OBJECTS DEFINITION AND THE LIFE CYCLE OF OBJECTS

M. Alshawi[1]

Abstract: In order to produce a complete and efficient objects definition for an integrated environment, it is important that objects are considered within a well defined environment. Objects are subjected to a number of phases during their life cycle depending on the environment within which they are serving. This paper briefly explains a proposed structure for an integrated environment within which a new concept for object life cycle is introduced. The new concept is illustrated in an elements specifications application.

Keywords: Integrated environments, product model, object definition, object life cycle.

INTRODUCTION

Integrated Design and Construction has been the main issue in Computer Integrated Construction (CIC). It has been widely recognised that in order to achieve the best performance, such integrated systems need to share data via a central core, i.e. a product model. Great efforts therefore have been devoted towards the development of a flexible and comprehensive data structure, for a central data base, with the aim of serving all possible downstream applications. However, central cores can't be populated with project specific information without clearly defined objects (elements) which are anticipated to be transferred dynamically to the central core. The issue of generating information for building elements and defining their type and format which are required by downstream applications have not yet been fully addressed. The STEP standards, so far, have materialised in the form of data modelling and data representation and not into protocols for defining and transferring of geometric and non-geometric data. Recently, committees have been established for defining application protocols for construction applications along with a building construction core model.

This paper is discusses this issue and highlights problems that might be associated with the implementation of the theoretical development of construction data models. It then briefly explains a proposed structure for an integrated environment within which data can be exchanged over their life cycle. It also proposes four phases for objects life cycle and then illustrates this concept with an elements specification application example.

[1]Dr. M. Alshawi, Senior Lecturer, Department of Surveying, University of Salford, Salford, M5 4WT, UK.
Email: m.alshawi@surveying.salford.ac.uk

STEP AND APPLICATION PROTOCOLS (APS)

The STEP initiative is built upon the key concepts of Product Models (PMs) and Application Protocols (APs). A Product Model is a formal information model of how the totality of (engineering) information concerning a particular type of artifact can be coherently represented to facilitate information sharing (Watson & Boyle, 1993). An AP is a standard that defines the context and scope for the use of product data and specifies the interpretation of the standardised integrated resources in that context to satisfy an industrial need (Palmer et al, 1991). In May 1994 a Building Construction Core Model has been proposed with the aim of provide the overall definition of the entities and their interrelationships in such a way as to enable the different participants and different applications software to use the definitions as a common basis for the exchange of project data and knowledge.

There are two APs have been developed; Building Structural Frame: Steelwork and Building Services Heating, Ventilation and Air Condition. The former AP is concerned with the steel frame buildings and similar structures while the latter is concerned with the design and installation phases of HVAC over the building life-cycle.

PROBLEMS WITH THE IMPLEMENTATION OF DATA MODELS

Application protocols, if developed effectively, can play an important role in any future development of integrated environments. However, the development of such protocols in isolation of any implementation attempt will lead to the development of theoretical models which could prove difficult to implement as implementation problems can be of a completely different nature. Moreover, the comprehensibility of such large data models and their proposed interaction with each other, if implemented in an integrated environment, can create complex development conditions which make the management of the development process almost impossible. This is mainly occur due to the involvement of a large number of hardware/software/construction professionals in the development and testing processes where the overall responsibility is shared between the various professionals.

Moreover, the absence of a strategic view of how an integrated environment can be structured and run make different developers interpreting and implementing the proposed application protocols differently depending on their understanding of STEP principles or the context for which the protocols have been developed. The case is not so bad when dealing with a single application protocol e.g. the design stage of steel framed buildings. However, the situation worsen when developers attempt to integrate or exchange data between various data models within one or more application protocols. This has been demonstrated in the European CIMsteel project where all the proposed applications over the life cycle of a steel building

are planned to be integrated through one central core i.e. an integrated database. Among other problems, it has not been able to demonstrate the integration through one central core. The demonstration has been focused on file transfer from one application to another.

Data exchange between the various data models depends highly of the structure of these models and the context of the environment that they are prepared for. Objects need to know the type of information and knowledge that they should retain within their frame and those that they should refer to e.g. common data. This can't be done without a clear understanding of the phases that objects go through during their life cycle. It is very likely that objects inherit different type of information at different phases of their life cycle depending on the structure and context of their environment. It is therefore important to develop an agreed structure for an integrated environment in parallel to the development of the data models and before attempting to establish "standards" for objects' data. This can significantly improve the implementation process and can yield an important feedback on the developed application protocols.

THE INTEGRATED CONSTRUCTION ENVIRONMENT (ICE)

A framework has been proposed for the Integrated Construction Environment (ICE) with an aims of integrating various construction applications under one environment. Project's information is controlled and manipulated by one central core from which all integrated applications can access their relevant information (Faraj & Alshawi, 1995). This paper briefly explains the conceptual structure of the proposed ICE.

The proposed framework, as shown in Figure 1, consists of three main parts i.e. the central core, external applications, and project specific data. The central core represents a repository for the ICE whereby all information related to a project type (context), e.g. concrete office buildings, is maintained. It consists of a number of data models which describes the life cycle of the project type under consideration. These models have been classified as building data model and other data models. The building data model mainly describes the physical features and behaviour of the project type and should support different levels of data abstraction. The extent and structure of the building data model depend on the scope, and the main objectives of the ICE. Different ICEs may have different structures of building data models, level of abstractions, different coverage of the project's life cycle, etc. Other data models represent data required by the various applications within the ICE such as Construction Planning Data Model, Site Planning Data Model, Specification Data Model, Estimating Data Model, etc. Each data model should be developed to achieve a well defined set of objectives and to fulfil the need of a particular application. For example, a construction planning application requires a data model to support the information required by

the package e.g. generic construction activities, resources available, construction methods, etc.

The second part of the ICE represents the down stream construction applications such as design , construction planning, estimating, site planning, virtual reality modelling, etc. Such applications could either be external, i.e. existing stand alone applications, or internal i.e. developed within the ICE software tools. In either case each application has 1) a built-in user interface where users can manipulate the output information, and 2) a specially developed two way communication channel to transfer information between the application and the central core at real time, Figure 1. The third part of the ICE is the information required to trigger off the applications i.e. information related to a specific project where solutions are required.

Such an environment is normally triggered off by feeding in project specific information through a design package e.g. a CAD system where large amount of project's information can be extracted (Dym and Levitt, 1991). As the design progresses, design information is dynamically transferred to the central core, at a high level format i.e. design elements (Alshawi, 1994). Once the central core is populated with the project specific information, users can run any other application at any stage of the design. For example, the cost of the so far developed design can be dynamically determined by running the estimating application. The central core, in this instance, transfers the design elements along with their specifications and quantities to the estimating package to produce either the total cost or cost break down of the current design product. Users should be able to alter the generated cost figures, add cost constraints on certain design elements, etc. using the estimating package interface. This altered information is then transferred to the central core where actions are triggered-off if the altered information is non-feasible or does not comply with regulations, standards, in-house database, etc.

Users can switch between various applications at any stage at any time. Applications respond to requests from the central core and their behaviours are limited to the supplied amount of information.

Objects in such an integrated environment are populated with different type of information and knowledge at different stages of their life cycle. In this proposed ICE, an objects can go through four phases during its life cycle. These phases are explained in detail in Alshawi, 1995. This paper however discusses the type of data supplemented to objects at their creation phase i.e. in a CAD application.

OBJECT DEFINITION

In order for such objects to respond correctly and efficiently to the needs of different users at any stage of the project's life cycle, each object has to contain the necessary information and knowledge to serve all possible down stream

construction applications. Due to the complexity and duplication of such information and knowledge, it is not practically feasible to inflate each object with all the required information and knowledge. This will create data management problems, i.e. data control and maintenance, and data modelling difficulties.

Objects must have an optimised definition in order to enable them to behave adequately, efficiently, and intelligently during their life cycle. This definition is highly dependent on the aims and the structure/context of the environment within which the objects are serving. For example, an object can be created in a database application to satisfy a particular need. This object inherits its features from those provided by that application. The information and knowledge retain by this object will be different from those created in a graphical application where the information inherited service only the graphical manipulation of the object.

In an integrated environment, objects are supposed to react to various requests in the most efficient way. Objects can expose their data to other applications or they can point to common data. This implies that object's information can be saved within the frame of the objects, while others can be shared with other objects. The following section introduces four phases for objects' life cycle and outline the information which objects inherit during their life cycle.

THE LIFE CYCLE OF AN OBJECT

The life cycle of an object can be described in four phases; create and amend, supplement object with data, use object, and decommission object. Figure 2 is an IDEF0 diagram which represents these phases in detail. The first phase refers to the creation of an object in an application. An object can be created in any construction application such as CAD packages, databases, planning packages, etc. and is constrained by the user/firm experience and building codes and standards. These constraints plays an important role at this phase of the life cycle of an object as they control the type and amount of information and knowledge which can be attached to the object. However, once an object is created in an application, basic information which is relevant to that application, is attached to the object's frame.

The second phase of the life cycle of an object is to supplement object with data. In an integrated environment such as that proposed earlier, this phase is carried out in two parts. The created object firstly attached to its related data model in the ICE and then to its specialised class. This process is constrained by the limitation of the implementation package(s), i.e. if an object is transferred between stand alone applications then it will be constrained by the utilities provided by the selected implementation packages. Thus, this part can have a significant impact on when and how data is transferred and in what format. Secondly, the object is populated with a another set of information, after it has instantiated in a data model, utilising the object oriented features of the ICE i.e. inheritance and

polymorphism. When objects are transferred to the central core of the ICE, they are normally attached to a Building Data Model (BDM) which is an elemental representation of the project's type under consideration. The type and amount of information and knowledge inherited by the transferred object depends on the structure and knowledge of the data model. Normally this type of information is of a descriptive nature where an object inherits its identity, classification, and general behaviour. If an application within the ICE requires a graphical representation of objects, such as Virtual Reality, then an object must generates its standard graphics files after it has been created and must retain a reference to that file.

Once an object is supplemented with data after its creation, its existence will be featured by other applications within the ICE. The object enters the third phase of it life cycle, i.e. use objects phase. At this stage, objects should have all the necessary information and knowledge to serve all down stream applications within the ICE. Objects' response to requests from applications is constrained by the domain knowledge, i.e. the context of the ICE, and the structure and knowledge of the applications' data models within the central core of the ICE. Thus, objects can provide descriptive information from its own frame including referencing to its graphics files, generate relevant information within other data models, or share common data with other objects. Any instruction issued by the object to other data models i.e. either to generate or share data, is documented within the frame of the object so it can refer to it in feature, if necessary. The object becomes more mature as its get more involved with the applications.

During the third phase, the situation of an object can be altered on a request from an application by altering one or more of its definition features. In this case, the object automatically adjust itself by going back to either phase one, if the changes effect its basic definition, or to phase two, if the changes effect the supplemented data. In either case, the object will behave according to its new situation. Any changes can be documented within the object's frame which will enable the object to return to its old situation if need be.

The last phase of an object life cycle is the decommission phase. Users can terminate the life of an object by deleting it from the BDM. This will result in an end to the existence of the object and to all its relevant information including those which have been generated by the object. The out put of this phase will be an object free model.

IMPLEMENTATION OF THE ICE CONCEPT

The above concept was implemented by the AIC research group (Automation and Integration in Construction) at the University of Salford to establish an effective objects definition within the proposed ICE. The ICE concept has been implemented in an integrated environment named as "SPACE" (Simultaneous Prototyping for An integrated Construction Environment) which runs on a PC

platform. At its current stage of development, SPACE integrates four external and two internal applications with central core. The external applications are; design, construction planning, virtual reality modelling, and site layout planning, while the internal applications are elemental bill of quantity and elements' specifications. All applications have been implemented using commercial software i.e. Auto CAD for the design and site layout planning, Super Project Expert for the construction planning application, and World Tool Kit for the virtual reality application. KAPPA has been used as the central core environment. The elemental bill of quantity and elements' specifications application have used KAPPA as their implementation media. The following section explains how the concept of object definition has been implemented in SPACE.

Implementation Of A Wall Object

To illustrate the object definition concept, an object e.g. a wall will be defined through its life cycle using one down stream application i.e. elements' specifications. This application generates the specification of design elements during the design stage and allows users to alter them at later stages.

In SPACE, project specific information are normally entered through the CAD package and then dynamically transferred to the central core. When a wall is created in AutoCad, a representative object is automatically created (Ewen and Alshawi, 1993). It encapsulates all the necessary information that defines the basic features which are required by the down stream applications within SPACE (Alshawi and Putra, 1995). For a cavity wall these are; rotation angle, associated elements (objects that are built in the wall i.e. windows and doors), start co-ordinates, level, X Y Z Dimensions, cavity thickness, an inner leaf thickness, and an outer leaf thickness. Each of these data has a role to play in serving the rest of the applications. At this stage, the basic definition of the wall is established and the object is transferred to the central core where it enters its second phase of the life cycle.

When the wall object is received at the central core, an instance is created for the wall and attached to the building data model under the class cavity walls. The basic definition features are stored within the object's frame along with those inherited from the building data model. The latter include an object identity i.e. member of the cavity walls class, its construction activities, associates type (i.e. type of relationships that the cavity wall has with other objects such as supported by, embedded in, etc.), CISfB code, CPI code, and SMM7 code. The SfB code serves to identify the functional attributes of the object, the CPI code represents the work sections according to the CPI CAWS proposals, SMM7 code describes the object in terms of its work content. These codes assist other applications to classify building elements in various format. For example, a wall can be part of brickwork elements, super structure, envelop, etc. At this stage the object will establish an

independent status which is recognised by the ICE and in turn by all other applications.

At this stage the wall needs to generate/refer to its graphical and specification data which are considered as a vital pre-request to all other applications. The wall, after it has been initiated in the central core, requests the CAD package to generate a standard graphical file under its identity name. At the same time, the wall requests the specifications application, which is an application data model linked to standard elements and standard materials database, to refer it to its specification. The specifications application checks whether an identical wall exists. If so, the application sends a reference number to the wall object which in turn stores it in the its own frame. Otherwise, the specification application:

a. retrieves the wall's basic features
b. matches them with those available in the data base,
c. retrieves those specifications which best match the wall's features,
d. creates a specification object in its data model
e. sends a reference to the wall object.

When a wall is deleted, its instance is deleted from the building data model as well as its related specification object, if it is the only wall type that refers to that specification object.

CONCLUSIONS

It has been widely recognised that in order to achieve the best performance, such integrated systems need to share data via a central core, i.e. a product model. Great efforts therefore have been devoted towards the development of a flexible and comprehensive data structure with the aim of serving all possible downstream applications. However, central cores can't be populated with project specific information without clearly defined objects (elements) which are anticipated to be transferred dynamically to the central core. In this context, the STEP APs are considered to be an important step towards the identification of application specific data models. However, the development of such protocols in isolation of any implementation attempt will lead to the development of theoretical models which could prove difficult to implement as implementation problems can be of a completely different nature. Thus, it is important that an agreed structure for an integrated environment be establish in parallel to the development of the data models. This can significantly improve the implementation process and can yield an important feedback on the developed application protocols.

This paper has briefly explained a proposed structure for an integrated environment within which a new concept for object life cycle is introduced. Four phases have been highlighted for objects life cycle. These are; create and amend, supplement object with data, use object, and decommission object. Objects are

inflated with data at the various phases of their life cycle. After initiation, objects can either take a new status, i.e. when edited, or become mature objects. At the end of their life they can either be deleted form the central core or become obsolete.

The object's life cycle concept along with the proposed structure for an Integrated Construction environment have been successfully implemented in the "SPACE" integrated environment at the University of Salford. The two concepts have proven to be very effective where data are stored in the most efficient way.

REFERENCES

Alshawi, M. and Che Wan Putra, F., 1995, Object Definition In An Integrated Environment, *Civil-Comp 95*, London, August.

Alshawi, M., 1994, A Run Time Exchange Of Elemental Information Between CAD And Object Models: A Standard Interface, *The International Journal of Construction Information Technology*, 2 (2), pp 37-52.

Dym, C. and Levitt, R., 1991, *Knowledge-based Systems in Engineering*, McGraw-Hill Inc., pp 182-183.

Ewen I. and Alshawi M., An Object Oriented Approach For Architectural CAD Drawings, the Application of Artificial Intelligence To Civil and Structural Engineering, *Civil-Comp*, Edinburgh, August 1993, pp 167-174.

Faraj, I. and Alshawi, M., 1995, A Generic Approach For An Integrated Construction Environment, Sent for publication, ASCE, *Computer in Civil Engineering*, Special issue on product and process modelling.

Palmer, M., Gilber, M., and Anderson, J., 1991, Guidelines For The Development And Approval Of STEP Application Protocols, May.

Watson, A., and Boyle, A., 1993, Product Models and Application Protocols, *Information Technoogy For Civil And Strutural Engineering*, B.H.V. Topping (ed), Civil-Comp, Edinburgh, pp 121-129.

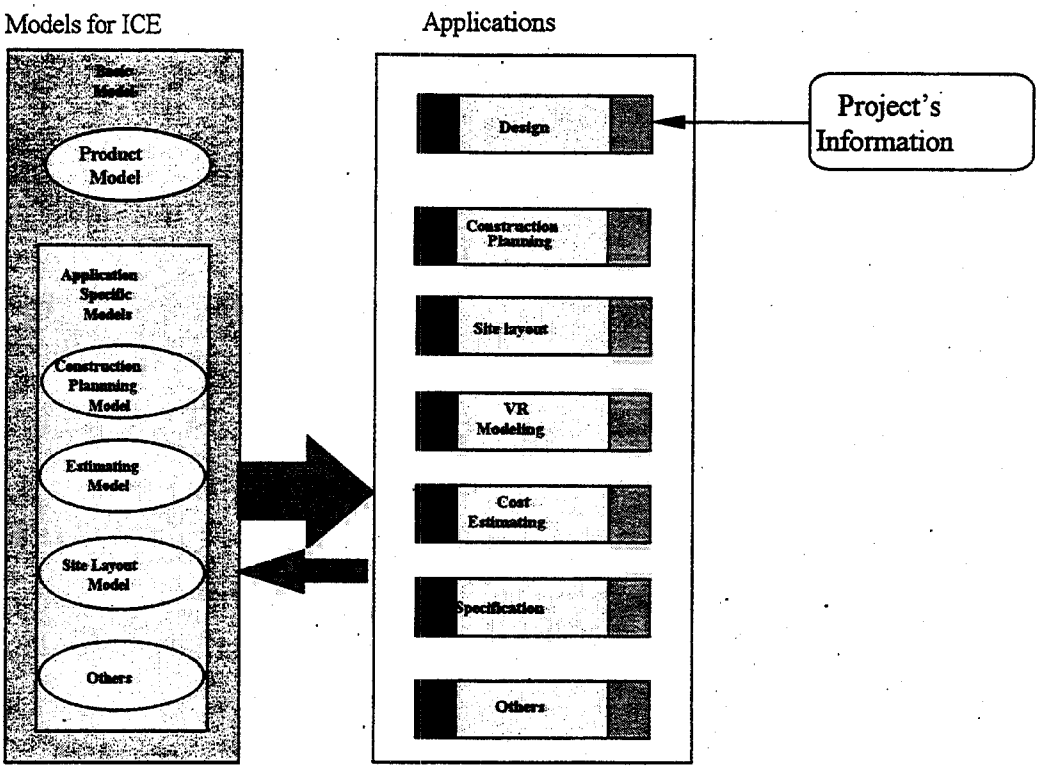

Figure 1: Conceptual representation of the ICE

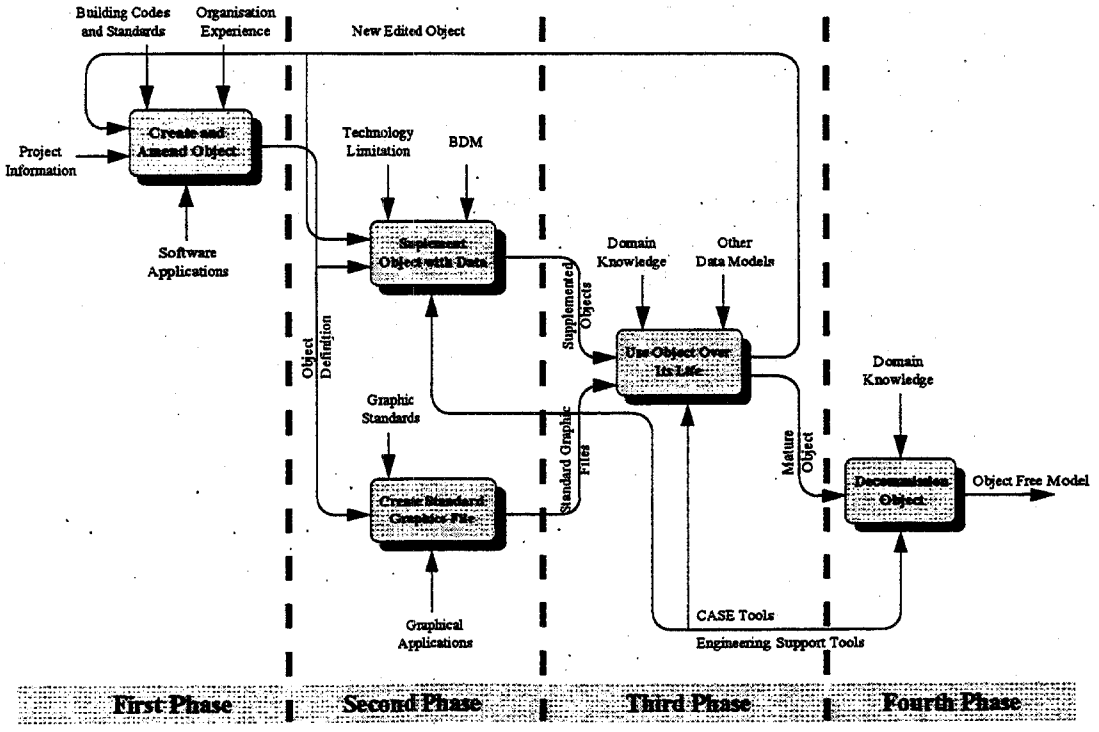

Figure 2: IDEF0 for Object's Life Cycle