Managing Dynamic Life-Cycle-Dependent Building Objects in a Distributed Computing Environment

Kwok-Keung Yum
Division of Building, Construction and Engineering,
CSIRO,
Highett, Melbourne, Vic 3190, Australia
Kwok-Keung.Yum@dbce.csiro.au

Robin Drogemuller
Division of Construction Management
James Cook University of North Queensland
Townsville Qld 4811, Australia
Robin.Drogemuller@jcu.edu.au

Abstract

Building and construction business processes involve various stakeholders located in different locations over the life cycle stages of buildings. In order to develop any automated solutions for improving these processes, there is a need to develop a flexible framework that can deal with two important issues:

(a) common data for sharing and access across networked computers over time;
(b) management of data for the protection of rightful access and for the reduction of information overloading.

While there had been many attempts in the past to provide "logical" industry information frameworks for integration, they proved to be difficult for a "fragmented" industry like the construction industry to adopt. For any integration framework to be useful for collective implementation, this paper argues that the AEC (Architecture, Engineering and Construction) business/enterprise views should be captured in an open interoperable architecture. The gist of the business/enterprise view point is that users can *play* various roles (owner/operator, architects, etc.) *at* various life cycle stages (briefing, conceptual design, detail design, construction, operation, etc.); and *through* these roles, users can connect to various building model servers and various software tools. Embedded in this business model is a simple and yet powerful threaded relationship "users → business roles and life cycle stages → tools and data". It is powerful because it supports a generic data management regime: *users* can select various permissible *roles* in various *life cycle stages* to access legitimate *tools* and *data* within various building model servers. It is simple because it is compatible with today's network operating system login procedures and the password protection mechanism of files and folders. As people, end users and developers alike are familiar with the basic paradigm of "data manipulation through software tools". The above two features of the business view reinforce each other for gradual acceptance by the AEC industry. What is needed is a critical mass from an industry alliance to initiate a feasibility study of the interoperable architecture, its business views and other supporting view points (information views, engineering views and technical views) for a quick demonstration.

This paper also presents some usage scenarios demonstrating how a user logs on as a designer and connects to a design tool accessing data objects in a building model server.

Keywords

Modeling methodologies and technologies; discipline/phase specific models; interoperable architectures, business models, information models.

Introduction

Bjork (1995) summarised the research on building product modelling from thev1970s to 1995 as focusing on conceptual schema development and prototype implementations. During this period, the *de facto* research project stages of development consisted of:

- Choice of a particular information modelling methodology.
- Gathering of empirical knowledge about the building information model.
- Definition of information structures as formalised conceptual schemas.
- Experiments with prototypes to test the conceptual schemas.

In the early 90s, Bjork noted, it was recognised within the research community that it was virtually impossible to define and validate a comprehensive product data model catering for all needs of all parties involved in all life cycle stages (e.g., design, construction and maintenance) in one single research project. For the R&D in data modelling to be carried out, the work was re-focused on the definition and verification of less ambitious product data models which catered for the need of applications in limited domains only. The terms, aspect models (information structures particular to a specific discipline for a particular life cycle stage, cf Bjork 1995) and core model (generic information structures that are common to most disciplines and phases in design, construction and maintenance of buildings, cf BRE 1997) were used to described work that is focused on these limited domains. However, even after the restriction of domains, the complexity and scalability issues remained and few research prototypes have reached a stage of maturity for testing in real design/construction projects (cf the European COMBINE projects).

The complexity and scalability issues originate from the technical problems of maintaining compatibility of the aspect or core models for the information structures common to a wide range of applications. From the technical side, it makes sense to restrict the scope of investigation by breaking down the research into smaller domains. However this is not the only problem that is facing today's researchers in building data modelling. Adding to the complexity problem there are also financial considerations. In the current R&D climate, unless the research community, the granting agencies and the industry are *convinced* that their projects will form one of the jigsaw pieces that fit in a whole picture, there is little or no incentive for them to channel their research investment in this direction.

There is no doubt about the merit and the contribution of the "conceptual industry frameworks" research paradigm in pioneering work. However, as the world moves on to take up global marketing issues, the shortcomings of the "conceptual framework approach" are becoming apparent:

(a) The uptake cost of a comprehensive conceptual industry framework is prohibitively high;
(b) there is no guarantee that the other parts of the world will take up the same framework and implement it in a complementary manner;
(c) the end users are left in the cold for a long time before seeing any tangible benefits.

This paper presents an "Interoperable Architecture" approach as an alternative to the conventional "Conceptual Industry Frameworks" approach. The design principles of the architecture is aimed at tackling the above mentioned problems by providing:

1) Object-Oriented Analysis to be driven by business models, and from there deriving information models, computational models, etc.
2) Distribution of software components over various parts of the computer networks
3) Decoupling of software components for separate development over time

4) Separation of concerns between applications and the distributed environment, as well as between the generic data management/control part and the application-specific part of the architecture.

Point (1) above keeps the core business of conceptual modelling within the development of the architecture. Point (2) is relatively new; its purpose is to support the need for developing distributed solutions for the AEC community. Points (3) and (4) facilitate incremental development over time.

These design principles are much in align with the TINA-C's (Telecommunication Information Network Architecture Consortium, cf TINA-C 1997) TINA architecture, which is based on the next generation of distributed processing standard RM-ODP (ISO/IEC DIS 10746, Reference Model of the Open Distributed Processing, cf RM-ODP 1997)

From Frameworks to Interoperable Architectures

One of the major rationales behind the design principles of the Interoperable Architecture approach is ensure the prominence of the various stakeholder roles in AEC processes and to formalise them in a modeling formalism such as EXPRESS-G. This will enable stakeholders, in particular those who are prepared to fund various projects, to gain a better understanding of what they can expect in the short term and in the longer term. Formalising the business roles can produce a number of inter-related benefits for the continuation of work:

- The formalisation of business roles separates business issues from other technical issues and thus makes it easier for project and funding managers to consider the merits of the projects. This is an important factor to motivate stakeholders to pay for the cost of development in the short term and in long term. Without it, the development work may not even start, let alone be sustainable.
- The formalisation of the business models, information models and computational models enables technical work to be divided and coordinated for both short-term and long term delivery and planning.
- The abstraction and formalisation of these models facilitate further development irrespective of who is developing it using whatever underlying technology (standards, languages, programs, resources, networks).

Through these formalisms within the interoperable architecture, multiple industry partners can work together in an accountable manner while appreciating the significance of their own contributions to the whole picture. Similar strategies for open integration have been developed since the early 90's, e.g., the TINA initiative (see TINA-C 1997) and the IAI (International Alliance for Interoperability) initiative, (see IAI 1997). International experience in the past 5 years indicates that the approach of "Interoperable Architecture" supports wider industry participation and collaboration than the traditional approach of conceptual industry framework. Most importantly, this approach supports incremental development with continuity over time and by various organisations.

Connection Architecture

The connection architecture presented in this paper is an interoperable information architecture which abstracts out the relationship between user (as a person or a legal entity), services (a telecomputing term which refers to what AEC people call applications or tools), building model servers (also called online building data repositories) and resources (e.g.

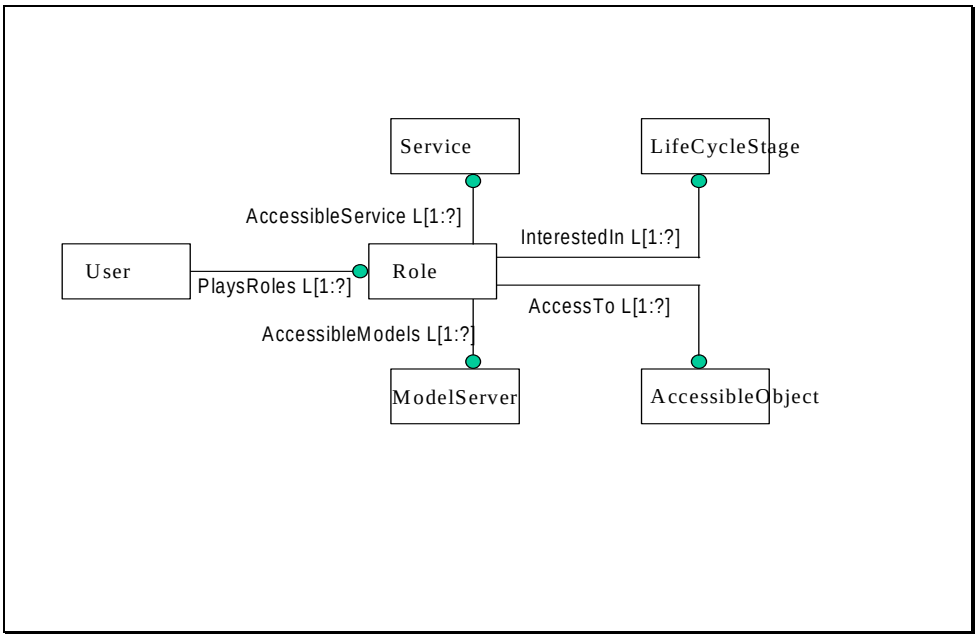

Figure 1 : Top level EXPRESS-G diagram for User-Role-Model Server-Service-Data Object relationship

graphical representations of building data, emails, which assist proper delivery/execution of services). This architecture is conceived based on the following rationales:

I. Since any individual service (AEC application) can operate on various building model servers, there is a need to separate the service from the data it operates on.
II. Services, objects in the building model server, and resources have no way of knowing who and in what role is connecting to them unless there is a user login-connection mechanism to inform the services. Only through such a connection mechanism can the appropriate access by users to various building data objects be protected.
III. It is economical in terms of user training and development cost to have multiple services to share the same connection architecture rather than to have each service providing its own way of linking to building model servers and resources.

Figure 1 is the top level information model showing relationships between a user (an individual or a legal entity), roles (as an actor or as a stakeholder in a building project), services, resources and building data objects using the EXPRESS-G formalism (Spiby 1991). For clarity inverse relationships between entities are not shown in this diagram.

The business model above enables a user, as an individual or a legal entity, to log on to an implementation of the connection architecture, by selecting from a list of roles they can play (owner/operator, architects, etc.). During the logon session, the user can specify the life cycle stages they are interested in (e.g., briefing, conceptual design, detail design, construction, operation, etc.), together with the AEC services and building model servers they want to connect to. It is the responsibility of any implementation of the connection architecture to verify such claims. When the user claims are verified, the system allows the user to connect to respective services and model servers and utilise resources for the access of building data objects for their applications.

Embedded in this business model is a simple and yet powerful threaded relationship "users → business roles and life cycle stages → tools and data". It is powerful because it supports a generic data management regime: *users* can select various permissible *roles* in various *life cycle stages* to access legitimate *tools* and *data* within various building model servers. It is simple because it is compatible with today's network operating system login procedures and the password protection mechanism of files and folders. As people, end users and developers alike, are already familiar with the basic paradigm of "data

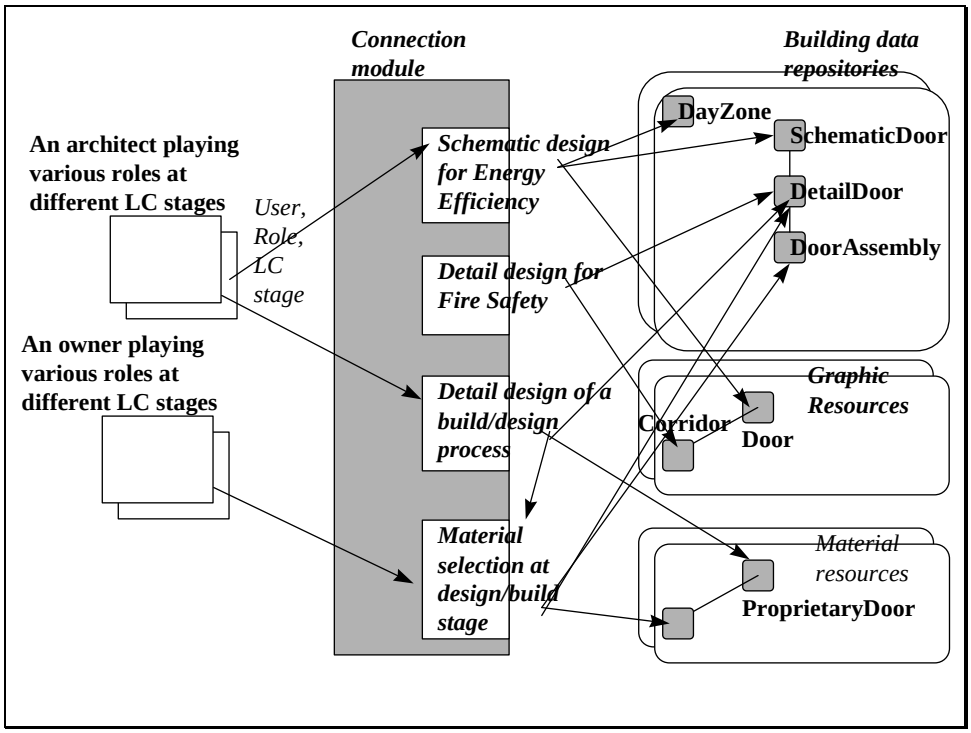

Figure 2: An AEC services connection architecture.

manipulations through software tools", the above two features of the business view will reinforce each other for gradual acceptance by the AEC industry.

The above business model is separated from, and yet supported by the following viewpoints:

- Information viewpoint specifying the information required for various business roles without concerns of the design of the functions that actually manage the information. Computation viewpoint can be specified by using an object modelling formalism such as EXPRESS-G.
- Computation viewpoint which describes the system as a set of interacting objects, providing various functions needed by the system. The computational view point encapsulates the necessary functions in the system without concerns of the distribution of objects on the networked computers. The formalism used in specifying computation viewpoints can be IDL (Interface Definition Language) or its variant, e.g. MIDL.

For such a computational viewpoint to be transparent, the architecture should be developed over some distributed computing environment (e.g. CORBA and DCOM)[1] so that the services, objects and resources can be hosted on various networked computers. This also allows user connections and their business activities (such as subscriptions, request for services, and billing) to be managed remotely.

Figure 2 shows the key components of the generic service connection architecture for AEC integration purposes, which is similar to existing networked OS (operation system) service connection models. The middle part of the figure represents a connection module. The connection module provides two major functions: (a) allowing the user to log in and connect to various services and model servers; (b) allowing various AEC services (applications) to plug in, either locally or remotely. On the right hand side, there are building model servers and graphics and material resources. Graphics resources reside on

[1] CORBA, Common Object Request Broker Architecture standards developed by OMG. DCOM, Distributed Component Object Model standard developed by Microsoft.

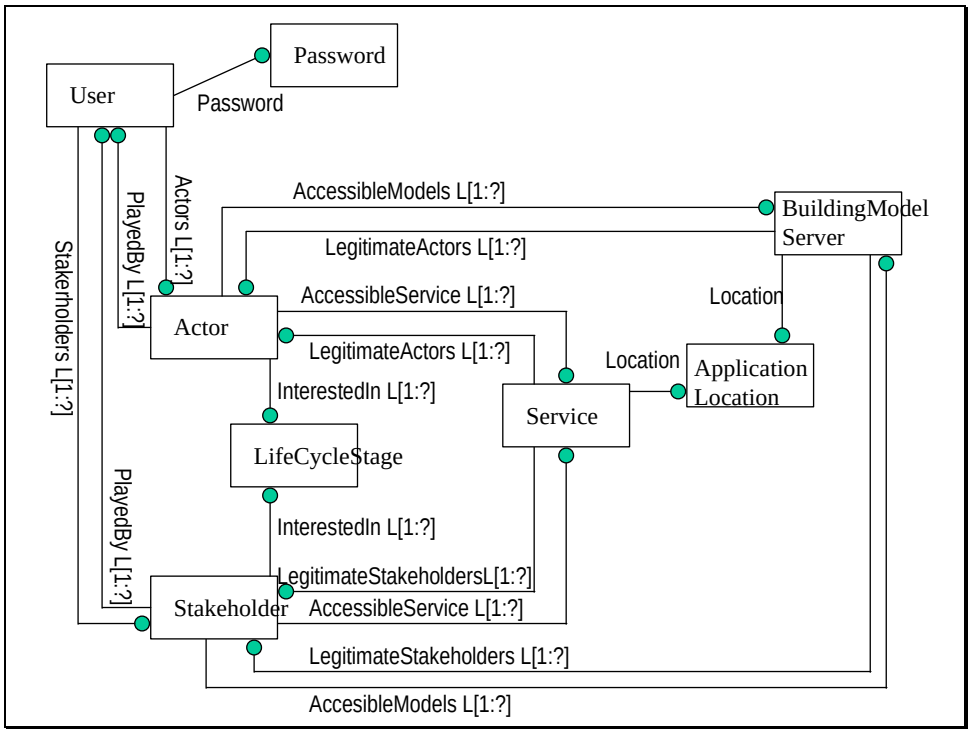

Figure 3: Model of the connection module (EXPRESS-G diagram)

the user's local computer in order to provide efficient display of the forms of design; however, both the building model servers and the material resources can be either on the local computer or on remote networked computers.

The basic concept of the architecture is to have the connection module sitting between the user and the applications and data they may wish to connect to, so that the user's password, permitted roles, user's choice of services and data repositories can be registered, validated and then passed to the applications.

Information Viewpoint

The data model (Figure 3) of the connection module reflects the relationship between the user, their roles as an actor (functional participants in a project who acts on the object, e.g. the architect who designs a schematic wall is the actor of the schematic wall object) and stakeholder (participants of a project who have interest in knowing what has been done to the object in order to make decisions, e.g. the re-design of the schematic wall object will have an impact on the builder, so the builder is a stakeholder of the schematic wall object.), the services they connect to, and the building model servers they want to access.[2] The model specifies that a user can play different roles of actor as well as stakeholders; each role of actor or stakeholder can be interested in various life cycle stages. Each role of actor or stakeholder has the access right to a number of services and building model servers.

Typical subtypes of the actor object type include owner, schematic designer, detail designer, construction project manager, facility manager, etc. Typical subtypes of this stakeholder type include owner, schematic designer, detail designer, construction project manager, facility manager, etc. All these subtypes are not shown Figure 3.

In any implementation of the connection architecture, the data as defined in the above model will be stored in the connection module. Simple data management tools are

[2] In this paper the discussion of the resource requirements is skipped. For a consideration of resources, we need to incorporate extensive interoperable information infrastructures, which is premature at this stage.

available to make changes to the data dynamically to reflect the changing roles of users during the project life (e.g. an architect may leave a project team and a replacement may be found 1 week later.) Only trusted people are permitted to run such tools.

Life Cycle Dependant Objects and Their Access Rights

For data sharing to be effective in improving communications in a "fragmented" AEC industry with distributed expertise, the following factors must be considered: (1) various stakeholders (including sharing for the same person playing different roles and sharing between various disciplines), (2) various geographical locations, and (3) time (including sharing between life cycle stages, e.g. briefing, schematic design, detail design, construction, operation and facility management, etc.) This theme of sharing has been echoed by many building research scientists in the area of integration frameworks (e.g. Rezgui *et al.* 1996). Adding to the consideration of sharing, this connection architecture examines the issues of data ownership for the protection of project stakeholders' proprietary data and for the reduction of information overloading.

The information requirement for sharing data across disciplines and life cycle stages draws a logical design for centralised data (e.g. ISO 130303 part 106, see BRE 97, and Industry Foundation Classes (IFCs), see IAI 1997). However, not all sharing is welcome in the AEC industry. Data sharing is designed for the benefit of two ends of communication: the source end represents the player who wants to provide information; and the receiver end represents the player who wants to receive information. From the source's point of view, they want to send the right information to the right receiver, and they want protection against unwarranted access in order to protect their own vested interest during collaboration (security). From the receiver's point of view, they only want to see interesting and relevant information (differentiation of views). Since these issues have not been included in the agenda of the Building Construction Core Model (BCCM), for the secured and differentiated data sharing to happen, building data need to be structured beyond the current scope of BCCM.

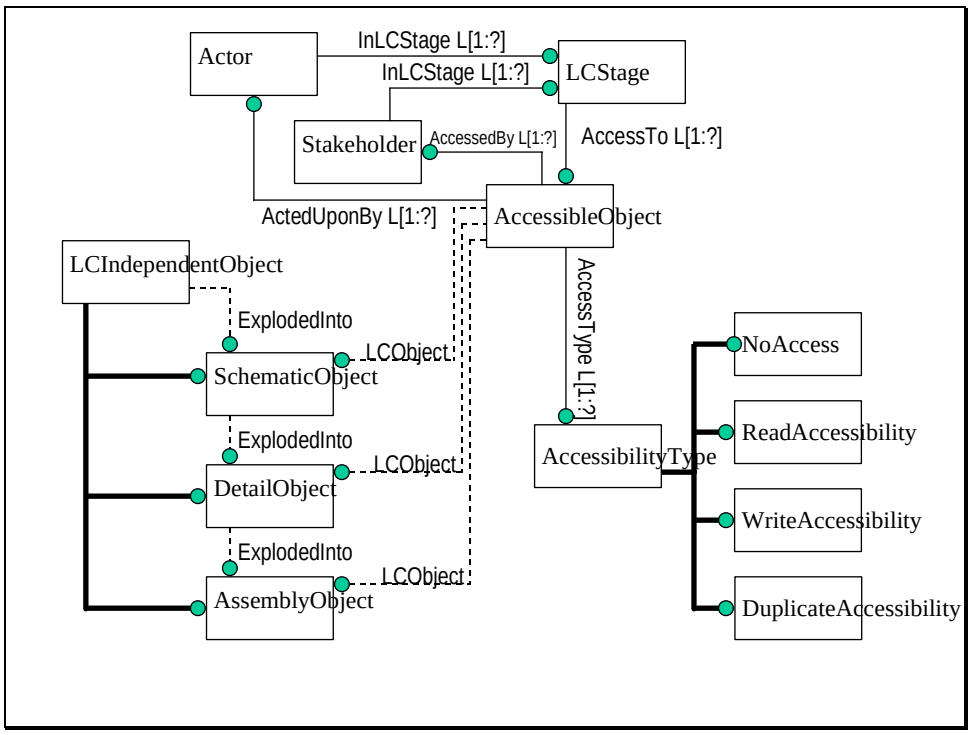

Figure 4: Model of Life cycle dependent objects (EXPRESS-G diagram)

The following is a brief description of the object management model for the building model server. The implication of the data management model to business practice can be found in a separate paper (Yum & Salomonsson 1997). In the building data management model, there are four object types that are critical for the management of accessibility to objects for the purpose of secure and differentiated data sharing (Figure 4).

I. The **actor** object type represents the role of a user who instantiates and makes changes to a piece of building data. Typical subtypes of the this actor object type include owner, schematic designer, detail designer, construction project manager, facility manager, etc. (these subtypes are not shown in Figure 4). A number of life cycle stages can be the concern of an actor (e.g. an architect can act on building objects designed in both schematic design and detail design stages).

II. The **stakeholder** type represents the role of a user who has interest in knowing what has been changed. Typical subtypes of this stakeholder type include owner, schematic designer, detail designer, construction project manager, facility manager, etc. (these subtypes are not shown in Figure 4). A number of life cycle stages can concern a stakeholder (e.g. an owner has vested interest in knowing all things happening in all life cycle stages).

III. The **AccessibeObject** type represents objects that can be "accessed". The accessibility types are differentiated into four subtypes: NoAccess (object and its properties cannot be accessed), ReadAccessibility (object and its properties can be read), WriteAccessibility (object and its properties can be altered), DuplicateAccessibility (object and its properties can be duplicated in the client's (user's) computer.) Note that the specification here describes only the *permissions* for the access of the AccessibleObject; it does not specify how access is performed. The actual action of access (e.g. dimensioning a floor) will be performed by the application tool used by the user, which is not considered by this data model. Each instance of the AccessibleObject contains: (a) the actor object which specifies which role can act upon the object, (b) a list of stakeholder objects, which specifies which roles have interest in knowing changes, (c) the actual life cycle dependent object (see below) with which the AccessibleObject instance is associated, (d) a list of AccessibilityType objects, which specifies the permissions of access.

IV. The **LCIndependentObject** type represents building objects that contain properties that are independent of life cycle stages. For example, an LCIndependentWall object may contain its dimension property which has to be known by all project participants over the span of almost all life cycle stages. The LCIndependentObject type explodes into various "exploded" objects (types) which are specific to certain life cycle stages. At present, for the sake of experimentation, three "explosion" types are considered: (a) SchematicObject type represents objects that are specific to the schematic design stage; (b) DetailObject type represents objects that are specific to the detail design stage; (c) AssembyObject type represents objects that are specific to the construction stage. For example, AssemblyWall object with its graphical representations (2D and 3D) is most meaningful in the construction stage and not in others; SchemticWall object with its 2D geometry would be most meaningful in the schematic design stage. Also, if an HVAC designer wants to create data objects that are specific to their indoor energy modelling method, they may want to keep these objects away from potential competitors, since these objects and data are their proprietary secrets. It would then be appropriate for the HVAC designer to create life cycle dependent objects at the schematic design stage and put NoAccess values to respective AccessObject instances against other stakeholders except themselves.

In implementations of a model server, building objects in the building model server and resources have no way of knowing who is connecting to them and in what role. Such information is passed to them through the connection module if the user is playing a legitimate role for an admissible life cycle stage. In this way the access rights of various building objects be protected.

Usage Scenario

A typical usage scenario is that a user, playing the role of an architect, logs on to a session using an auto design tool on a building model server. For this purpose, the user opens up a user-interface of the connection module from the local computer (Figure 5). The user then types in his/her login name and password as in many network sessions. After the login name and password are verified, the connection module shows the roles and the life cycle stages that the user is permitted to access. The user then selects from the permitted lists a role s/he can play (in this case, an architect) and a life-cycle stage s/he is interested in (in this case, the detail design stage). When this is completed the lists of accessible services and model servers for the selected role and life cycle stage appear. The user then selects the services and the model server s/he is interested in. When the user confirms his/her selections, the connection module activates the respective service components and the model server. Immediately after the activation, the information about the selected role and life-cycle stage is passed to the activated program.

After the schematic design tool (in this case, the Auto Detail Wall/Door Design-er) has been activated, it queries the connected model server according to the role and life-cycle stage it has been designated to. For any objects in the model server, which have been annotated with corresponding actor and stakeholder roles, they will respond with messages according to the accessibility right they have been assigned in any previous session. When the tool receives the messages, it activates the local resource (graphical) components and displays the view of the object according to the respective roles and life

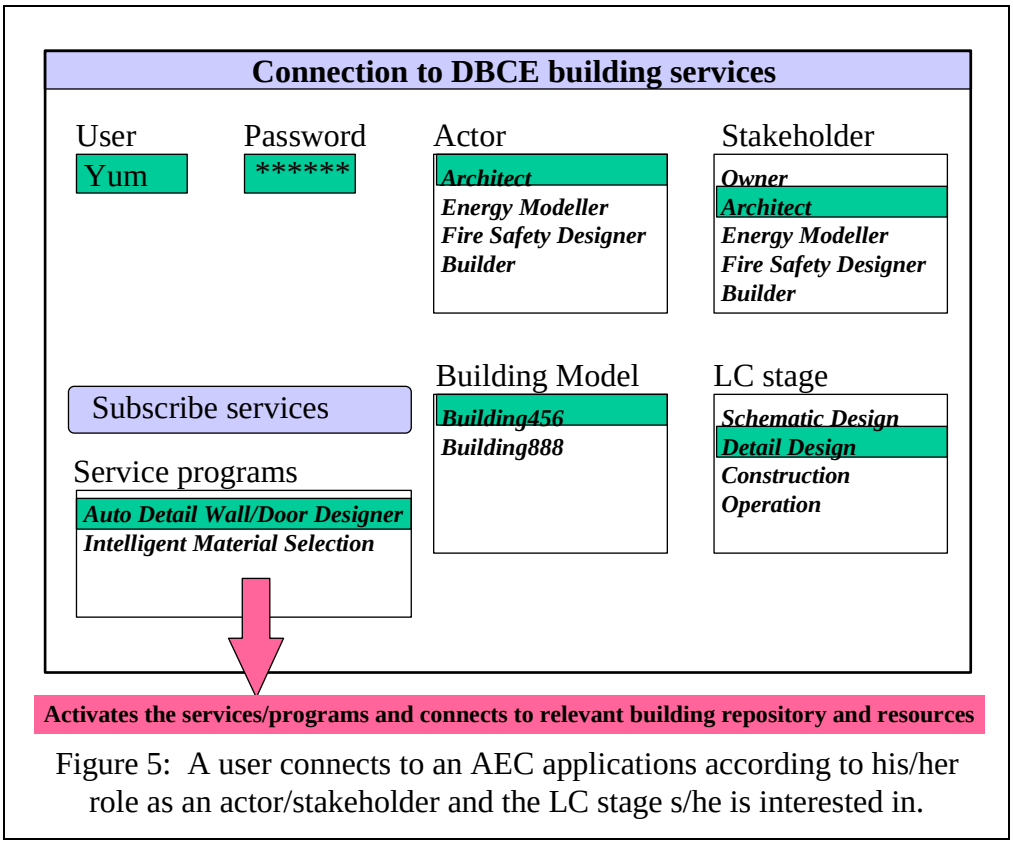

Figure 5: A user connects to an AEC applications according to his/her role as an actor/stakeholder and the LC stage s/he is interested in.

cycle stage.

Although the model server provides the functionality of creating, deleting and modifying the life cycle dependent objects within the server, when and where to create objects is controlled by to the services (applications).

Related Work

Here both the IAI (International Alliance for Interoperability, see IAI 1997) and the TINA (Telecommunication Information Network Architecture, see TINA-C 1997) initiatives are relevant.

In a presentation to the San Diego STEP Meeting in 1997, Thomas Liebich (Liebich 1997) of the German Speaking IAI Chapter presented the IFC (Industry Foundation Classes) Architecture as follows:

- First layer: Independent Resources
- Second layer: Core layer: Kernel (non AEC specific), Core Extensions (AEC specific, abstract part)
- Third layer: Interoperability layer: Conceptual objects that are used across domains but not quite in fit in the core layer.
- Fourth layer: Domain Model Layer: May provide a mapping mechanism for the "plugging in" of non IAI compliant domain models - an "adapter" layer.

The independent resource layer approximately corresponds to the resource parts in Figure 2. The only difference is that the graphical representation that is now in the core layer of the IFC model will be taken out and assigned to the resource part of the Connection Architecture. This is for performance reasons as we may just want to distribute the model part, but not resource part (transferring graphical pieces across network could be expensive). The core layer of the IFC architecture roughly corresponds to the model server part of the Connection Architecture. It is not clear at this point in time what common conceptual objects are in the interoperability layer. However, according to the brief documentation from the San Diego Meeting (Leibich 1997) the Interoperability layer of the IFC model seems to correspond well to the business model of the Connection Architecture (i.e. the relationships between user, role, life cycle stage, model server and services). Currently IFCs are primarily implemented on Microsoft's Component Object Model (COM) and hence they offer no distributed services in true distributed computing sense. As DCOM (Distributed COM) was introduced in to the market last year (1996), it will generate an interest within IAI to develop distributed solutions in the future.

TINA is an international alliance of expertise for the speedy delivery of interoperable solutions through global and regional industry collaborations within the telecommunication industry. The TINA project started in 1992 and will end in the end of 1997, involving major telecommunication companies around the world. TINA's business model defines the relationship between major business roles, viz., Consumer, Retailer, Broker, Connectivity Provider and Third Party Service Provider (TINA-C 1997). TINA's distributed services are implemented on the CORBA distributed Processing Environment. TINA's business model has little to do with the AEC industry. However, the TINA work provides an important reference point for this industry and demonstrated that it is *possible* to focus on the business role based on an open, interoperable distributed architecture for gradual uptake of the integration work.

University of Salford (Brown, *et al.* 1996; Rezgui 1996) developed a CORBA-based prototype of their COMMIT Information Management Model (CIMM). The work covers a section of data management concerns in the AEC industry: rights and responsibility, notification, version support and representation of intent. Like the Connection

Architecture, CIMM sits on top of the building model and Application Protocol. Similar to the IFC Architecture, COMMIT adopts a layered model architecture. While the Salford work is still at the feasibility stage, it raises some important questions that are relevant to the theme of this paper: How do we structure major AEC stakeholders' roles so that different roles has different responsibilities that are reflected in their business models? How do roles of different expertise in different life cycle stages are going to communicate with each other to serve their core business? Issues like these have not been explored neither in this paper nor in Salford's work. Perhaps TINA's approach of starting the integration work from business models can offer a viable and practical opportunity for the continuation of the integration work in the AEC industry.

The major difference between the Connection Architecture and the IFC architecture or CIMM architecture is that the former is a component-based architecture whereas the latter two are layered architecture. In analogy terms, the layered architecture favours the existing approach of software plugin technology akin to Netscape browser's; whereas the component-based architecture favours the next generation component technology adopted by today's Microsoft Internet Explorer. The Connection Architecture just treats services (applications), model servers, resources, and even the connection module itself as software components. By this token, the Connection Architecture seems more in line with IT deployment trends for the next generation.

Summary and Future Work

This paper explores an alternative to the conceptual industry framework approach which has been criticised for being slow in delivering industry results. The core of the proposed alternative is to establish an interoperable architecture and gather together key industry stakeholders to work on it. The interoperable architecture offers formalisms for the industry partners to define business roles and their relationships; from there on they can work collaboratively and incrementally based on existing models (business model, information model and computation models). The industry stakeholders will be attracted to this scheme because the interoperable architecture allows them to specify the roles and relationships that concern them.

In this paper, the business model and the information model of the Connection Architecture is only a first step toward the interoperable architecture. The business role presented in the paper deals only with the access rights of various business roles. It has not gone into details, like those in TINA, of specifying the actual AEC business role, e.g., the business role of a building designer as a consumer accessing the material resource which could be a service provider of its own right. However, this paper does present a way of abstracting out the business viewpoint of data access and formulating its underlying information view in an object-oriented approach. These views will be implemented in a distributed processing environment such as DCOM or CORBA for evaluation in the future.

The paper's presentation of the business models and information models of the Connection Architecture is generally compatible with the RM-ODP methodology. Due to the limited scope of the work, the power of RM-ODP is not yet fully utilised. Further effort is needed to ensure RM-ODP compatibility when moving existing R&D to a larger domain area. For this to happen, key stakeholders will be invited to join an industry alliance for the propagation of distributed computing to the AEC industry.

References

Bjork, B.-C. 1995, *Requirements and Information Structures for Building Product Data Models*, VTT Publications 245.

BRE 1997, "Building Construction Core Model", http://www.bre.co.uk/~itra/ceic.htm.

Brown, A., Rezgui, Y., Cooper, Yip, J., Brandon, P. 1996, "Promoting computer integrated construction through the use of distributed technology", *ITCON electronic Journal*, http://itcon.org.

IAI 1997, End Users Guide to Industry foundation Classes, http://www.interoperability.com/pubframe.htm.

Liebich, T. 1997, IFC presentation at the STEP San Diego Meeting, http://www.leeds.ac.uk/civil/research/cae/step/meetings/sandiego.htm.

Spiby, P. 1991, *EXPRESS Language Reference Manual*, ISO 10303-11.

TINA-C 1997, *TINA Business Model and Reference Points Version 4.0*, http://www.tinac.com.

Rezgui, Y., Brown, A., Cooper, G., Yip, J., Brandon, P., and Kirkham, J., 1996, "An information management framework for concurrent construction engineering", *Automation in Construction*, Vol. 5, No. 1, pp.343-355.

RM-ODP 1997, "RM-ODP Information Service", http://www.dstc.edu.au/AU/research_news/odp/ref_model/ref_model.html.

Yum, K.-K. and Salomonsson, G. 1997, "A generic connection architecture for the support of CPR in the AEC sector", *Proceedings of CPR-97 Conference, Gold Cost, Queensland, Australia*.