Notification and Change Propagation Support in a Concurrent, Multi-actor Environment

Grahame Cooper, Yacine Rezgui, Paul Hayes, Mike Jackson,
Farhi Marir, Jim Yip and Peter Brandon

Introduction

A great deal of work has been carried out in the area of information modelling to support the sharing of information across participants in a construction project. Efforts in this area include projects such as ATLAS (Bohms et al., 1994) (for large scale engineering), COMBINE (Dubois et al., 1995) (for HVAC and building design), RATAS (Björk, 1994) and ICON (Aouad et al., 1995) (for building design and construction management). It is also the subject of standardization efforts such as STEP (ISO/TC184/SC4, 1994), and more recently the IAI (see: http://www.interoperability.com/)

In order to support collaborative engineering, however, it is important not only to share information, but also to manage that information in a manner that actively promotes integration. This paper describes the central models underlying the COMMIT project, which is defining mechanisms to handle a number issues relating to the management of information to support decision-making in collaborative projects. The project is being carried out in consultation with a well-established steering group comprising standardisation bodies, industrials and researchers. However, the paper describes ongoing research, and comments are invited regarding the models to allow improvements to be made over the remaining life of the project.

The COMMIT project is supported by the Engineering and Physical Sciences Research Council in the United Kingdom.

The COMMIT Project

This project is concerned with the management of information to support decision making in multi-actor environments. Whilst the work of the project is not construction specific, it is being carried out within the context of construction, and therefore emphasises construction project integration as an important goal.

COMMIT addresses six primary issues that are central to information management:

- ownership, rights and responsibilities;
- versioning of information;
- schema evolution;
- recording of intent behind decisions leading to information;
- tracking of dependencies between pieces of information;
- notification and propagation of changes.

Whilst many of these are distinct issues, they have been found to be closely inter-related, making it difficult to address them individually. Indeed, much of the modelling effort is expended on the need to understand the interplay between these different issues, as will be illustrated later

The project is being carried out in two phases. The first phase is primarily concerned with the creation of an object model describing the concepts and the interplay between them. This is the role of the CIMM (see below). The second part of the project concentrates primarily on the capture of intent behind decisions in a matter that involves only the smallest possible intrusion into the process. This work is currently being carried out. Throughout both of these phases, implementation work is being carried out on the creation of prototype software

Modelling Methodology

The COMMIT project is committed to a fully object oriented approach, in which an emphasis is placed on inter-working between software objects rather than on sharing of data using common formats. This approach provides a number of advantages in terms of the ability to use abstraction to handle complexity. (See: Cooper, 1995; Booch, 1994).

Initially, the models were defined using a CASE tool called Object Engineering Workbench. This is an object-oriented CASE tool that is quite closely bound to the C++ programming language, and uses a proprietary modelling language. The current version of the COMMIT model is being re-built using UML (the Unified Modelling Language)

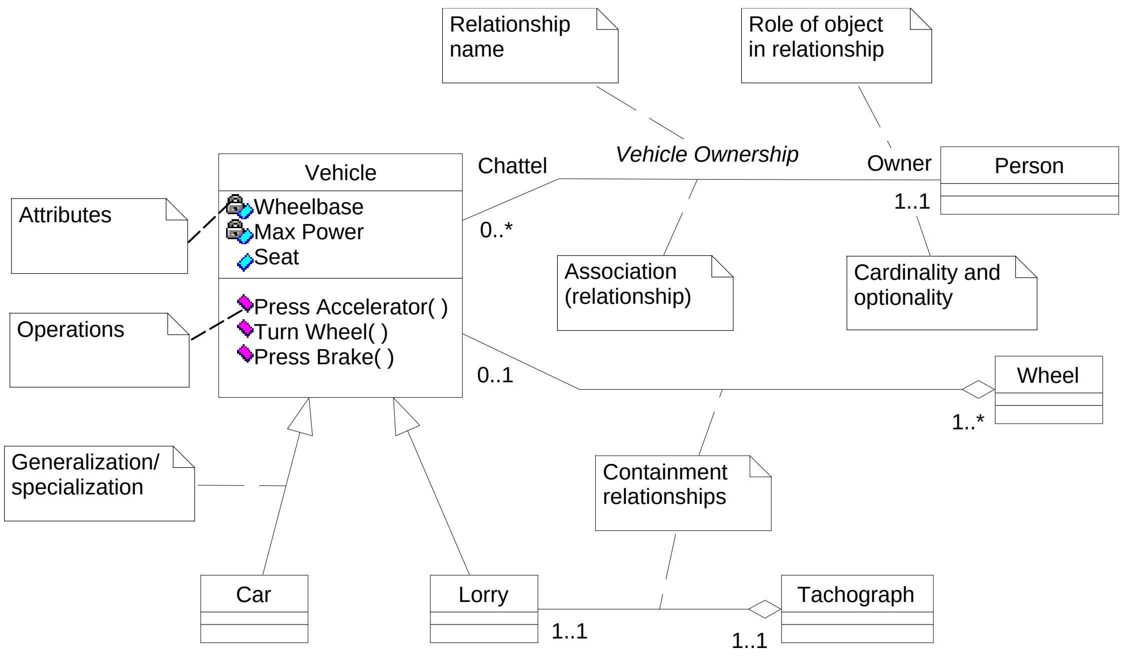

Figure 1 Illustration showing the main elements of a UML class diagram

in the Rational Rose CASE tool. This language is rapidly being established as a de facto standard for object-oriented modelling, and is well supported by the Rational Rose CASE Tool. UML is developed primarily from two of the most popular modelling formalisms for object-oriented modelling, OMT (Rumbaugh et al., 1991) and Booch

(Booch, 1994), and currently being considered for adoption as an international standard within the OMG (Object Management Group). The models shown in this paper are expressed in the UML. Figure 1 illustrates the main elements of UML that are used in this paper. Further information on UML can be found in (UML, 1997). The use of a CASE tool such as Rational Rose is helpful because the tool can automatically generate both C++ code and CORBA IDL (Common Object Request Broker Architecture Interface Definition Language). This allows the COMMIT implementation immediately to take advantage of the inter-working facilities provided by an Object Request Broker and the CORBA standard.

The COMMIT Information Management Model (CIMM)

At the core of the COMMIT architecture is the CIMM (COMMIT Information Management Model). The purpose of the CIMM is to model the concepts that surround the six issues addressed by COMMIT. This model forms the framework in which information management is carried out within COMMIT, and is used to generate the software object classes that make up to realization of these concepts through the COMMIT prototype. The model is described in detail in the following sections. It is important to remember that the scope of the CIMM covers information management only. In particular, it does not include project management, and this has been an important guiding principle in defining the scope of these models.

Versioning and Schema Evolution

If we are to record the relationships between information that is current in a project and information that was defined at some earlier time, it is necessary to keep old versions of information. It can also be useful to keep alternative versions of current information to deal with situations in which a decision has not yet been made from a

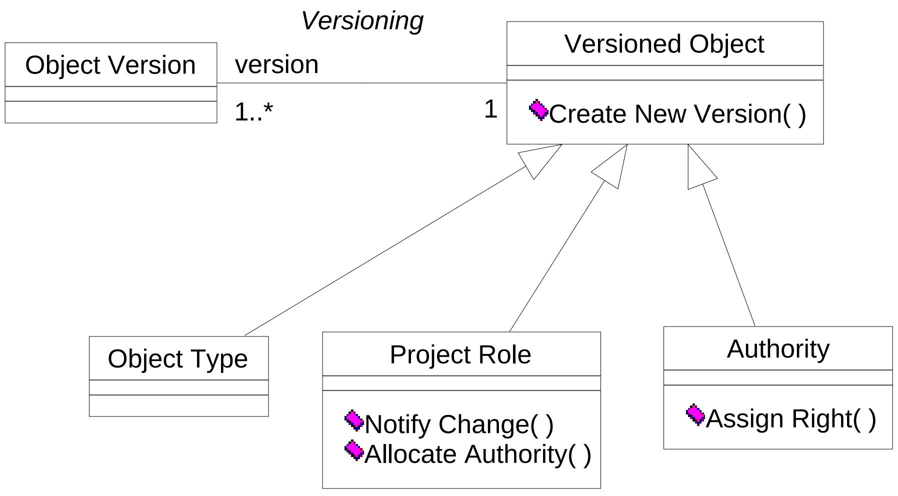

Figure 2 Versioned objects in the CIMM

number of possible alternatives. Figure 2 shows some of the object types that are versioned in the current version of the CIMM. In addition to these, all the core project objects such as Wall, Task, Resource, etc. are versioned (though they are outside the scope of the CIMM itself).

Figure 3 recognizes that object types themselves are versioned, and thereby introduces the concept of schema evolution. This is important because the schema (or model)

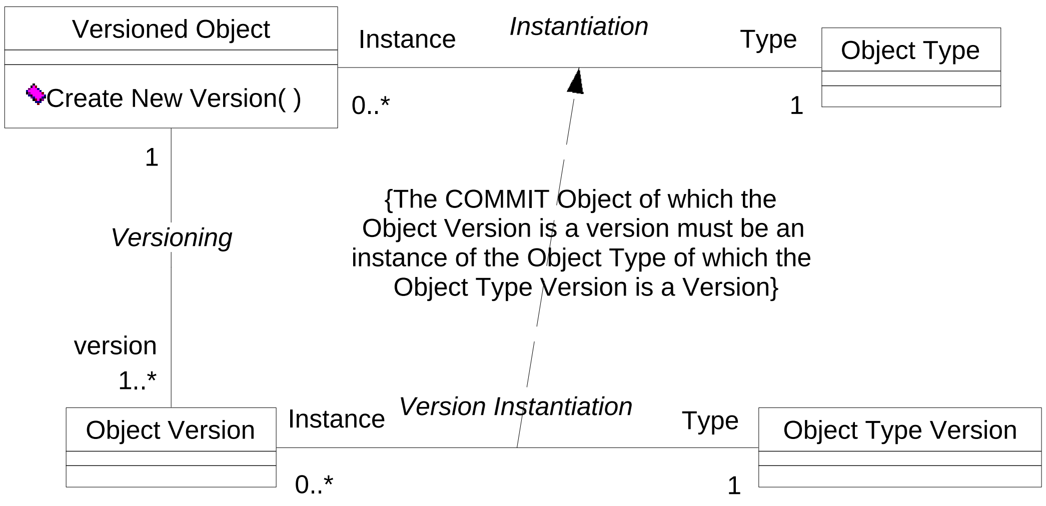

Figure 3 Versioning and Schema Evolution in the CIMM

provides a semantic context for all objects in the system, but may itself need to change over time. Thus, earlier versions of object classes need to be kept in order that the semantics of earlier versions of objects may be understood.

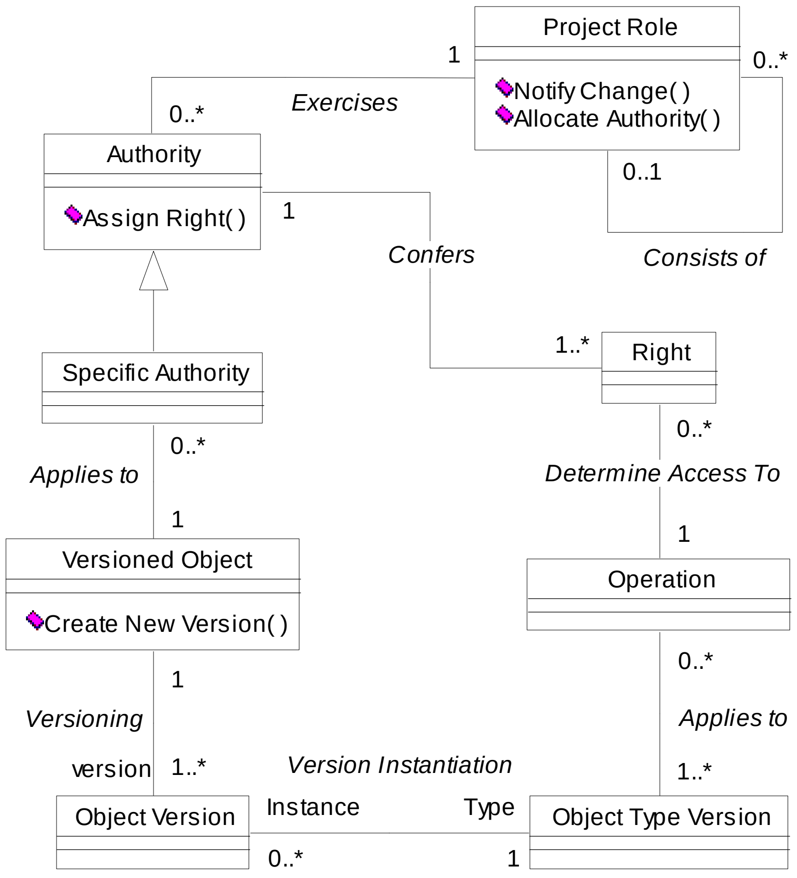

Figure 4 Roles, Rights and Responsibilities in the CIMM

Project Roles, Rights and Responsibilities

Figure 4 shows the basis for the handling of ownership, rights and responsibilities in the CIMM. All actors participate in a project by means of one or more roles. Thus it is the concept of a role rather than an actor that falls within the scope of the CIMM. Through a role, an actor exercises authority over some parts of the project information, and each Authority is characterized by a number of responsibilities that relate to a particular object. In order to discharge those responsibilities, the actor (through the role) needs to have certain rights to perform actions (or Operations) on the object in question.

An important principle here is the use of operations on objects to define rights. This is in contrast to the conventional approach of assigning rights as: *Create*, *Read*, *Update* and *Delete*. In the COMMIT approach, the emphasis is on using real world concepts through the abstractions that may be represented in a true object-oriented model. For example, a project manager might have authority over a task in the project plan. In order to discharge the responsibilities associated with that authority, the project manager would need the rights to assign resources to the task, to move the start date of the task, etc. However, he or she might not have the right to delete the task, or change its duration. Naturally, this would depend on the specific situation, and the rights that the project manager would have over tasks might be different for different tasks. This suggests that rights must ultimately be defined at the instance level (e.g. over a particular task rather than over tasks in general).

In practice, however, there could be a huge number of combinations of rights to be assigned to roles and objects. For this reason, mechanisms are provided in the CIMM

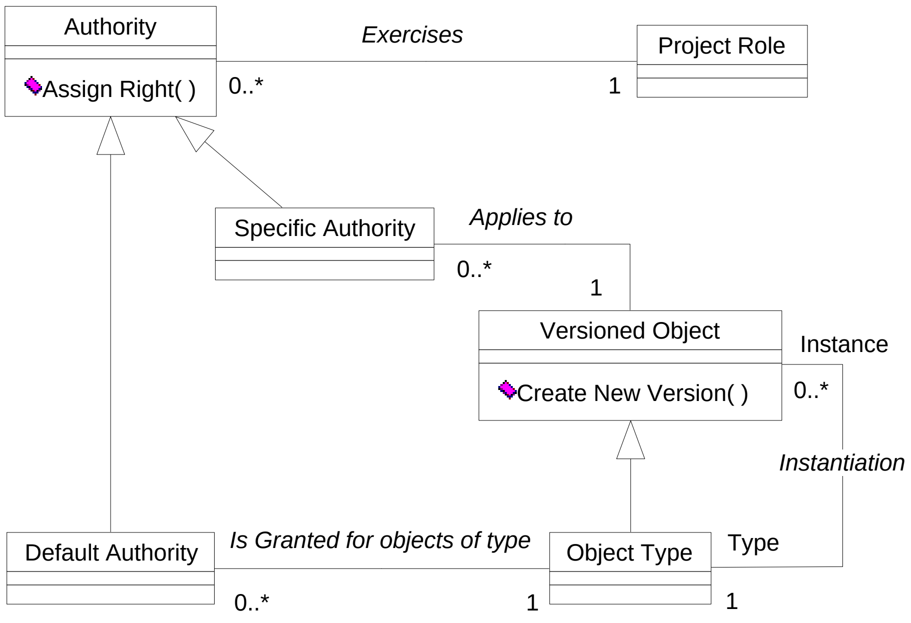

Figure 5 Use of type to define default Authorities.

to allow default authorities to be defined over all objects of a given type. These authorities would apply in the absence of a specific authority for an instance. Similar mechanisms are being developed in relation to containment, whereby default authorities may be allocated in respect of an object by virtue of its being contained, or owned, by another object.

Change Notification and Propagation

Notification is the process by which amendments made to objects are notified to other objects that may have an interest. Figure 6 shows how this is handled conceptually in the CIMM. A notification obligation represents the fact that an object is required to invoke some operation on another object whenever a particular kind of change occurs (i.e. whenever an operation is performed on it). The invocation of an operation is considered to be identical to the passing of a message to an object. It is then the responsibility of the object receiving that message to decide how to respond. This is

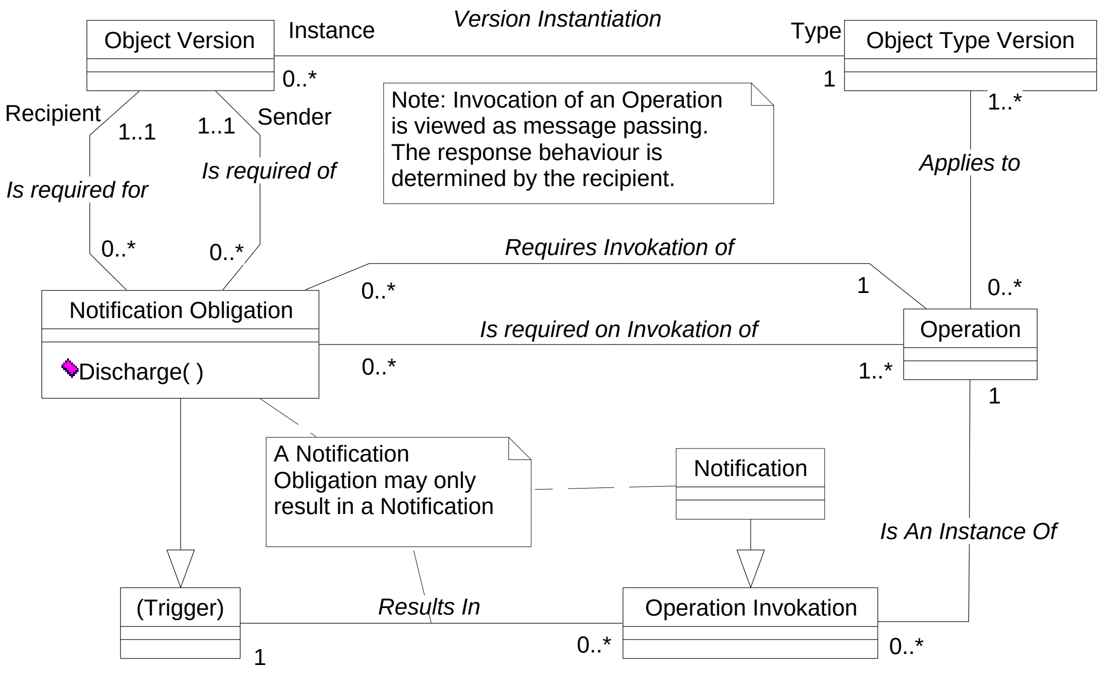

Figure 6 Change Notification and Propagation in the CIMM

an important element of abstraction in object-oriented systems and makes for a good separation between the semantics of an object in terms of its interface, and the implementation of that object in software.

In general, some trigger is responsible for causing an operation to be invoked. A trigger is a reason for some operation to be performed upon an object. In the current version of the CIMM, there are two kinds of trigger. One is that a decision has been made, and this is discussed in the following section. The other is that a notification obligation has been discharged as a result of some other change. A notification obligation is discharged whenever a particular operation is performed on a particular object. It results in another operation being invoked on another object, and this operation is considered to be a notification.

In some cases, the object to be notified of a change might be another object in the project. E.g. a window may need to be notified that the position of the wall containing it has been moved; or a project task to build a wall may need to be informed that the wall has been increased in size. In other cases, the object to be notified might be a role, in which case a message would need be sent to the actor or actors carrying out that role.

The spread of electronic mail and fax means that the CIMM can not only represent the notification of actors through roles, but also can, to a large extent, automate it. Many individuals will be unused to working in an integrated construction environment in which shared project information changes at a rapid pace. A notification mechanism is therefore essential for keeping actors aware of project changes and also supports the automation of other information management processes such as approval. Propagation involves changing the properties of a set of target objects because of a change introduced to a source object.

It is important, however, that the concepts of notification of actors, and change propagation, which is the notification of objects that will then automatically update themselves, are unified under the general heading of Notification. This unification allows decisions that are currently made by human beings to be automated at some time in the future through the appropriate use of knowledge-based system technologies.

Decisions, Interdependency and Capturing Intent

When a decision is made in a project, it should be made on the basis of available information, and will result in new or changed information. This is the underlying concept behind the tracking of dependencies between pieces of information in the

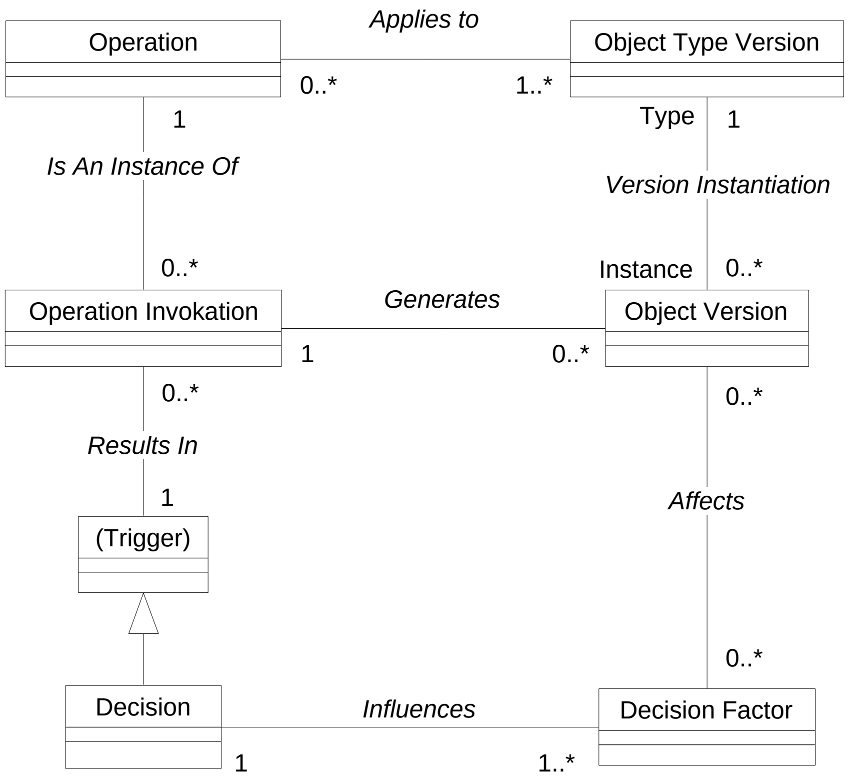

Figure 7 Decisions and Interdependency in the CIMM

COMMIT system. Figure 7 shows the way in which decisions are recorded in the CIMM, and the manner in which the dependencies are handled between decisions and the various versions of information stored in the system. It shows that a decision is influenced by a number of factors, each of which is determined by information stored in the form of objects (or rather versions of objects). Once the decision is made, it is a trigger for operations to be performed on objects, which will result in the creation of new versions of objects and new objects.

As an example, consider the situation in which the thickness of a wall needs to be maintained at the same width as that of an adjacent column, say for aesthetic reasons. A notification obligation would be placed on the column to inform the wall whenever its own thickness is changed (perhaps by the structural engineer). By default, the wall could inform the architect that the column has changed size and that its own size might need to be changed. It could even respond by asking for (or requiring) a message to go from the structural engineer to the architect explaining the reasons for the proposed change to the column, and asking for that change to be approved and taken account of. In this case, the content of the message could be captured and stored as part of the decision factors for the decision to change the width of the column. Later, as design technology improves, it may become possible for this change propagation to be automated through the use of knowledge based software embedded in the wall object. However, the underlying model of notification, propagation and capturing of intent remains the same.

Conclusion

This paper has described the underlying models of the COMMIT project, which is concerned with the management of object based information in a construction project context. These models are continuously being developed and improved in the light of experience from prototype implementation work, and feedback from the project's partners in standardisation, industry and research.

The emphasis in the COMMIT project has now moved onto ways of utilizing the models to improve integration and decision-making in practice. Issues are being investigated surrounding the capture of intent behind decisions, in ways that interfere as little as possible with the working practices of the construction project participants. This involves capturing information about the object versions that were consulted in reaching a decision, and the use of information captured in requests for approval of changes, etc.

Also under investigation is the creation of techniques to analyse the network of information captured within the COMMIT system to improve the availability of timely and relevant information to support decision making in real time. This work is in its early stages and will be reported on in a future publication.

Acknowledgements

The COMMIT team wishes to acknowledge the contributions made by the members of the ICON steering group through their involvement in the ICON, OSCON and COMMIT projects.

This work is supported by the EPSRC under the project grant “Intelligent Integration of Information for Construction”.

References

Aouad, G. et al., (1994), ICON Final Report, University of Salford.

Björk, B-C. (1994). RATAS Project - Developing an Infrastructure for Computer-Integrated Construction, Journal of Computing in Civil Engineering, Vol. 8, No. 4, 400-419. http://www.vtt.fi/cic/ratas/index.html

Bohms, M., Tolman, F. and Storer, G. (1994). ATLAS, a STEP Towards Computer Integrated Large Scale Engineering, Revue internationale de CFAO, Vol. 9, No. 3, 325-337. http://www-uk.research.ec.org/esp-syn/text/7280.html

Booch, G (1994) Object-Oriented Analysis and Design with Applications, 2/e. ISBN 0-8053-5340-2. Addison-Wesley.

Cooper, G.S. (1995) Object-Oriented Databases from an Information Technology Viewpoint: Knocking Down Some Walls, in Object Technology and its Application in Engineering. ISBN 0902 376 209, Ed. Professor James Powell, proc Conf on Object Technology and its Application in Engineering, Glasgow, March 1995. DRAL (1995).

Dubois, A.M., Flynn, J., Verhoef, M.H.G. and Augenbroe, F. (1995) Conceptual Modelling Approaches in the COMBINE Project, presented in the COMBINE final meeting, Dublin. http://erg.ucd.ie/combine/papers.html

Froese, T. and Paulson, B. (1994), OPIS: An Object Model-Based Project Information System, Microcomputers in Civil Engineering, No. 9, 13-28. http://maillist.civil.ubc.ca/~tfroese/pubs/

ISO/TC184/SC4 (1994), STEP Part 1: Overview and Fundamental Principles, International Standard, ISO, Geneva, (11). http://www.igd.fhg.de/www/igd-a2/hyperstep/iso-10303/part1/gen.html

OMG (1995), The Common Object Request Broker: Architecture and Specification, OMG. http://www.omg.org/corbask.htm

UML (1997) UML Document Set, Version 1.0, 13 January, 1997 (Rational Software Corporation). http://www.rational.com/uml/index.html

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Lorensen, W. (1991), Object-Oriented Modelling and Design. Englewood Cliffs, New Jersey: Prentice-Hall.