THE ARCHITECTURE AND IMPLEMENTATION OF A DISTRIBUTED COMPUTER INTEGRATED CONSTRUCTION ENVIRONMENT

Alex Brown[1], Grahame Cooper[1], Yacine Rezgui[1], Peter Brandon[2] and John Kirkham[1]

ABSTRACT

This paper argues that distributed object technology can profitably be put to use in the field of computer integrated construction, and describes its use within the COMMIT[3] project. The COMMIT project aims to provide a computer integrated construction (CIC) environment that provides facilities such as versioning, notification, object rights and the representation of intent. To this end the COMMIT Information Management Model (CIMM) has been proposed, a brief description of which is given here. The paper describes an architecture for a distributed implementation of the CIMM. This architecture uses the CIMM as a bridge between services provided the CORBA distribution standard and core objects in a computer integrated construction environment. A detailed description of the CIMM's implementation is given, in particular focusing on CIMM objects which reuse CORBA services through specialisation. Several techniques for integrating the CIMM with existing CIC systems and legacy applications are described. Finally, the user interface through which project participants interact with distributed components in the integrated environment is presented.

KEYWORDS: Computer Integrated Construction, Distributed, CORBA, Component, Implementation.

1. DISTRIBUTED OBJECTS AND CORBA

The next few years will undoubtedly see major changes in the use of computers. Improvements in hardware technology - in particular, the advent of inexpensive, high performance CPUs - are resulting in increasingly powerful desktop machines. The computing resources of an organisation or project team are now spread across many (often heterogeneous) platforms in different locations. At the same time, business application end users are increasingly seeking more local autonomy and responsibility. These changes are being reflected by a growing interest in distributed object technology. Many computer industry analysts see this as the next evolutionary step in a process which has already moved from monolithic applications to the client/server era.

There are currently two major emerging standards for distributed objects, Microsoft's Object Linking and Embedding (OLE) (Brocksmidt, 1995) and the Object Management Group's Common Object Request Broker Architecture (CORBA) (OMG, 1995 [1],[2]). Without going into detailed comparisons of OLE and CORBA which have been made elsewhere (Orfali et al., 1996), this paper will refer exclusively to CORBA for the following reasons. At the time of writing CORBA would seem to have more to offer projects aimed at computer integrated construction; it fully supports the principles of object-orientation whereas OLE

[1] Information Technology Institute, University of Salford
[2] Department of Surveying, University of Salford
[3] COnstruction Modelling and Methodologies for the Intelligent inTegration of information.

does not support inheritance. It also offers a wider range of facilities and services for integration; one estimate puts CORBA about two years ahead of OLE (Orfali et al., 1996). If OLE (or some other) standard prevails, we assume it will offer services along similar lines to those currently offered by CORBA.

The heart of the CORBA specification is the Object Request Broker (ORB) which is responsible for managing objects, message passing between objects, and the interface between objects and external services (Fig. 1). An object is an identifiable, encapsulated entity that provides one or more services requested by clients, the requests are events that have parameters and a target object, and may be generated by a client object when a service or piece of information is required. Objects have references which are unique throughout the system; these references may be used by applications without knowing where objects reside. This provides 'network transparency', and allows programmers to develop systems with a network wide address space.

The ORB allows applications to be constructed at both compile and run time. The Interface Definition Language (IDL) defines object structures so that applications with prior knowledge of the objects with which they will interoperate can be constructed. The Dynamic Invocation Interface (DII), allows object structures to be accessed at run time and so allows applications to be assembled from objects that have no prior knowledge of each other.

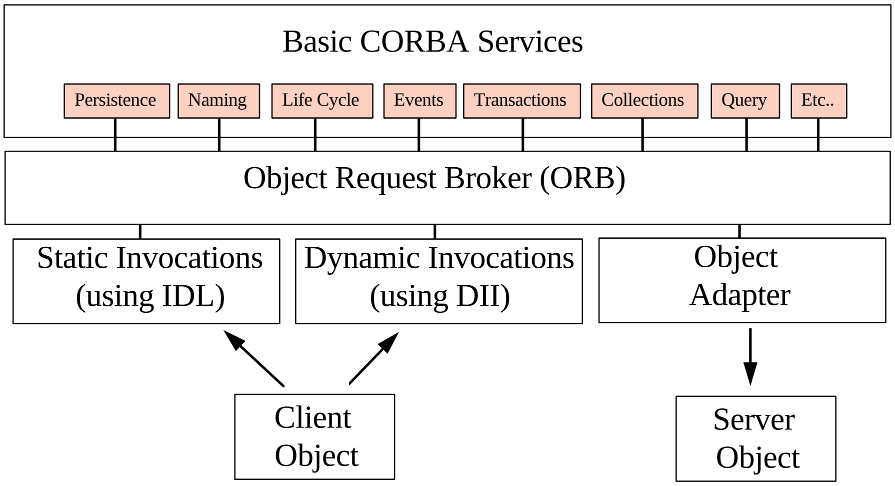


Figure 1. Simplified CORBA Architecture

The ORB also provides the low level services that are necessary for distributed computing. These services include naming, life cycle, persistence, events, transactions and concurrency. It is worth noting that, in the past, these services have been provided by a range of different vendors and technologies. Persistence, for example, has typically been provided by relational and object-oriented databases, while naming has been provided by services such as ISO’s X.500. The rationale behind much of CORBA is that it can transparently encapsulate these existing services by providing a standard interface to distributed objects.

2. DISTRIBUTED COMPUTER INTEGRATED CONSTRUCTION

Background research in construction shows that integration has been addressed in a variety of ways: communication between applications (achieved through specific software integrating a unique and invariant set of chosen applications), integration through geometry (often the case in commercial CAD packages where integration is based on and limited to geometrical information), knowledge-based interfaces linking multiple applications and multiple databases (Howard, 1991) and integration through central project databases holding all the information relating to a project according to a common infrastructure model. The latter category includes ATLAS (ATLAS, 1992) (for large scale engineering), COMBINE (Dubois et al., 1995) (for HVAC and building design), RATAS (Bjork, 1993) and ICON (Ford et al., 1994) (for building design and construction management). Although these projects have, in many cases, successfully demonstrated the sharing of construction information, we argue that there are significant benefits in adopting an approach to integration based on distributed components.

The nature of the construction industry is such that virtual teams are often brought together for projects before being broken apart again on completion. The software applications used may vary from one construction project to another. Organisations and individuals participating in a team will bring their own unique skills and resources, which may include legacy applications and data. Any integrated environment should provide a means by which the component objects provided by organisations and application vendors can interoperate in a seamless way. This interoperability should not be limited to those components which have a prior knowledge of each other; components should “plug and play” and so be usable in ways which were unanticipated by the original developers.

There is therefore a need to concentrate on the interfaces between objects as well as standardising the objects themselves. Instead of integration being achieved through “core models” that define the structure of shared information (in the form of files or databases), integration should be made through frameworks which define semantic relationships between the interfaces of separate components. The CORBA architecture provides for these frameworks through business object facilities (also known as vertical common facilities) (OMG, 1996) and distinguishes them from basic services (such as naming, persistence, transactions, etc.) and horizontal common facilities (such as user interfaces and system management). Business object facilities are already under development for several business areas including computer integrated manufacturing and banking. It is to be expected that existing STEP standards and other integration models will play an important part in the definition of a CORBA facility for computer integrated construction.

One advantage that CORBA frameworks have over existing methods of integration is that they are highly flexible and designed for reuse. Business objects within the framework are capable of recognising their environment and interacting with other business objects. They can therefore take part in a wide range of integration scenarios, perhaps even across industrial sectors. They can also easily be specialised to meet the needs of particular organisations or projects. This is of particular relevance to the construction industry where organisational structures and practices may vary from one project to another.

In addition to the broad advantages mentioned above, a component-based approach to integration has several technology driven benefits. Commercial pressures often dictate that existing construction industry software applications satisfy a very broad range of user needs.

This results in large monolithic applications. Component based applications could be assembled on the basis of user needs and would be smaller than existing applications. They would run on hardware which is less expensive and more portable, thereby creating the opportunity to involve construction industry practitioners who have hitherto been excluded. This might, in particular, promote the use of computers on-site. Smaller, component based applications are also easier to distribute over networks, particularly with the advent of interpreted platform independent languages such as Java (Gosling and McGilton, 1995).

3. THE COMMIT PROJECT

Projects in the construction industry are increasingly characterised by large numbers of actors working concurrently at different locations and using heterogeneous technologies. In order to further improve support for computer-integrated construction, project information needs to be conceptually modelled throughout its lifecycle, along with the events that impact upon it. The COMMIT Project aims to address many of the problems surrounding this kind of collaborative work, such as versioning, notification, object rights and ownership. The model also facilitates the recording of the intent behind construction project decisions, thereby providing a complete project history. This paper discusses the distributed implementation of these models, and addresses both software architecture and user interface requirements.

3.1. Versioning Support

Construction projects frequently involve multiple actors making changes and working simultaneously. The primary reasons for providing a structured environment in which such changes are recorded are firstly to provide the client with a complete project history, and secondly to facilitate backtracking. In addition, because actors often refer to previous versions of objects, a structured historical versioning environment reduces ambiguity by standardising version numbers and allowing the examination of previous versions of objects.

3.2. Rights and Responsibilities

As a construction project progresses, actor’s rights and responsibilities with respect to specific construction project objects change. We propose to provide support for the evolution of actor’s roles and rights (according to object version and state) during the entire project lifecycle. Defining roles in this way is essential for effective management of concurrent multi-actor engineering and needs to be tightly integrated with support for versioning.

3.3. Recording Intent

An intelligent construction information management system should record the intent behind construction project decisions. A record of intent is one component necessary for providing the client with a complete project history. It could also be useful if disputes occur, not only for resolving litigations, but also to help actors to see the project from other viewpoints. Intent representation promotes actor’s understanding of the reasons for project changes, which in turn reduces problems arising from misunderstanding intentions. A complete record of project intentions could be useful for decision support in future projects in the form of a Case-Based Reasoning analysis of objects and intentions.

3.4. Notification

Under a knowledge-based system paradigm intelligent object interaction can be embodied by rules concerning objects which are executed by an inference engine. Under an object-oriented paradigm the same idea is represented by a method, belonging to the object but making use of notification relationships with other objects, which is invoked in case of object modification. In either case, the basic idea is the same - changes to a particular object can facilitate changes in, or at least warnings relating to, other objects in the instantiated model.

4. THE COMMIT INFORMATION MANAGEMENT MODEL

In order to provide the necessary concepts and slots for handling multiple actors, versioning, rights, notification and intent, the COMMIT Information Management Model (CIMM) has been proposed (Rezgui et al., 1996 [1]). It is assumed that all project information will be managed and governed according to the concepts of the CIMM, a partial view of which is shown in Fig. 2. Although a complete description is impossible within the context of this paper, an outline of some CIMM concepts is necessary in order to describe the benefits of adopting a distributed approach to their implementation. For a full description of the CIMM, the reader is referred elsewhere (Rezgui et al., 1996 [2].

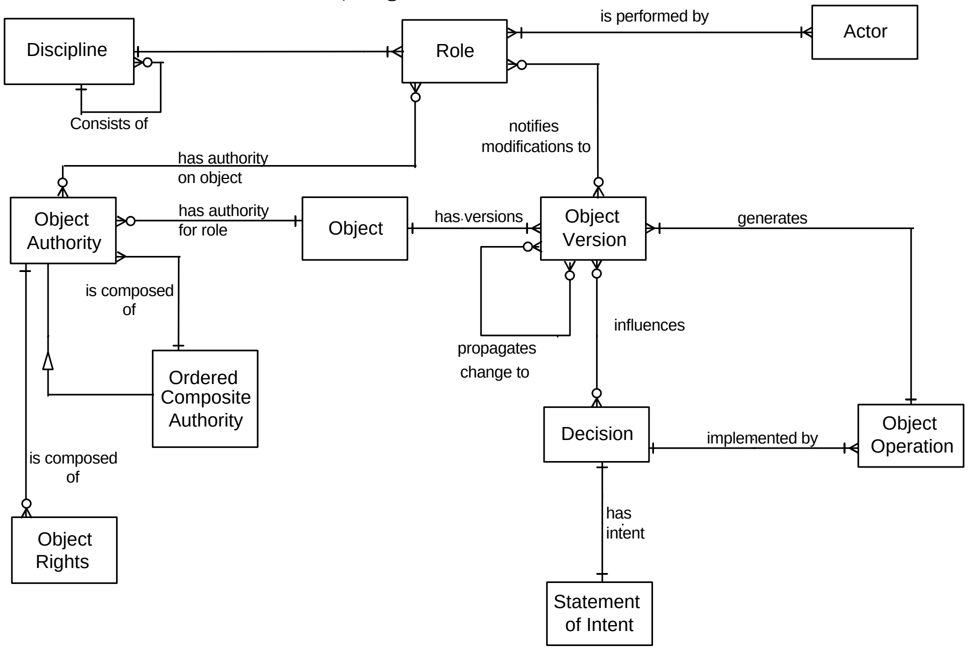


Figure 2. Primary Concepts of The COMMIT Information Management Model (CIMM)

Construction project processes involve numerous actors, each of whom performs a set of roles. An action or sequence of actions which delivers a result (which might be a component of the product) is therefore defined as a *Role.* Each role is defined by specific responsibilities and rights on pieces of project information. As each piece of project information can be concurrently acted upon by one or more roles, one might argue that potential conflicts could occur when several roles access the same object for different purposes. This issue has been rigorously addressed through the use of object *Authorities* that define the rights of roles on specific versions of objects, as described later. Roles can be grouped into *Disciplines* which represent industrially recognised domains. Examples of roles include architectural design and cost estimation; examples of disciplines include civil engineering and architecture. As with conventional stages, the concept of a discipline can be specialised into a set of generic disciplines, each of which can be recursively decomposed into sub-disciplines; for example, at the first level of decomposition the Building Services discipline comprises HVAC *(Heating Ventilation and Air Conditioning),* Electrical and Piping disciplines.

The performance of all project roles invokes operations on the objects which constitute project information. The CIMM specifies whether a particular role has the right to invoke a particular object's operation by means of an *Object Right*. This definition of object rights is one illustration of the benefits of adopting an object-oriented approach to modelling construction project information. Encapsulating operations within objects allows us to define rights in terms of specific operations on specific objects; for example "dimension the wall" or "estimate labour costs". Rights are grouped into sets, each set representing the rights a particular role has over a particular object. Such a set is termed an *Authority*. At a given moment, each object will be related to many such authorities, one for each role in the project. Conversely, each role will have a separate authority for each object in the project. An *Ordered Composite Authority* is an authority that consists of an ordered sequence of other authorities applying to various objects and roles. The use of this concept is illustrated by its specialisation to construction industry approval circuits.

A primary aim of the CIMM is to record the intent behind decisions made by actors during the construction project lifecycle. In addition to the advantages of a model-based (as opposed to implementation-based) approach to version modelling already given, version modelling also complements a record of intent. An *Object* has one or more *Object Versions,* one of which is the current version. Object versions both current and non-current influence a *Decision* - a human conclusion which is recorded by a *Statement of Intent* and effects one or more *Object Operations.* These operations in turn generate one or more new object versions. The intent behind project decisions is thereby recorded not only explicitly by the statement of intent, but also implicitly by the relationships between various object versions and a decision. This model allows the complex chains of modifications and amendments which typify construction projects to be recorded.

Notification is the process by which amendments made to objects are notified to the various roles concerned with that object. Many individuals will be unused to working in an integrated construction environment in which 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 the CIMM making inferences on the basis of project information. It is modelled in a similar way to notification; an object type, object and object version may all have a unique set of objects to which a modification is propagated. Under an expert system paradigm such a set would constitute a rule.

5. A DISTRIBUTED IMPLEMENTATION OF THE CIMM

The COMMIT project has chosen to implement the CIMM as a set of CORBA compliant distributed components which make use of CORBA services. This section describes an architecture that makes this possible, and gives examples of the specialisation of CORBA service objects into construction industry specific CIMM objects. Three strategies for integrated existing construction industry software and CIC systems are also given.

5.1 Architecture

Figure 3 shows the proposed architecture for the COMMIT prototype. The backbone of the COMMIT integrated construction environment is an ORB which manages the low level interoperability between components. The ORB also provides the low level CORBA services necessary for distributed computing.

The CIMM provides a framework within which construction components can interoperate. It defines the concepts necessary handling multiple actors, roles, documents, versioning, rights and notification. It will also enable the intent behind construction project decisions to be recorded, thereby providing a complete project history. It is assumed that all project information will be managed and governed according to the concepts of the CIMM. Components participating in the distributed environment are specialised from CIMM objects, whether they reside independently, in the CIMM user interface, or in legacy applications (via object wrappers).

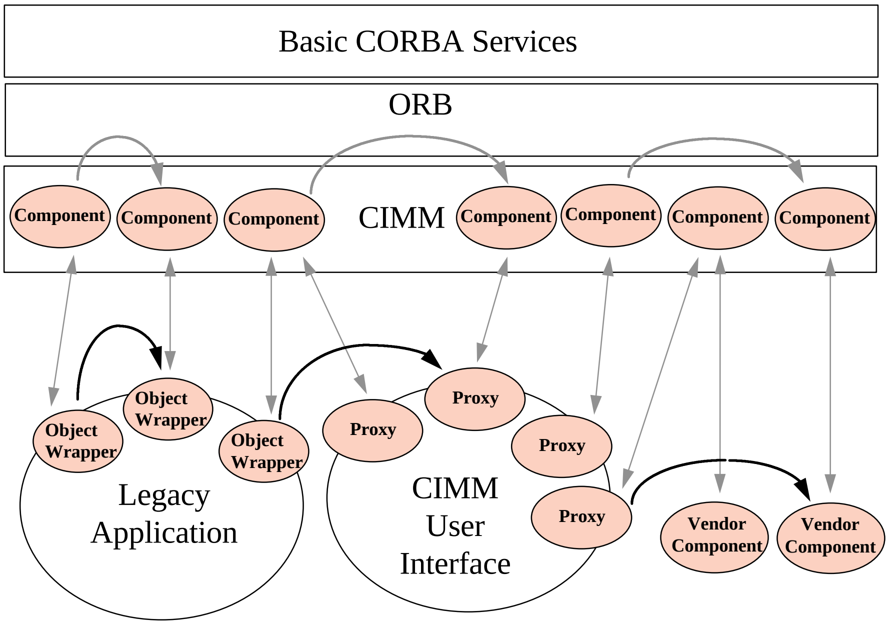


Figure 3. COMMIT Implementation Architecture

5.2. Integrating the CIMM with CORBA Services

The procedure for the basic implementation of objects and relationships in a distributed environment does not differ greatly from their implementation in an object-oriented database. In the CORBA methodology, object interfaces must be described in an Interface Definition Language (IDL); they can then be implemented in the language of the developer’s choice. Distribution of components and interoperation is handled by a commercially available implementation of the CORBA standard ORB. ORBs also provide, to varying degrees, the various basic CORBA services which currently comprise Naming, Events, Life Cycle, Transactions, Concurrency, Persistence and Queries. Relationship, System Management and Security services are in the process of being finalised. CIC projects have tended to expend considerable effort researching and implementing these features, for example, some of the system level elements in OPIS (Froese and Paulson, 1994) or the work of Eastman and Kutay (1991). Although useful results have often been produced, we argue that the implementation of a distributed CIC system can be considerably simplified by re-using the services CORBA provides. Where services do not match the particular requirements of the construction industry, existing services may be specialised or, in the worst case, new services provided. Research efforts can focus on issues more directly related to construction while at the same time re-using the efforts of technologists devoted purely to computing.

The concepts within the CIMM are specific to the construction industry and facilitate the management of information in a distributed CIC environment. The COMMIT implementation architecture proposes that concepts in the CIMM make use of CORBA services by specialisation. In this way, the CIMM provides a bridge between basic CORBA services and CIC systems - in CORBA terms, a vertical market common facility. All CIMM objects inherit from three basic CORBA objects; *Persistent Object*, *Transactional Object* and *Life Cycle Object*. Clearly implementing the CIMM requires the use of the CORBA persistence service; CIMM objects must remain in the construction environment after the programs which initiated their creation have terminated. The *Transactional Object* provides consistency, concurrency and recovery in a distributed object system. While the underlying CORBA implementation of services may often be complex (as in the case of transactions and concurrency), this complexity is largely hidden from the developer. The *Life Cycle Object* provides services for creating, copying and deleting objects while maintaining referential integrity constraints.

The CIMM also contains several concepts that require the use of CORBA collection service[4]. For example, an *Object* is related to the set of *Object Authorities* (one for each *Role*). These will need to be stored in a collection class which can be traversed with an iterator, for example, if the information manager is specifying the rights that different roles have over core objects. In addition to the CORBA objects that all CIMM objects inherit from, there are several CORBA services which CIMM objects make use of through method invocations. All CIMM Objects make use of the *Naming Context* object (provided by the CORBA naming service) in order to give CIMM objects a unique system wide identity by which they can be located.

The CORBA event service allows objects to register their interest in specific events by invoking an appropriate method of the *Event Channel* object. This service allows the invocation of a distributed object's method to cause the transparent invocation of a CIMM object's method. This is used to implement notification and rights access in the CIMM by decoupling the communication between objects. In the CIMM model, each object has a list of roles to notify in case of modification (which is maintained by a CORBA collection class). For each element in the collection a message is sent to the CORBA *Event Channel*, which registers the role's interest in this particular object. Any subsequent invocations of the object's method cause a message to be sent to the role object. This message can then be

[4] The CORBA Collection service is not finalised at the time of writing. Basic collection services are however provided by the CORBA Query service.

acted upon - the role object creates an appropriate message box on the workstation of the actor performing that role.

The development of other CORBA services is ongoing; we expect the examples of re-use given here to be the first of many. It is expected that CIMM objects will re-use additional CORBA services as they are finalised, particularly in the area of rule-based, trader and system management services.

5.3. Integrating Existing Software with the CIMM

For the CIMM to be effective, it should be possible to integrate it with both existing CIC systems and legacy applications. This existing software could then take advantage of the CIMM's information management functionality. Three categories of existing software and systems that have the potential to be integrated are described below.

One of the most favoured methods of computer integration in the construction industry is to use shared object-oriented database (OODB) technology (Galle, 1995). This type of CIC system can be integrated using inheritance; concepts in the database model are specialised from concepts in the CIMM. This has the effect of converting the CIC system to a set of CORBA compliant distributed components which are managed by the CIMM. Although the existing CIC models must be translated to IDL interfaces (a process which can be made easier by the use of CASE tools), much of the original implementation and integration code can be retained. This approach has advantages - the full functionality of the CIMM is utilised to produce a system which is easily extended by adding new CORBA compliant components. However, potential difficulties arise in cases where the existing CIC functionality overlaps with the CIMM. This strategy has been chosen to test both the feasibility of integrating the CIMM and its effectiveness for managing information. The existing CIC system to be integrated has been developed by a companion project (the OSCON Project) and uses an integrated core model and the ObjectStore object-oriented database (Tracey et al., 1996).

It is not unreasonable to describe most construction industry software packages as monolithic legacy applications. For such systems, the integration strategy is similar to that adopted by CIC systems based on OODBs. Object wrappers for the legacy application are modelled and coded, but, rather than being in the language of the OODB, they are described in IDL. The IDL object wrappers can then be specialised from CIMM concepts. Again, this uses the full functionality of the CIMM and effectively converts the application into a set of distributed components.

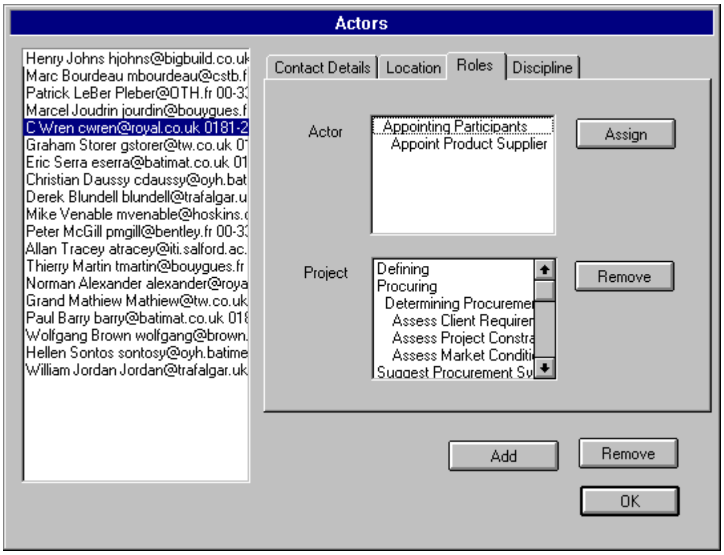
Applications based on distributed components are still in their infancy, but will undoubtedly appear over the next few years. CORBA's DII (Dynamic Invocation Interface) makes it possible to integrate such applications at run time (that is, with no code changes or recompilation). It would, for example, be possible for the CIMM to use the DII to query a component, and subsequently control actor's rights to perform operations on that component. Full CIMM functionality would, however, only be available if components were specialised from CIMM concepts.

6. THE CIMM USER INTERFACE

In order to make a distributed CIC system effective, it is necessary to provide users with a uniform user interface through which they can access project information. This information consists of not only the usual concepts found in construction integration models (for example, buildings, walls, and tasks) but also the CIMM's information management data (for example, roles, actors, rights and responsibilities). The CIMM organises project information into discrete units accessible by a project's actors. It also contains the basic concepts for implementing a distributed Electronic Document Management (EDM) system. The CIMM's user interface reflects this organisation.

Ideally the CIMM user interface would be implemented as a set of distributed user interface components. Unfortunately, there are two reasons why current technology does not offer sufficient support for this approach. Firstly, distributed user interface components (for example OpenDoc or OLE) are not as yet sufficiently mature or platform independent. Secondly, current limits on network bandwidth render the transmission of data intensive user interface components impracticable[5]. The CIMM user interface has therefore been implemented as a standalone application which can interact with the distributed components of the CIMM through the use of object proxies. The software is being developed using Visual C++ Version 4.0 for the Windows NT and Windows 95 platforms.

Fig. 4. shows the CIMM user interface panel in which the project's actors are described and managed. Through this interface any actor may browse the details of other actors, or, in the case of the information manager, specify project participants. In addition to describing the contact information, physical location and main discipline of each actor, the panel describes the roles each actor performs. Within the distributed CIC, it is therefore possible to see exactly who is responsible for carrying out a given activity.



Henry Johns hjohns@bigbuild.co.uk
Marc Bourdeau mbourdeau@cstb.f
Patrick LeBer Pleber@OTH.fr 00-3
Marcel Joudrin jourdin@bouygues.f
C Wren cwren@royal.co.uk 0181-2
Graham Storer gstorer@tw.co.uk 0
Eric Serra eserra@batimat.co.uk 01
Christian Daussy cdaussy@oyh.bat
Derek Blundell blundell@trafalgar.u
Mike Venable mvenable@hoskins.
Peter McGill pmgill@bentley.fr 00-3
Allan Tracey atracey@iti.salford.ac.
Thierry Martin tmartin@bouygues.fr
Norman Alexander alexander@roya
Grand Mathiew Mathiew@tw.co.uk
Paul Barry barry@batimat.co.uk 01
Wolfgang Brown wolfgang@brown.
Hellen Sontos sontosy@oyh.batime
William Jordan Jordan@trafalgar.uk

Actor: Appointing Participants, Appoint Product Supplier

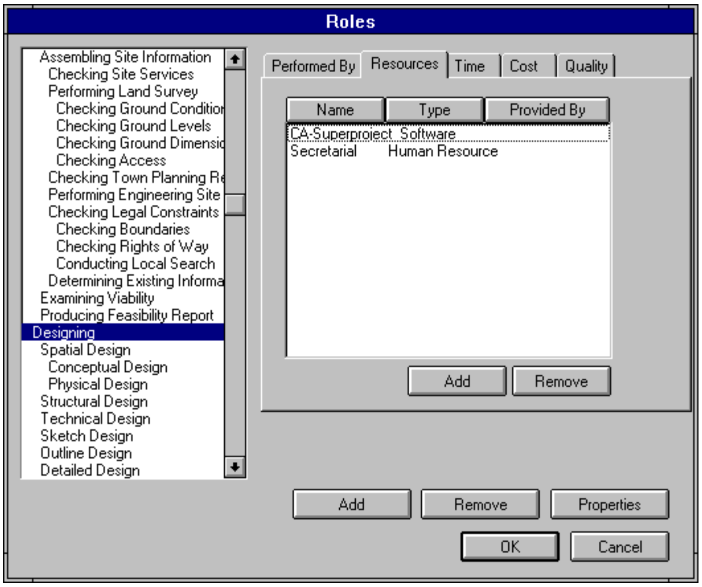
Project: Defining, Procuring, Determining Procureme, Assess Client Requirer, Assess Project Constra, Assess Market Conditi, Suggest Procurement Sy

Figure 4. CIMM User Interface - Actors

In addition to managing the actors involved in a project, it is also important to manage the roles or tasks they perform (Fig. 4.). For each role, the CIMM user interface describes the actors who will be involved, the resources this role uses (hardware, software, equipment or

[5] Particularly when one considers that most ORBs currently use the Internet to communicate with each other.

human), the timeframe for the performance of the role, the costs of the role and any quality standards that need to be adhered to.



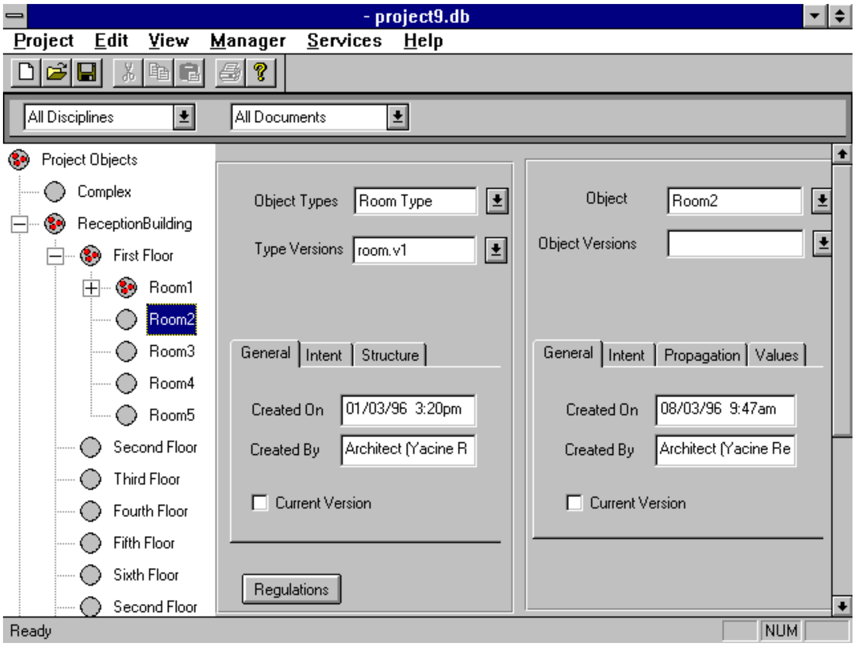
Assembling Site Information
Checking Site Services
Performing Land Survey
Checking Ground Conditio
Checking Ground Levels
Checking Ground Dimensio
Checking Access
Checking Town Planning R
Performing Engineering Site
Checking Legal Constraints
Checking Boundaries
Checking Rights of Way
Conducting Local Search
Determining Existing Informa
Examining Viability
Producing Feasibility Report
Designing
Spatial Design
Conceptual Design
Physical Design
Structural Design
Technical Design
Sketch Design
Outline Design
Detailed Design

Name	Type	Provided By
CA-Superproject	Software	
Secretarial	Human Resource	

Figure 4. CIMM User Interface - Roles

Fig. 5. shows the user interface panel through which actors browse managed project information[6]. This information includes the building core concepts (walls, beams, etc.) which constitute the project information base - it is the CIMM's view of the distributed components which make up the whole CIC environment. Of course, much of this information will be created and amended through the use of construction industry software applications. It is, however, important to allow actors to interact with information in a way which is application independent. The management properties of information must also be made explicit. For example, a given object has a creator and a creation date and an intent behind its creation. Note that the panel describes objects at both the type and instance level; the prototype supports schema evolution through the creation of new versions of object types. This is described more fully elsewhere (Rezgui et al., 1996 [2]).

[6] Although in this example objects have been organised according to physical decoposition, this is not necessarily the case and organisation principles will vary from one project to another.



- project9.db

Project Edit View Manager Services Help

All Disciplines | All Documents

Project Objects
- Complex
- ReceptionBuilding
  - First Floor
    - Room1
    - Room2
    - Room3
    - Room4
    - Room5
  - Second Floor
  - Third Floor
  - Fourth Floor
  - Fifth Floor
  - Sixth Floor
  - Second Floor

Object Types: Room Type
Type Versions: room.v1

General | Intent | Structure

Created On: 01/03/96 3:20pm
Created By: Architect (Yacine R
Current Version: ☐

Regulations

Object: Room2
Object Versions:

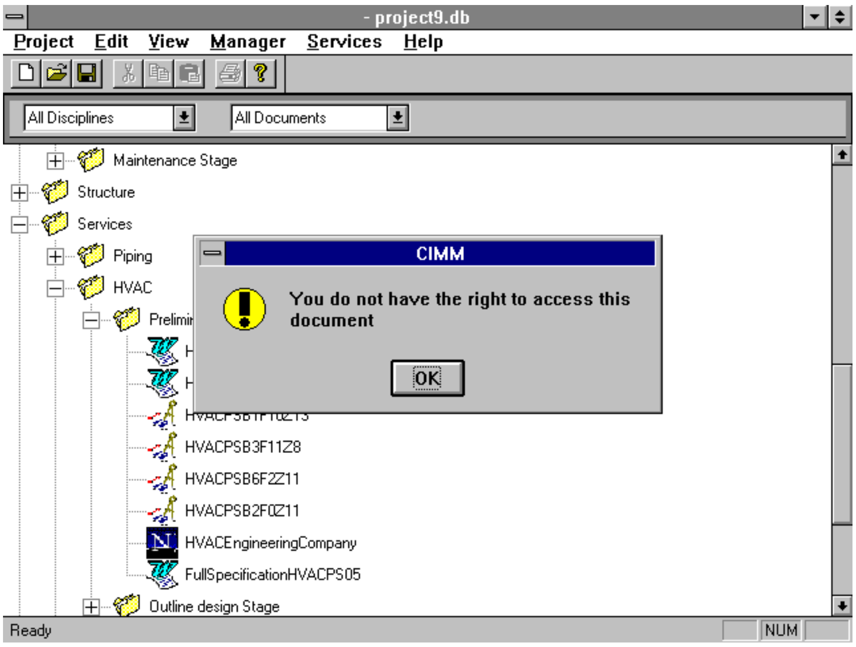
General | Intent | Propagation | Values

Created On: 08/03/96 9:47am
Created By: Architect (Yacine Re
Current Version: ☐

Ready NUM

Figure 5. CIMM User Interface - Objects

One of the aims of the COMMIT project is to provide a migration path for the introduction of intelligent distributed CIC. Project participants should be able to access information in a familiar way, and current practices evolved to fully take advantage of the benefits of integration and distribution. To this end, the CIMM models include the concept of interpreted object, which may represent a document or other interpretation (e.g. multi-media) of a core concept. Fig. 6. shows the CIMM user interface for accessing interpreted objects. Treating documents as another form of managed information has several additional benefits; for example, all the project's participants share documents and document organisation and access to documents may be controlled according to rights.



You do not have the right to access this document

Figure 6. CIMM User Interface - Documents.

CONCLUSIONS

To conclude this paper we would like to stress the potential of distributed computing in computer integrated construction. The paper has outlined the benefits of adopting distribution standards such as CORBA, and described a framework for distributed computer integrated construction. In this framework the COMMIT Information Management Model acts as a bridge between objects in an integrated construction system and CORBA services. In the CORBA methodology, the CIMM is equivalent to a business object facility for the construction industry.

This paper has presented a distributed implementation of the CIMM, and shown how CIMM objects can be specialised from CORBA service objects. Techniques for integrating the CIMM with existing CIC systems and legacy applications have been presented. The CIMM user interface, which allows project participants to manage distributed project information, has also been described.

The work presented here is ongoing; the prototype demonstrating the ideas described in this paper is currently being refined. In the longer term, it is hoped that the COMMIT project, which is supported by a UK steering group comprising regulation bodies, research institutions and industrials, will successfully demonstrate the technical, economic and sociological benefits that would be gained by adopting an approach to information integration founded on an object-oriented information management model and distributed object technology. While the work presented here is specific to the construction industry, many of the ideas behind it are generic. Future work also includes generalising the CIMM across industries.

ACKNOWLEDGEMENTS

The authors would like to thank Dr G. Aouad, Professor M. Betts, Dr J. Yip and Professor B. Lawson for their helpful advice given during the preparation of this paper. The authors also wish to acknowledge the financial support of the EPSRC, and the assistance provided by members of the steering committee throughout the duration of this research project.

REFERENCES

Brocksmidt, K. (1995), Inside OLE-2, Second Edition, Microsoft Press.

OMG (1995), *The Common Object Request Broker: Architecture and Specification*, OMG.

OMG (1995), *The Common Object Request Broker: Services*, OMG.

Orfali, R., Harkey, D. and Edwards, J. (1996), *The Essential Distributed Objects Survival Guide*, John Wiley & Sons.

Howard, H.C. (1991), Linking design data with knowledge based construction systems, CIFE Spring Symposium 1-24.

ATLAS (1992), Architecture, Methodology and Tools for Computer Integrated Large Scale Engineering - ESPRIT Project 7280, Technical Annex Part 1, General Project Overview.

Dubois, A.M., Flynn, J., Verhoef, M. and Augenbroe, G. (1995), Conceptual Modelling Approaches in the COMBINE project, presented in the COMBINE seminar, Dublin.

Bjork, B.C. (1993), The RATAS Project - An Example Of Cooperation Between Industry And Research Toward Computer Integrated Construction, *ASCE Journal for Computing in Civil Engineering*.

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

OMG (1996), Common Facilities RFP-4: Common Business Objects and Business Object Facility, OMG TC Document Number 96-01-04.

Gosling, M., and McGilton, H. (1995), The Java(tm) Language Environment: A White Paper, Sun Microsystems.

Rezgui, Y., Brown, A., Cooper, G., Yip, J., Brandon, P. and Kirkham, J. (1996) [1], An Information Management Model for Concurrent Construction Engineering, to be published in *Automation in Construction*

Rezgui, Y., Brown, A., Cooper, G., Aouad, G., Kirkham, J. and Brandon, P. (1996) [2], An Integrated Framework for Evolving Construction Models, to be published in the *International Journal of Construction IT.*

Froese, T. and Paulson, B. (1994), OPIS: An Object Model-Based Project Information System, *Microcomputers in Civil Engineering*, **9** 13-28.

Eastman, C. and Kutay, A. (1991), Transaction Management in Design Databases. In *Concurrency in Engineering Data Management,* eds. D. Sririam R. Logcher and S. Fukuda, Springer, New York.

Galle, P. (1995), Towards Integrated, “Intelligent” and Compliant Computer Modeling of Buildings, *In Automation in Construction*, (**4**).

Tracey, A., Child, T., Aouad, G., Brandon, P. and Rezgui, Y. (1996) Developing Integrated Applications for Construction: The OSCON Approach, International Conference on Computing and Information Technology for AEC, Singapore.

Rezgui, Y., Brown, A., Cooper, G., Brandon, P. and Betts (1996) [3], M., Intelligent Information Versioning Support In The Context Of Collaborative Construction Engineering, International Conference on Computing and Information Technology for AEC, Singapore.