instances

Fig. 3: Knowledge-based server architecture

The main component of the proposed server architecture is its *knowledge base* containing a set of predefined data model schemata, as well as one or more product models associated with each model schema and representing the separate work spaces and views of each actor on the designed building product. Each data model schema in the knowledge base is organised in a frame structure, as described in the previous section, and contains exclusively *class frames*, roughly corresponding to the entity structuring given in the EXPRESS-C schema of the model. The instantiated product models, on the other hand, contain only *instance frames* with inherited attributes and behaviour from the respective underlying data model schema. All frames can act both as ”normal” objects, responding to messages from the outside and/or from other objects, as well as to participate in rule-based queries and assertions. The knowledge base can accommodate also overall rules similar to the global rules in EXPRESS.

The *interface module* has the task to resolve properly the requested operations by a remote client, and to activate the appropriate server object methods responsible for the execution of the client requests. All incoming requests and result responses are also based on the formal EXPRESS-C specifications in the data models, regardless of the particular internal server method used (pure object-oriented or knowledge-based implementation). In this way, the interface module allows to hide from the users and clients the implementation details of the server, which simplifies greatly client implementation.

The actual processing of a request is done in the following steps: (1) the request is parsed, the data are transformed to an appropriate internal format and the specific task is identified; (2) except when explicitly given, the expected server load is judged and the task is classified as a synchronous, high-priority asynchronous or low-priority asynchronous (heavy-duty) task and is then queued respectively; (3) when the task is awaked for execution, an appropriate server method is chosen and activated, and (4) the results of the performed method(s) are stored on a solution blackboard and in the respective data model in order to be available for other server modules, and are at the same time transformed in a respective server response to the calling application.

The *reasoning agent* has the task to perceive changes in the environment and to react to such changes with appropriate goal-driven, utility-driven or reflexive actions (Russel and Norvig, 1995). Within the PDM server architecture each reasoning agent is invoked by an object-oriented method whenever a knowledge-based function is addressed. This can be done explicitly, by executing a respective operation, such as:

```
IfcRelUsesProducts.find (
   searchExpr:(FOR ?X DO (?X HAS AT LEAST 2 RelatedObjects)))
```

or happens automatically, when the state of an attribute monitored by a demon method is changed, as shown e.g. in fig. 2 for the `TC_MODEL` operation `setAttribute`.

The general syntax of a search expression in EBNF format is

(FOR { ***?X**$_i$* }* **DO (** <***searchTemplate***> { **AND** | **OR** <***searchTemplate***> }* **))**

where ***?X**$_i$* is a free variable, bound dynamically to the set of values satisfying the search, and **<*searchTemplate*>** is used for executing a predefined query, such as subclass membership, maximum cardinality, equality, attributes satisfying a given condition etc.

Typically a reasoning agent will retrieve the required data from the knowledge-base together with the appropriate reasoning rules and will place them in the working memory of the server. After that the inference process on the retrieved data and rules is started, and, on its completion, the result is returned to the calling method. Currently, the server implementation incorporates an inference engine including forward and backward chaining as well as different search algorithms. However, these methods can be extended using the same component architecture.

The remaining (dashed-box) components shown in fig. 3 are given only for comparison with typical expert system architectures. An explanation component is difficult to envision for a server software, and a comprehensive knowledge acquisition component has been beyond the scope of the server implementation, although new knowledge can (and is) continuously introduced in the knowledge base through respective ”assertion” requests.

5 Knowledge-based product data services

The Application Programming Interface (API) of the prototype implementation of the proposed knowledge-based PDM server encompasses more than 40 operations that can be used by a remote client application. Most of these operations are based on SDAI (ISO 10303-22 1996) or support data exchange functionality on the basis of STEP physical files (ISO 10303-21 1994). However, in order to provide an enabling infrastructure for concurrent engineering, several sophisticated services involving knowledge-based operations have been specified as well. Such services include model mapping, model matching and model merging, conflict recognition, generic consistency checking etc. They have all been prototyped on the basis of the distributed client-agent-server architecture described in the previous sections.

To illustrate the approach here we shall consider a simple practical example involving model mapping. The realisation of the other knowledge-based services adheres in principle to the same overall methodology.

5.1 Model mapping

Any application needs to extract a view from the product model, and the transformation from the product model to such view is unique to the application (Clancy 1985). The goal of model mapping is to enable such model transformation - in the ideal case automatically, without any user interaction.

Model mapping involves the conversion of one modelling representation to another without awareness of the context, i.e. already existing local data in the target model (Scherer and Katranuschkov 1997). The mapping process is performed on the basis of a formal specification of all necessary translations on class level, and includes full or partial transformations of the data instances according to this specification. As a basic interoperability problem in product data based environments, model mapping has been intensively investigated in the last years, and several formal mapping languages and implementation techniques have been proposed (Liebich et al. 1995; Katranuschkov and Scherer 1996). However, whilst all proposed generic approaches are quite useful for creating full new views or versions of a product model, they are complicated to use when context-dependent transformations are required because all generic specifications are done on class level and are therefore static, i.e. they cannot reflect the changes in the environment. Such context-dependent transformations can be realised much easier with the help of the suggested knowledge-based approach.

To focus the discussion, let us consider the following example: ”mapping from an IFC product model to a foundation design system”.

The necessary input to the foundation design system is fairly simple. It is comprised roughly of the geometric and material properties of the ground plate of the examined building, the soil properties of the site, the points at which the overhead structure interacts with the foundations and the loads applied to these points. Though not very voluminous, this information is available in an IFC model in quite different form. Moreover, it is depending also on the specific instantiated context, i.e. the particular relations between the product data instances. The needed loads can for example be represented in IFC as support reactions of IfcWall and IfcColumn entities, stored in appropriate property sets[*)], but this is only valid for elements connected to the particular ground plate under consideration.

Such mapping conditions are extremely difficult to specify on class level, and the use of a generic implementation approach can lead to considerable computational time. On the contrary, with the help of the described knowledge-based query functions the same conditions can be easily specified with minimal user input and the mapping can be performed automatically by a mapping agent, whenever it detects that the necessary input data are available. In fact, the concrete realisation of the considered mapping example has required only 4 different knowledge-based queries, such as:

```
Obj.find
    (searchExpr:
       (FOR ?D ?V DO
            ((THE HasProperties OF Obj IS ?Y) AND
             (((THE Descriptor OF ?Y) = ”NODE_REACTIONS”) OR
              ((THE Descriptor OF ?Y) = ”LINE_REACTIONS”)) AND
```

[*)] Such property sets are not elements of the current IFC project model specification. They have been used in the implemented prototype to enable the consideration of structural engineering software in the environment.

```
        (THE HasProperties OF ?Y IS ?Z) AND
        (THE Descriptor OF ?Z IS ?D) AND
        (THE ValueComponent OF ?Z IS ?V))))
```

Such queries may appear complicated, but their specification is straight-forward and, because of their expressive power, the full implementation of the mapping (in the Java language) is fairly simple – in this case just about 100 lines of code.

6 Discussion

Concurrent engineering requirements are not well supported in existing IT environments for AEC. One reason for that are the limitations of object-oriented technology which is used as basis in most known implementations. The hybrid approach proposed in this paper allows to overcome many of these limitations with the help of appropriate knowledge-based server-side tools. With such dedicated tools it is easy to implement various client adapters, with a GUI support for all needed PDM services.

The suggested knowledge-based architecture of the PDM server provides the following benefits: (1) separation of data structures from high-level rule-based data management knowledge, (2) encapsulation of knowledge-based methods within the procedural attachment of operations to the modelling objects, enabling the use of uniformly defined function calls from the outside, (3) full consistence with the conceptual object-oriented modelling approach. On the basis of this architecture, a comprehensive PDM environment incorporating different building design services as suggested e.g. in (Han et al. 1999; Turk et al. 1998 etc.) can be envisioned.

7 Acknowledgements

This work has been supported by the Commission of the European Community under contract No. 20587, for which we want to express our thanks.

8 References

Clancy, W. J. (1985). *Heuristic Classification in Artificial Intelligence*, Elsevier, Amsterdam.

Han C.S., Kunz J.C., Law K.H. (1999) Building Design Services in a Distributed Architecture. *Journal of Computing in Civil Engineering* 13 (e), pp. 12 - 22.

IAI (1997). *IFC Object Model for AEC Projects.* IFC Release 1.5 Model Reference Documentation, Final Version, IAI Publ., Washington, DC.

ISO 10303-1, -11, -21 IS (1994). *Product Data Representation and Exchange - Parts 1, 11, 21.* ISO TC 184/SC4, Geneva, Switzerland.

ISO 10303-22 DIS (1996). *Product Data Representation and Exchange - Part 22: Implementation Methods: Standard Data Access Interface Specification.* ISO TC 184/SC4, Geneva, Switzerland.

ISO/TC184/SC4/WG5/N202 (1994). *PISA Information Modelling Language (EXPRESS-C).*

Katranuschkov, P. and Hyvärinen, J. (1998) Product Data Server for Concurrent Engineering in AEC, in *Product and Process Modelling for the Building Industry*, Proc. of the ECPPM'98 (ed. R. Amor), BRE, Watford, UK.

Katranuschkov, P. and Scherer, R.J. (1996) Schema Mapping and Object Matching: A STEP-Based Approach to Engineering Data Management in Open Integration Environments, in *Proc. of the CIB W78-96 Workshop Construction on the Information Highway*, Bled, Slovenia.

Khedro, T., Genesereth, M.R. and Teicholz, P.M. (1994) Concurrent Engineering Through Interoperable Software Agents, in *Proc. 1st Conference on Concurrent Engineering: Research and Applications*, Pittsburgh, PA.

Liebich, T., Amor, R. and Verhoef, M. (1995) A Survey of Mapping Methods Available Within the Product Modelling Arena, in *Proc. of the 5th Int. Conf. of the EXPRESS User Group*, Grenoble, France.

Prasad, B. (1996) *Concurrent Engineering Fundamentals (Integrated Product and Process Organization).* Prentice-Hall, Englewood Cliffs, NJ.

Russel, S. and Norvig, P. (1995) *Artificial Intelligence: A Modern Approach*, Prentice-Hall, Englewood Cliffs, N. J.

Scherer, R.J. (1998a) A Framework for the Concurrent Engineering Environment, in *Product and Process Modelling for the Building Industry*, Proc. of the ECPPM'98 (ed. R. Amor), BRE, Watford, UK.

Scherer, R.J. (1998b) AI methods in concurrent engineering, in: *Artificial intelligence in structural engineering*, (ed. Ian Smith) Lecture note in AI, Vol. 1454, pp. 359 -384, Springer, Germany.

Scherer, R.J., and Katranuschkov, P. (1997) Framework for Interoperability of Building Product Models in Collaborative Work Environments, in *Proc. of the 8th Int. Conf. on Civil and Building Engineering*, (Eds. C.-K. Choi, C.-B. Yun, H.-G. Kwak), pp. 627 - 632, Seoul, Korea.

Turk, Z., Cerovsek T. and Duhovnik J. (1998) Regulation Processing in ToCEE, in *Product and Process Modelling for the Building Industry*, Proc. of the ECPPM'98 (ed. R. Amor), BRE, Watford, UK.

Wasserfuhr, R. and Scherer, R. J. (1997) Information Management in the Concurrent Design Process, in *Proc. Int. Colloquium IKM'97*, Weimar, Germany.