TOWARDS OBJECT MODELS FOR INTEGRATED INTELLIGENT PROJECT MANAGEMENT

Tah, J.H.M.[1]

ABSTRACT: Current project management systems are too abstract and do not have a means of representing concrete knowledge about a problem domain. Previous research in this area has failed to address larger problem domains to a level of success significant enough to make an impact in the construction industry. Object-orientation appears to provide a powerful means of encapsulating knowledge in intelligent object classes and together with the product modelling approach promise to provide a solution for larger problem domians. This paper presents selected object models that have evolved from research work aimed at investigating new approaches to the provision of knowledge-based decision support within large integrated project management systems. The work utilises the object modelling approach to software engineering and is precipitating in what may potentially progress from foundation classes to software components for project management. The continual re-use and development of such classes and components will evolve standard application information requirements that can contribute towards ISO-STEPs efforts in developing application protocols and standards for data exchange and interoperability.

KEYWORDS: object models, object-orientation, integration, intelligent, project management, KBS.

BACKGROUND

Although there is a proliferation of software purporting to provide comprehensive project management facilities, they have failed to meet the needs of project managers. These systems are primarily founded on principles and methodologies derived from operational research developed in the 1950's. These systems by their very nature have been designed to be generic representing data at an abstract level. Consequently, there is little or no opportunity for them to retain concrete knowledge about individual problem domains and as such provide little decision support. They are designed to monitor the presence of a performance problem. They do not have the ability to diagnose the root cause of the problem, nor do they suggest necessary corrective action. It is the project manager's responsibility to diagnose the cause of the problem and to implement an appropriate response strategy. Furthermore, these systems require a considerable amount of effort in data input from the project manager which is both time consuming and to some extent error prone. This is due to the fact that they have been developed as stand alone systems with no ability to share data with upstream functions such as design. Consequently, this does not assist both designers and the project managers who wish to examine the consequences of alternative decisions and possible response strategies so as to select the best solution to a problem.

The purpose of this research is to utilise emerging technology from Artificial Intelligence research in an innovative approach to develop advanced knowledge-based decision support

[1] Senior Lecturer, Project Systems Engineering Research Group, School of Construction, South Bank University, Wandsworth Road, London, SW8 2JZ. Email: tahjh@vax.sbu.ac.uk

techniques for the deployment of future integrated project management systems. Previous research in the area of decision support for project management utilising knowledgebased systems techniques have failed to make an impact in the construction industry. This can be attributed to the use of expert systems shells which are too restrictive and are suitable for handling problems in highly specialised domains. Design and construction management problems are characterised by a multiplicity of products and processes which include tasks or work packages such as earthworks, temporary works, concreting, steelworks, drainage, tunnelling, piling, and services. It is, therefore, difficult if not impossible to develop a convincing all embracing knowledge-based system for say construction planning or estimating using an expert system shell. The basis of this research is to acknowledge the existence of several different work packages each with its own knowledge requirements, and investigate the application of more advanced techniques currently emanating from the Product Modelling, Object-Oriented Software Engineering, and Artificial Intelligence research communities. Our intention is to investigate the use of object-oriented knowledge representation and reasoning techniques together with the use of multi-agent and blackboard architectures to handle the disparate nature of construction problems. Furthermore, construction management problems are characterised by the presence of an abundance of quantitative or objective data. Solution strategies to these problems involve the use of these data together with information of a qualitative nature, often derived from experience. Therefore, new techniques should be capable of handling both quantitative and qualitative data.

The advent of Object-Oriented Programming (OOP) provides new opportunities for dealing with the limitations of rule-based paradigms typically found in former expert system shells in the handling of quantitative data. Object-oriented programming languages provide powerful information modelling capabilities. Information is modelled in the form of objects which represent and capture the structure and behaviour of real world entities, making them ideal for simulation purposes. An object is a data structure that combines both data and procedures. The encapsulation of procedures along with data gives it the ability to reason about its domain, hence allowing intelligent operations to be performed. OOP allows the representation and application, within an object, of both rules and procedures in ways that were impossible, or at least extremely difficult, in former rule-based formalisms.

Decisions in project management are often made using incomplete or uncertain information, particularly those pertaining to risk management. In such situations, project managers use their subjective judgements to assess project risks likelihood and severity. The parameters involved are described qualitatively in linguistic terms using imprecise non-numeric quantification. The linguistic terms or variables can be translated into mathematical measures using fuzzy sets. The object-oriented problem solving strategies developed in this research will incorporate approximate reasoning techniques based on fuzzy sets theory. The intention is to demonstrate the potential of fuzzy set theory in highly subjective decision making processes within construction management systems.

OBJECT-ORIENTED SOFTWARE ENGINEERING

Software development has progressed from machine code and assembly language, through to higher level languages. In addition there has been a progression from unstructured code, to procedures and functions, to libraries, and to shared libraries. Object-oriented software has been a logical next step in both of these trends. Similarly, the object-oriented software engineering methodology has been the result of the evolution of structured software

engineering methodologies. The trend has been driven by the need to evolve high-level data and procedural abstractions that facilitate the analysis, design, development, and maintenance of software. This has precipitated in the concept of a class in the object-oriented programming methodology as a behaviour-centred rather than a data-centred approach. These developments allow us to begin to attempt to handle the inherent complexities of a fragmented construction industry, with multiple disciplines, complex disparate processes and sub-processes.

Object Oriented Design and Re-usability

Objects represent dynamic entities in computer memory that define data states and serve to group data that pertain to one real world entity. An object encapsulates both state and behaviour by having a set of procedures (or methods) that specify operations. Sets of similar objects are grouped together under classes to simplify association of knowledge within objects by keeping the implementation details private within each class and allowing interactions between objects of different classes to be easily controlled and manipulated. The information about how an object behaves is hidden from the behaviours of other objects, only their interactions and relationships are visible, through the behavioural interface of the class. The close coupling of the data and operations is one of the most important features of object-oriented techniques.

The notion of classes is being used to develop class libraries or a set of common classes that can be re-used in software development. Classes or objects are inherently units of reusable code and systems built from them are extensible, using inheritance to add new or exceptional features to the system. Designing object-oriented systems is hard, and designing reusable object-oriented software is even harder. The process involves finding pertinent objects, factoring them into classes at the right granularity, defining class interfaces and inheritance hierarchies, and establishing the key relationships between them. The design should be specific to the problem at hand but be general enough to address future problems and requirements. It is desirable to avoid redesign, or at the very least minimise it.

Object-Oriented Analysis and Design Methodology

Modelling is used in diverse fields as a means of understanding a concept or an artefact and communicating its essential properties to interested parties before actually building it. The resulting model is a simplified version of a real-world concept. It is an abstraction of reality, representing important properties, thus allowing humans to deal with the complexity of a real problem. Modelling is a common practice in software engineering. Defining models in computers involve: representing a model of the real- world using a methodology, usually depicted graphically; transforming the model into source code; and allowing the user to use, amend, and add information to the model.

The burgeoning interest in object technology has led to a massive proliferation of object-oriented methodologies. These methods vary in complexity, scope and their degree of conformance to the object-oriented perspective. We use the industry standard methodology of Rumbaugh (1991).

The Rumbaugh's Object Modelling Technique (OMT) is based on the use of an object-oriented notation to describe classes and relationships throughout the software development lifecycle. The lifecycle is divided into seven phases: conceptualisation, analysis, system design, object design, implementation, testing/debugging, and deployment. During these

phases, three models are built and refined to provide three different but interrelated views of the system. The Object Model is augmented with a Dynamic Model and a Functional Model to describe all aspects of a system.

The Object Model
The Object Model describes the static structure of the system. It shows the structure of objects in the system and provides the foundation for the Dynamic and Functional Models. The Object Model consists of the following: class diagrams; object diagrams; sub-system Diagrams; and data dictionary.

The Dynamic Model
The Dynamic Model describes the control, or time-dependent, behaviour of the system. It shows the events and the allowable event sequences for each object. The Dynamic Model can consist of graphical representations shown by the following: state diagrams; use case diagrams; scenario diagrams; event flow diagrams; and object interaction diagrams.

The Functional Model
This describes the functional, or transformational, behaviour of the system. It models the aspects of the system with changeable values. The Functional Model can consist of graphical representations shown by the following; data flow diagrams; and object interaction diagrams.

The Object Model is most suitable for depicting the notion of re-use and as such will be used in the rest of the paper to present reusable classes and software components. The reader is refereed to Rumbaugh et. al. (1991) for the details of the notation used in the diagrams presented in this paper.

INTEGRATED INTELLIGENT PROJECT MANAGEMENT OBJECTS MODELS

Scope
Project management is a very large problem which cannot be fully tackled within the time scale of this research. Therefore, it is necessary to choose more specific representative problem domains that can be more manageable, and can demonstrate the capabilities of object-oriented knowledge-based systems (KBS) techniques for decision support more convincingly. Two problem areas of project management were selected. These are the formulation of a construction programme and the corresponding risk management through qualitative risk assessment. The programme formulation process consists of the following tasks: establishing general information concerning project details; developing a suitable Work Breakdown Structure (WBS) to represent tasks; establishing the quantities of work involved for each task on the WBS; selecting a method of concreting; selecting resources to suit the method of construction; and determining the duration of each task. The risk management process involves performance monitoring, risk identification, risk assessment, risk analysis, and risk handling. We focus on the in-situ concreting class of work covering both reinforced concrete and mass concrete structures associated with multi-storey building construction. To further delimit the boundaries of the knowledge sources the work covers only the super-structure element and the associated components.

Modelling Framework

Integrated project management involves several disciplines, each with its own view of data. Therefore, it is important to establish a framework that allows the categorisation of data into different levels of abstraction. Iosifidis et. al. (1995) propose a "Three Level Model" comprising of a discipline level, a view-point level, and a data level. The selection of a discipline and examination of its view of data, allows data to be gathered and organised for a particular area. The integrated project management problem presented in this paper involves two disciplines: the structural design; and the project management discipline.

This paper concentrates on the data level which involves the identification of objects, attributes, and their relationships. The first task in this process is to identify objects and to group them into classes. Once a sufficient number of classes has been identified, a decision has to be made as to whether they are part of an aggregation or inheritance hierarchy. An aggregation hierarchy allows a large composite object to be broken down into its component parts. It can form multiple levels where each level increases in detail, for example a building structure can be broken down into the sub-structure and the super-structure. If an object is a sub-type (i.e. a specialisation) of another it becomes part of an inheritance hierarchy, for example, a column and a beam are specilisations of a structural component. Once aggregation and inheritance hierarchies have been established, the relationships between them are defined to reflect dependencies in the read world. Other associations may be defined between classes. The relationships and associations model most of the semantics within a product model. Attributes and methods are then added to individual classes. In the ensuing section we present selected key object models that have been developed to reflect the data, behaviours, knowledge, and their inter-relationships. These include: a reinforced concrete structure object model; a task object model; and a risk management object model.

The concrete structure object model

The object model in Figure 1 represents the object classes within a reinforced concrete building structure. The building structural system is composed of the sub-structure and the super-structure, which are both composed of structural members. The structural member class is a base class for concrete structural members which may be a column, beam, wall, slab, etc. A structural member may or may not contain reinforcement. The structural member reinforcement class represents reinforcement within a structural member and it uses a reinforcement material database through the class reinforcement, the details of which are not shown in the diagram. The diagram shows that a structural member may be constructed by a task. The task class provides the link to the task system through the "set_task" method which sends a message to the task class to set appropriate work packets or units of work needed to construct the member.

The task object model

The task system object model in Figure 2 represents the information required in the planning and controlling of the production process of the space, separation, and structural systems. The task system is centred around the construction project plan which consists of one or many construction planning activities or tasks. An activity can be assigned to one or many resources. A resource can be a plant, material, labour, sub-contractor, or a resource group consisting of more than one resources, as shown in Figure 3. The resource object model is used to develop a generic object-oriented database of construction resources including labour, plant, materials, sub-contractors, and resource group build-ups. The task class is sufficient for general planning systems as is typical of current packaged project management software. However, for a knowledge-based system the task class is too abstract. Thus, the task class is specialised into more concrete classes. A task can be a

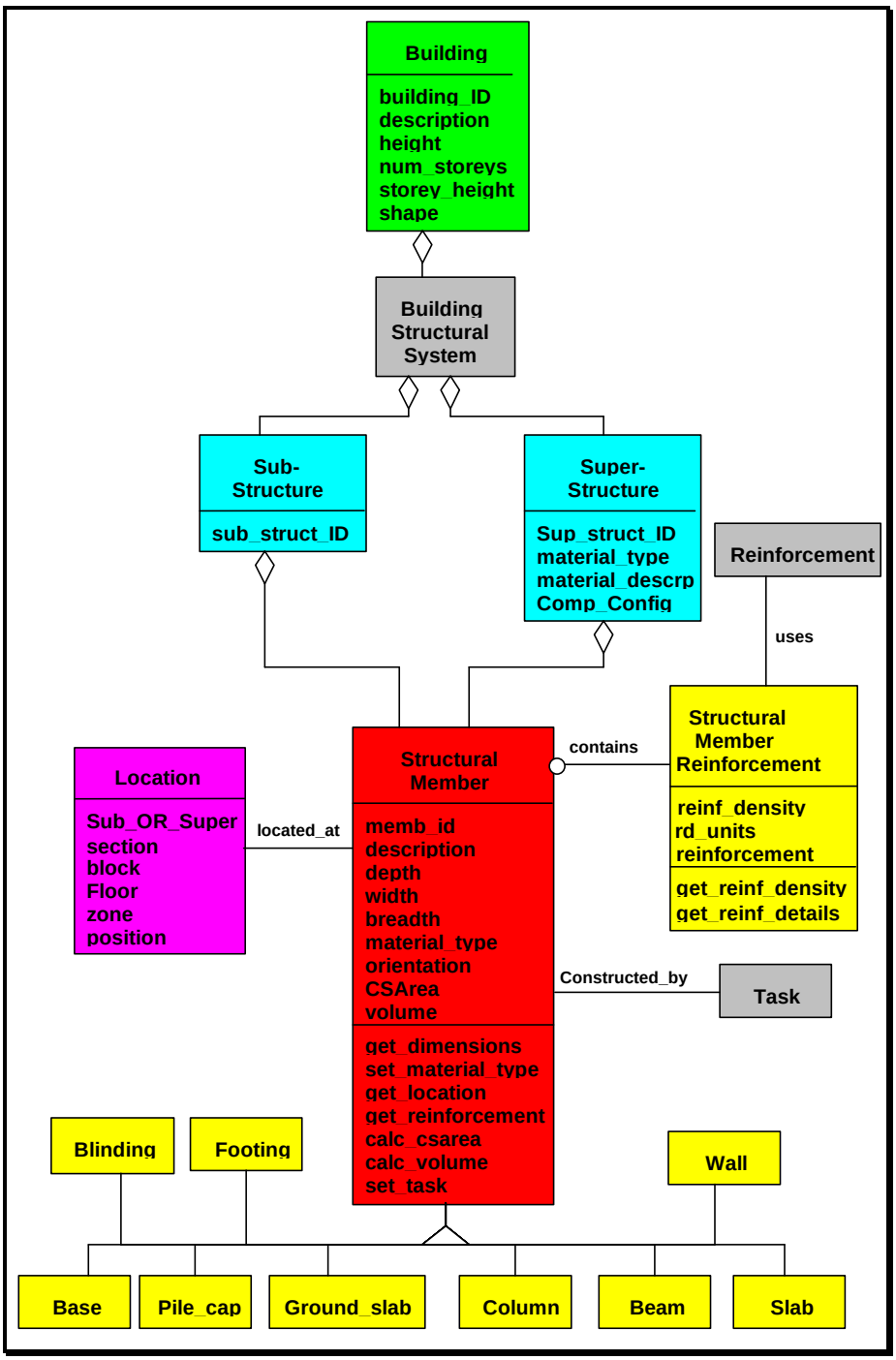

FIG. 1: The building structure object model

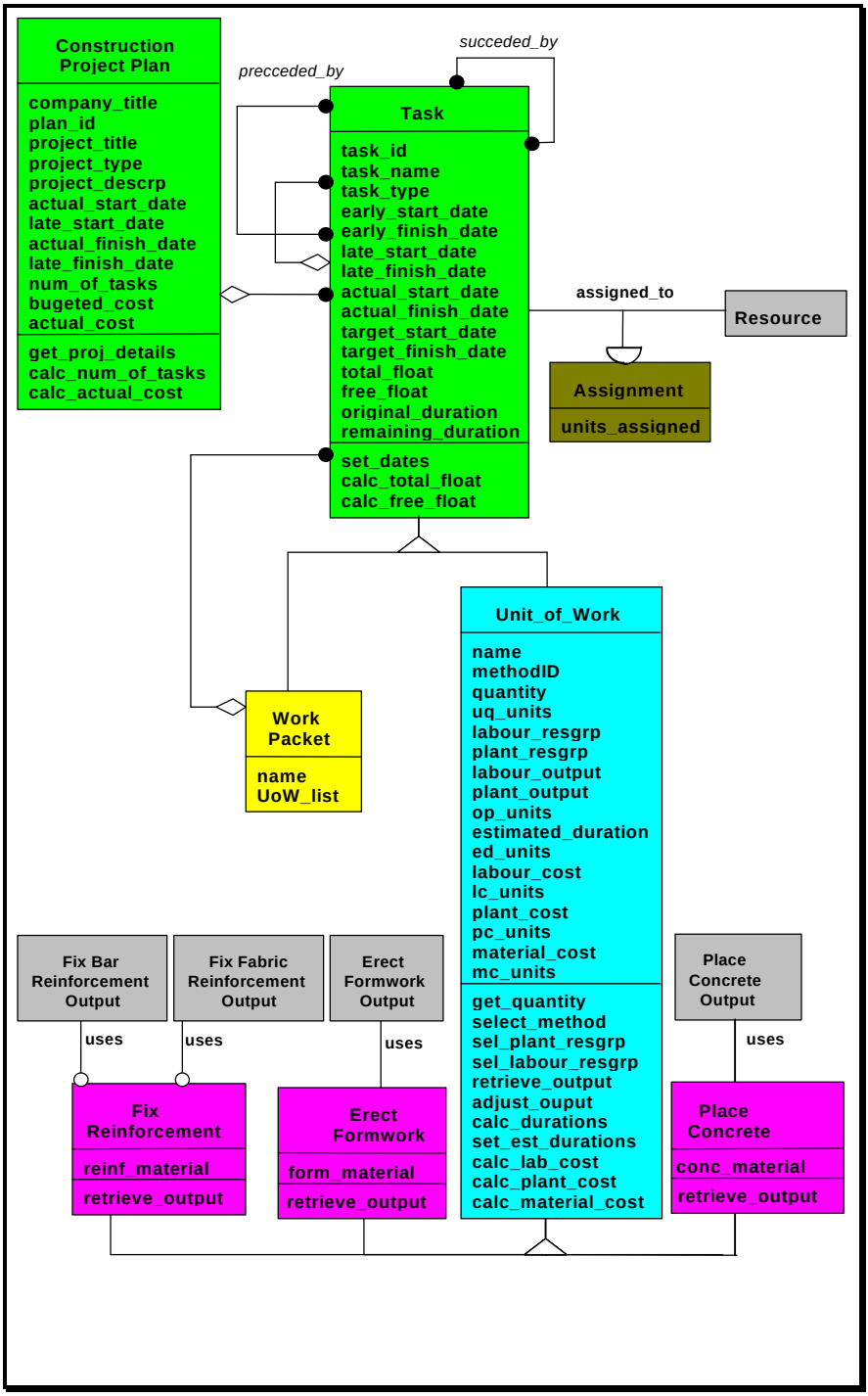

FIG. 2: Task system object model

"work packet" or a "unit of work" (UoW). The "unit_of_work" class acts as a base class to more concrete units of work which may be "fix reinforcement", "erect formwork", "place_concrete", "strike_formwork", etc. A work package consists of many units of work. The unit of work is very significant for the purposes of knowledge representation and intelligent decision support. Through its methods, several knowledge sources can be triggered to provide information and advice on the selection of for example methods of concreting, the selection of appropriate resources, and the retrieval and adjustment of output rates for task duration estimation.

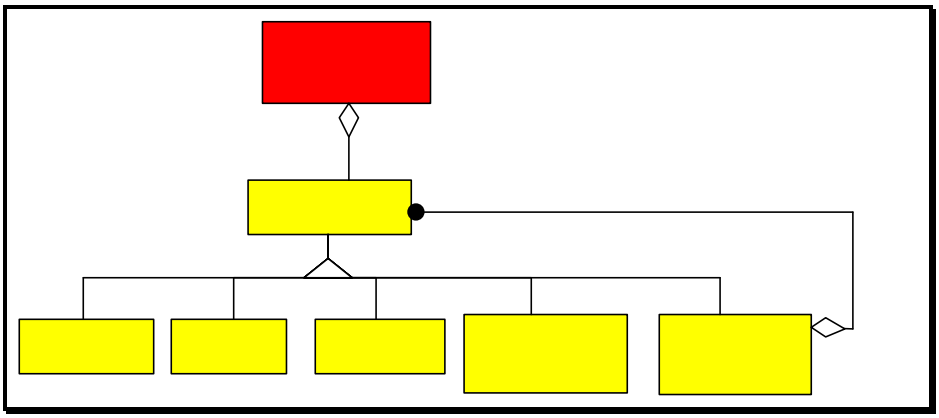

FIG. 3: The resource system object model

The risk management object model

The risk management object model in Figure 4 represents the information requirements for a risk monitoring and control sub-system. The classes within this model a centred around the risk class and reflect the processes involved in risk management. The risk catalogue class represents all possible risks that may be encountered in any project. The action catalogue class represents actions that can be taken in response to risks. The risk action catalogue class is a link representing risks and their associated actions. These three classes are used to maintain a generic database that can be utilised for individual projects. The risk class represents risks identified for a particular project and it uses the risk catalogue class for obtaining the details of risks. Risks may affect the whole project in which case they are considered global risks or they may affect individual tasks in which case they are described as local. The performance monitor class represents measured cost, schedule, quality, and safety performance deviations for individual tasks. The risk analysis class allows inter-dependencies between risks to be represented, qualitative assessments of risks likelihood and impact to be made, fuzzy computations of risk magnitude and prioritisation, and the selection of appropriate risk actions from the risk action catalogue. The risk handling class allows a project manager to use the results of the risk analysis to recommend appropriate response strategies to be implemented.

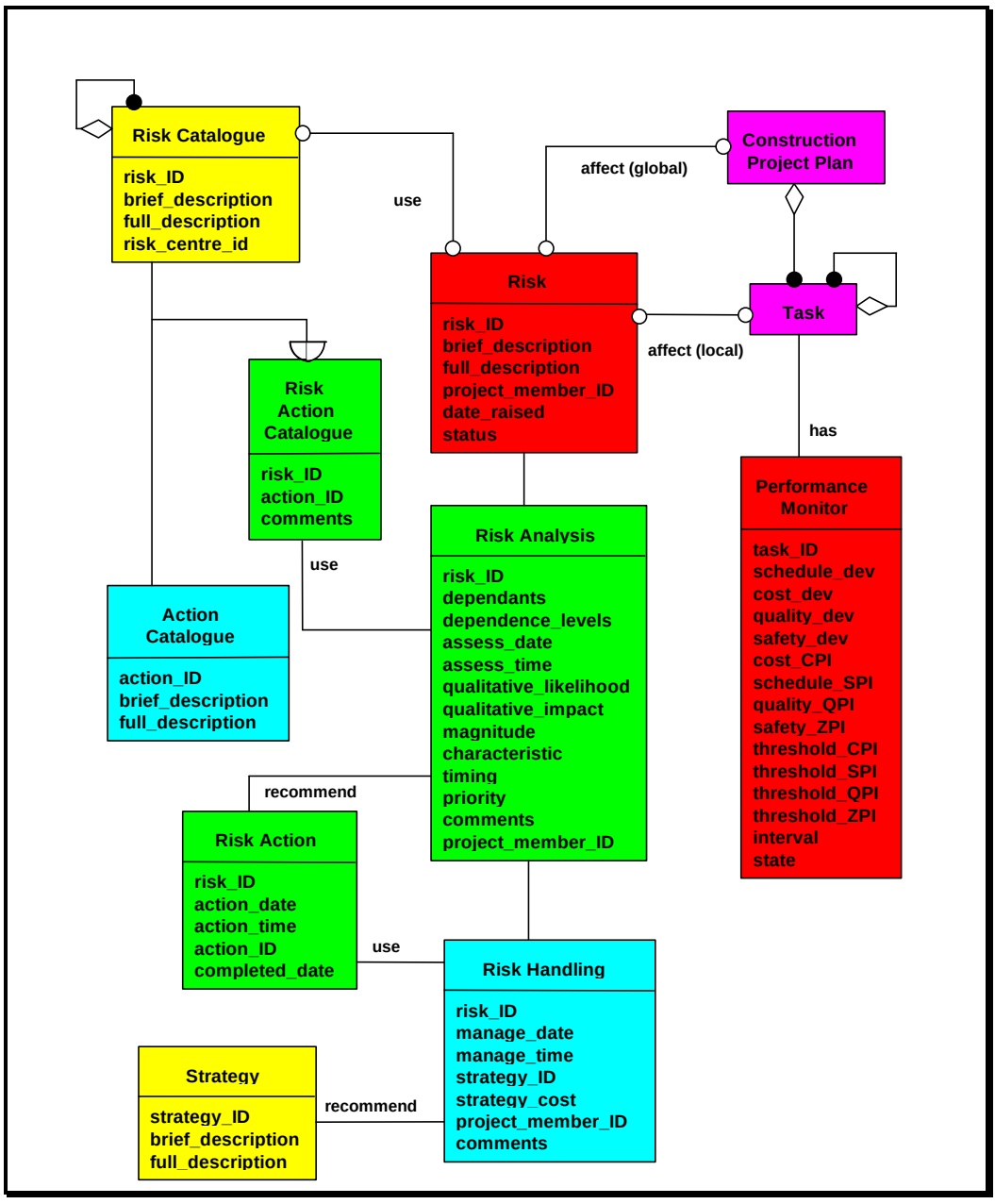

Figure 4. The risk management object model

INTELLIGENT DECISION SUPPORT

We are currently using the object models presented above as the basis for developing new approaches to address the application of KBS techniques within large integrated intelligent software environments for the construction industry. We are currently investigating the use of object-oriented knowledge representation and reasoning techniques together with the use of multi-agent and blackboard architectures to handle the disparate nature of construction disciplines and problems. The object modelling approach has facilitated the development of an ontology of terms for describing project knowledge. The ObjectStore (1994) C++ based object-oriented database development tool has been extended to support C++ -based knowledge-based systems reasoning mechanisms. ObjectStore has been extended to incorporate an object-oriented knowledge-base repository, backtracking and forward chaining inferencing mechanisms. We are using server-enabled active Object-Oriented Databases as the blackboard (or the repository) on which all generic data resides in the form of intelligent object classes which expose their functionality to the outside world. Multiple problem solving agents representing knowledge sources can, therefore, access information, solve problems co-operatively, and contribute information to the blackboard. Agents are being developed for the following problems: an agent for the semantic enhancement of design (CAD) data (Iosifidis*et. al.* (1995), Tah *et. al.*, 1995); an agent for the automatic generation of construction programme work-breakdown structures; several agents for method and resource selection and activity duration estimation for different classes of works; an agent for construction programme formulation from historical data using case-based reasoning; and a risk management agent utilising fuzzy logic for the quantitative assessment and analysis of risks using natural language expressions.

Although, advances have been made in knowledge representation and problem solving techniques, the problem of capturing and storing knowledge of experiences and decisions made as a historical database for future use, remains to be addressed. In the area of project management , the lack of a systematic approach to storing past construction plans and problems that were encountered together with the solutions that were used to overcome these problems means that plans have to be produced from scratch every time a new project is to be planned. We are currently investigating the potential offered case-based reasoning (CBR) to address this problem. Cased-based reasoning is a technique of solving new problems by adapting solutions that were used to solve previous ones, via retrieval and modification.

The work involves determining how much of the information used in construction project planning and control can be captured and stored away for future re-use. Construction programmes for a large number of multi-storey concrete-framed buildings have been analysed and some interesting patterns between structural components and tasks needed for their construction have emerged. There appears to be a relationship between component types and work packets which can be exploited for case-based reasoning purposes. The information requirements for the CBR work necessitated the re-use of all the object models or system components previously presented, which were initially designed for work involving KBS and the integration of design and construction.

CONCLUSIONS

Rather than viewing the world in terms of one individual object acting on another in a neat causal chain, we view the world in terms of decentralised interactions and feedback loops amongst simple components. We are studying how complex behaviours can emerge from interactions among simple components with local rules, and how complex patterns can emerge from interactions among these components. Local rules, though simple, encode the global essence through the inter-relationships amongst simple components. The object classes presented here provide a powerful means for representing knowledge and providing decision support in a large problem area. It has facilitated the use of a blackboard architectures and multiple agents representing knowledge sources to handle the disparate nature of construction disciplines and problems. The object modelling approach has facilitated the development of an ontology of terms for describing project knowledge. Our experience of object-oriented analysis and design is that it is difficult if not impossible to get it right the first time. Continual re-use results in several modifications each time. The classes presented in this paper have evolved through several projects and continue to be enhanced.

However, the adoption of an industry standard methodology has enabled us not to solve every problem from first principles. Rather, we have been able to reuse solutions that have worked in previous projects. This has been due to the recurring nature of patterns of classes and communicating objects in the systems within our problem domains. These patterns address specific design problems and make object-oriented analysis and designs more flexible, elegant, and ultimately re-usable. They help us to re-use successful designs by basing new designs on prior experience and allow the sharing of software components between projects. We believe that the continual re-use and development of such components will evolve standard application information requirements that can contribute towards ISO-STEPs and the Industry Alliance for Interoperability (IAI) efforts in developing application protocols and standards for data exchange and interoperability.

The object-oriented programming techniques of interface-based access to encapsulated data and methods are currently being exploited in components software technology to develop independent software modules that can interoperate and be replaced by modules with the same behaviour. By developing the interface of a component module based on a standard protocol, then the module can interoperate within any system that complies with the standard. Several standards for component software standards are emerging. The most notable are the Common Object Request Broker Architecture (CORBA) from the Object Management Group (1992) and the Component Object Model (COM) developed by Digital Equipment and Microsoft and forms the basis of Microsoft's Object Linking and Embedding (OLE). A detailed overview of these standards can be found in Orfali and Harkey (1995). It is sufficient here to say that these interfaces provide three basic services: unique identification of a running software component, a consistent way for components to present their interfaces or capabilities, and facilities for components to exchange information with one another. The future will be dominated by distributed computing standards such as CORBA and OLE which will allow the deployment of intelligent agents on enterprise networks for the co-operative solving of problems and decision support.

Acknowledgements

The support of the EPSRC in providing funding for this work under grant refs: GR/J42175 and GR/K26516 is gratefully acknowledged.

References

Iosifidis, P., Tah, J.H.M., and Howes, R. "An advanced object-oriented architecture for information exchange through shared objects". Modeling of Buildings Through Their Life-Cycle, CIB Proceedings, Publication 180, August 1995, pp102-110.

Object Design, Inc. "User Guide - Objectstore Release 3.0 for Windows NT and Win 32s". 25 Burlington Mall Road, Burlington, MA 01803, May 1994.

Object Management Group. "Common Object Request Broker Architecture and Specification". John Wiley, New York, 1992.

Orfali, R. and Harkey, D. "Client/Server with Distributed Objects". Byte, April 1995, pp151-162.

Rumbaugh, J., Blaha, M., Premalani, W., Eddy, F., and Lorensen, W. "Object oriented modelling and design". Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1991 .

Tah, J. H. M., Howes, R., and Iosifidis P. "Capturing Semantic Data in CAD for Construction Project Planning". Product and Process Modelling in the Building Industry, Scherer (ed.), Balkema, Rotterdam, 1995, pp287-293.