TRACKING DECISION-MAKING DURING ARCHITECTURAL DESIGN

SUBMITTED: January 2004
REVISED: May 2005
PUBLISHED: June 2005 at http://www.itcon.org/2005/10/
EDITOR: B-C Björk

Grahame S. Cooper, Professor
Information Systems Institute, University of Salford, England
email: G.S.Cooper@salford.ac.uk

Cristina Cerulli, Dr.
School of Architecture, University of Sheffield, England

Bryan R. Lawson, Professor
School of Architecture, University of Sheffield, England
email: B.Lawson@sheffield.ac.uk

Chengzhi Peng, Dr.
School of Architecture, University of Sheffield, England

Yacine Rezgui, Professor
Information Systems Institute, University of Salford, England
email: Y.Rezgui@salford.ac.uk

SUMMARY*: There is a powerful cocktail of circumstances governing the way decisions are made during the architectural design process of a building project. There is considerable potential for misunderstandings, inappropriate changes, change which give rise to unforeseen difficulties, decisions which are not notified to all interested parties, and many other similar problems. The paper presents research conducted within the frame of the EPSRC funded ADS project aiming at addressing the problems linked with the evolution and changing environment of project information to support better decision-making. The paper presents the conceptual framework as well as the software environment that has been developed to support decision-making during building projects, and reports on work carried out on the application of the approach to the architectural design stage. This decision-tracking environment has been evaluated and validated by professionals and practitioners from industry using several instruments as described in the paper.*

KEYWORDS*: decision tracking; architectural design; versioning of information.*

1. INTRODUCTION

Building projects require a design process capable of dealing with an extraordinarily wide range of complex issues. Decisions must be made about the layout, form, appearance, materials, methods of construction and many other factors in order to satisfy the needs of both 'paymaster' and 'user' clients, as well as a wide range of generic and type specific building control legislation. Many different professional consultants are involved in making and approving these decisions. Even average sized building contracts involve design processes which may span many months and possibly years. Larger building contracts may require design decisions to be made over periods extending into many years. Such decisions are multi-dimensional combining together factors which range from the highly subjective to the perfectly objective. The decisions are made by many individuals often belonging to different organisations and having different skills, process maturity and ICT capabilities. They are made over very long periods of time in an iterative manner and are commonly revisited weeks, months and even years after they were originally taken, leaving scope for a variety of potential problems and inconsistencies.
In this context, much research effort is expended on encouraging integration between actors in construction projects by integrating information representations through information model harmonization: for example, the Industry Foundation Classes (see (IAI 2005)) and (Marir et al., 1998), through document-based integration (for

example, the OSMOS project (Rezgui et al. 2001), through advanced use of computer visualization techniques (Liston et al., 2001 and Aspin et al., 2001), and through the use of knowledge management techniques to improve access to related areas of knowledge (for example Bourdeau et al., 2001, Kazi et al., 2001) and the CoBrITe project (Bouchlaghem et al, 2000). In such work, the main emphasis has been placed on the representation and standardisation of information in construction projects to better support collaborative working. However, recent research has demonstrated a need to create systems that explicitly support the decision-making process (Kohli & Devaraj, 2004).

The EPSRC funded ADS project was created to consider the evolution of project information over time, and how to manage the changing information in a project to support better decision-making. Decisions made during the design process are made by many, often non co-located, actors belonging to different disciplines. Moreover, there is a high risk of misunderstandings, inappropriate changes, and decisions that are not notified to all interested parties. When a decision is made, it may be only the knowledge of particular project participants regarding earlier decisions, that prevents unfortunate errors being made, assuming those participants are actually present.

The ideas behind the approach emerged from an earlier project, COMMIT (Rezgui et al., 1998:1; Rezgui et al., 1998:2). COMMIT uses information technology to provide an information management system that addresses information evolution related problems. Essentially COMMIT records decisions and the reasoning behind those decisions in a manner appropriate to decision making in a design process. That is to say it allows for many versions of designed features either simultaneously or in sequence. It does not prescribe a decision making sequence, which is left to the design team, but provides an infrastructure through which they can ensure that all members of the team can be aware of decisions, who made them and when as well as why. However the concern here is exactly how this system will be plugged into a design process. There are several strands to this problem. Firstly how a system captures the decisions from the designers? Secondly how it captures the reasoning behind those decisions? And, thirdly, how designers may access and browse this information about information or be alerted to it?

The ADS project addresses these issues by providing a software layer, above the conventional electronic information management services used by construction projects, that keeps track of meta-information (i.e. information about information) to support the decision-making in a project. Moreover, it is designed to track a wide range of heterogeneous information types across a project rather than only CAD data. The ADS project has been concerned with the application of the above methods in the context of the design stages of a construction project. To this effect, the concepts have been refined, extended, and implemented in the Java language, and the software has been integrated into a mainstream Computer Aided Design tool - Microstation/J (Krishnan & Taylor, 1999). Field trials have been conducted to investigate the use of the techniques in a practical design situation.

2. BACKGROUND AND CONTEXT

Design is a process that characteristically yields solutions that do not map neatly onto problems. Relationships between solution features and problem elements are often unpredictable in good design (Lawson, 1997). Speed and intensity are features of the design process that seem to result from the integrated nature of design solutions. Since so many disparate factors must be considered simultaneously in arriving at design solutions the process has been likened to juggling.

For example, the architect Michael Wilford describes the architect as like “a juggler who’s got six balls in the air…and an architect is similarly operating on at least six fronts simultaneously and if you take your eye off one of them and drop it, you’re in trouble. There is a sequential development but it’s on several fronts simultaneously.” The architect Richard MacCormac explores the significance of this by explaining that “it’s rather like juggling actually, you know one couldn’t juggle very slowly over a long period” (Lawson, 1994). So it is now commonly recognised that designers need and characteristically work with periods of intense high-speed activity in which many decisions are made in a short space of time. Tools to support such decision making then would be in danger of destroying this intense and concentrated nature of the process if they required the designer to break out in order to help an information management system capture decisions and the reasons behind them.

There is another quite practical reason why this is especially problematic in the case of architectural design. Architecture probably sits at the end of the spectrum of design in which the process cost to product cost ratio is at its least favourable. Automobile designers work on the design of mass-produced motorcars costing tens of thousands of pounds each but with design process budgets of millions of pounds. Architects work on one-off buildings costing millions of pounds with a design process budget measured in tens, or at best, hundreds of thousands of pounds. The emphasis in the UK as a result of the Latham Report (Latham, 1994) seems to be an attempt to even further shorten the time and cost of the design process. This is probably counter productive in terms of design quality but certainly conspires against the implementation of elaborate labour intensive information management tools.

These aspects of the speed and integrated nature of design pose considerable problems in implementing the proposed ADS system into a real design process. How does ADS capture its information? Quite simply it would seem almost inconceivable to attempt to implement ADS unless it was in tandem with some form of computerized information representation, capture and storage. The intention behind the COMMIT/ADS system is that it can bring together and relate information contained in many, heterogeneous information systems. However, for the purposes of the ADS project, a Computer Aided Design (CAD) system alone was used as it provided the best practical opportunity for demonstrating the principles. Thus one can assume that in this context ADS could in some way capture decisions and changes by storing the states of the design as recorded in the CAD system and noting the changes which are made to its features.

However in reality this poses a further problem. CAD systems divide into two-dimensional and three-dimensional systems. Those which are only two dimensional have no way of relating features in say an elevation drawing to those in say a plan or a section drawing. Indeed there is no knowledge within the CAD system itself of what the features are or that they may have related representations in other drawings. In many cases there may not even be knowledge in the CAD system that one drawing represents a plan and another an elevation. The three dimensional system does at least overcome this deficiency by storing a single three-dimensional model from which the various commonly used two-dimensional representations are generated.

However even such three dimensional modelling systems may pose yet another problem. They commonly work with a language of prescribed features specific in some way to the system and not necessarily related to the actual features of the real building yet to be constructed or to the features manipulated in the mind of the designer who is reasoning about them. This is a rather sophisticated argument and yet remains central to problem of implementing ADS information capture.

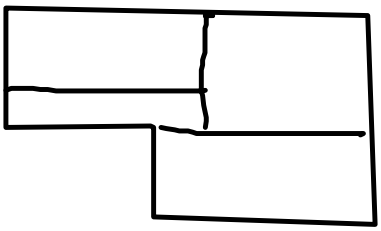

FIG. 1.a

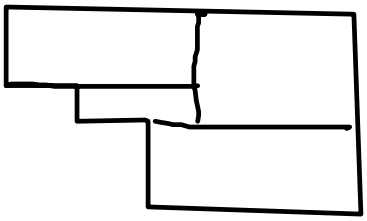

FIG. 1.b

The two simple floor plans below might be sketched at some stage during a design process. The right hand drawing (Fig. 1.b) shows a new state resulting from some decision made by the designer building on the drawing shown on the left (Fig. 1.a).

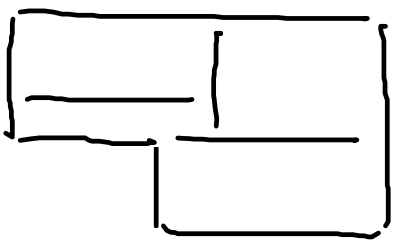

FIG. 2.a

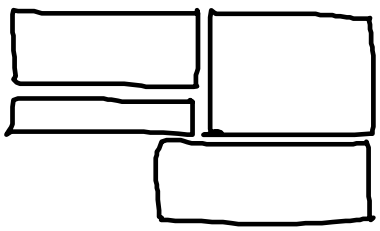

FIG. 2.b

The problem is that although elements within the CAD have been adjusted, it is difficult to know what elements were in the designer's mind at the time nor the reasons for the change. For example the first drawing could be interpreted in many ways including the two given below.

The left hand interpretation (Fig. 2.1) suggests the designer is thinking of the building as a collection of components. The components drawn here are walls, but other components in the designer's mind might also include roofs, floors, windows, doors and so on. In the example on the right (Fig. 2.b) the building is shown as a collection of spaces or rooms. These last two drawings are constructed deliberately to emphasise these different interpretations, and good examples of similar phenomena can be observed by analysing actual sketches done by architects during their design process. However once the designer is working with a CAD system it is likely that the vocabulary of the system will take over rather than the more natural expression of the designer's mind. This of course illustrates a major deficiency of CAD systems as currently realised (Lawson & Roberts, 1991). As design tools they fall well short of the power of expression and natural interface of the pencil and paper!

So to return to the design process, it is possible to capture the change made between the first two drawings, but only in terms of the vocabulary of the CAD system. Such a change may have been made for a number of quite different reasons. For example it may have been to save cost by reducing unwanted floor area in the corridor. It may have been made in order to increase the size of the main room. It might have been to enable daylight to penetrate directly into the main room behind the corridor. It might have been made to create a particular massing on the outside of the building. It might have been to enable the building to move nearer to a site boundary running across the scheme at an angle. Many other perfectly good reasons for such a change can be imagined, or of course a combination of some or all of them! All these important issues have been taken into account and tackled by the ADS project which has refined the concepts previously researched within COMMIT as described in the paper.

3. OVERVIEW OF THE APPROACH

The ADS approach is concerned with creating mechanisms that proactively encourage collaboration and sharing of project knowledge. In order to achieve this, the approach attempts to address the following three primary issues:

(a) **Recording of intent behind decisions leading to new or changed information.** It is important when changes are made in a project that earlier decisions can be examined where appropriate. ADS provides support for recording the factors that influence decisions that lead to information being produced or changed. Automated support for this is used as much as possible in order to keep to a minimum the degree of intrusion of the system into the human processes.

(b) Tracking of dependencies between pieces of information and their related decisions. When a change is made, it is important to be able to find out which earlier decisions are likely to be affected by the change. To facilitate this, ADS provides support for tracking which versions of which information resulted from a particular decision. Together with issue (a) above, this provides a potentially powerful way to record dependencies between different decisions that have been or are being made during the project.

(c) Notification and propagation of changes. Sometimes it is useful for an actor (human or computer-based) to register an interest in a particular part of the project information, ensuring that objects and actors are kept informed of relevant changes introduced to the project information base.

These three primary issues result in a need to address three secondary issues in the design of ADS. These are:

(d) **Ownership, rights and responsibilities.** Whilst the control of access rights is not a primary concern of the ADS project or approach, it is important in the context of decision tracking to be able to identify the actors involved in making and implementing decisions. The actors concerned will carry tacit knowledge relating to any changes in which they were involved. The control of access rights acts as an inducement to the actors to identify themselves before making changes to the information. Each actor is assigned a specific role (or a number of roles) in the project through which he or she interacts with the project information base. In this way, it can be possible, for example, to find the people involved in an earlier decision in order to consult them.

(e) **Versioning of information.** A mechanism is used to keep track of all the states in which an object has existed, including its current state. These versions are linked to decisions, with associated rationale information. This is important as the old versions of objects hold the information that was available when a past decision was made, and will form the basis for many other decisions carried out in other parts of the project.

(f) **Schema evolution.** The lifetime of a construction project can be very long. It is important that the underlying schema, which provides a context for interpreting data, is able to change over time, but without changing the meaning of existing information and without affecting the overall consistency of the project information base. Indeed any schema, which represents a description of the kinds of information available at a given time, form part of the information that was relevant at that time, and so needs to be subject to the same management as any other information in the project.

The ADS project is concerned with applying these concepts to construction design, and addresses issues of user interface design, mechanisms exploiting the networks of information that are built up within the COMMIT system during a design project, and ways of introducing such a system into the design process in such a way that any negative impacts on the process are minimized. Taking into account the level of IT maturity of the user organisation, which is in step with typical construction design organisations, the ADS project has focussed primarily on the areas of access rights, change tracking and capture of rationale, though the prototype infrastructure is capable of dealing with notification as well.

4. VERSIONING AND DECISIONS

The primary focus of the ADS project was the treatment of versions of information objects and their relationships with decisions made during the design process in order to understand better the consequences of decisions that result in changes to the information.

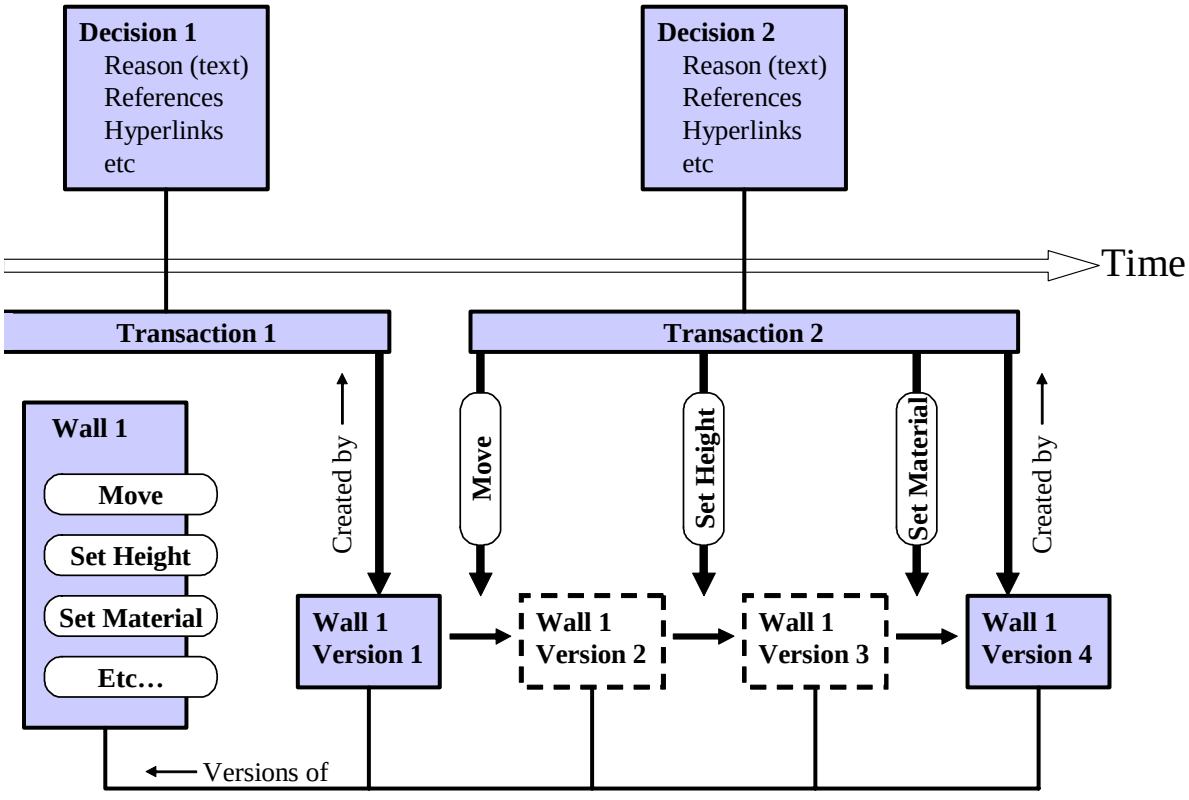

FIG. 3: Object versions and decisions in the ADS model

The ADS approach (see Fig. 3) characterises the project as a series of decisions, each of which results in some changes being made to the information in the project. Each decision is associated with a *transaction*, which

consists of a number of operations carried out on a range of different objects over a period of time. One transaction may involve operations on many different objects of different types, though the diagram shows changes only to one object.

Figure 4 shows a model at the class level of the relationships between object versions, the operations that created them, the decisions that resulted in those operations being performed and the transactions that are used in the system to encapsulate those decisions. Every object is held as a number of versions, each of which was created by a single transaction, which implements a particular decision. The object version was created by the invocation of a particular operation, which can be used to identify what was actually done to the object to create this version, and a record is kept of the actor that invoked the operation.

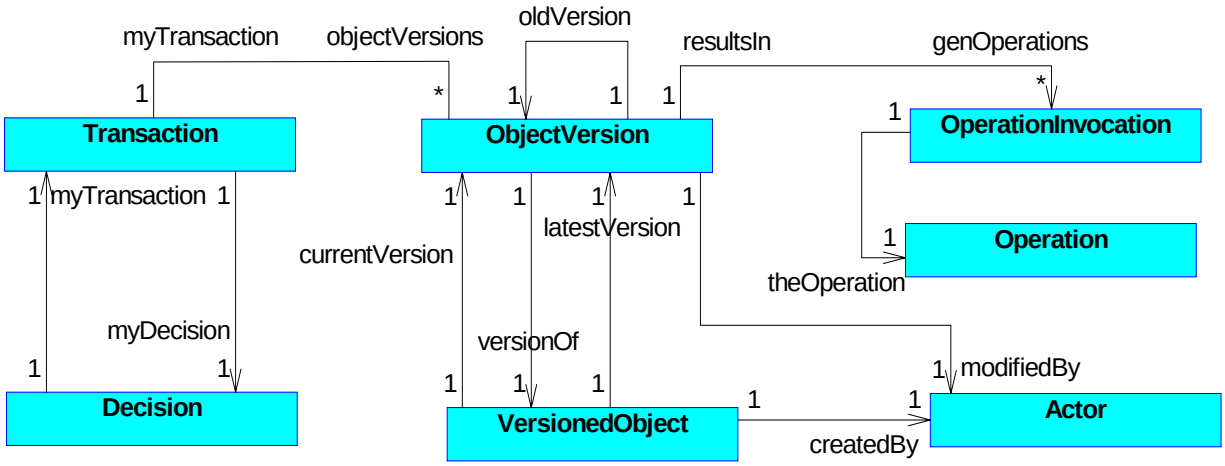

FIG. 4: Class level model of versioning and decisions in ADS

The decision itself is represented as an object that may contain a textual reason for this group of changes, along with any reference material in the form of references and hyperlinks to other electronic materials. It can also contain information on decision types, the actor that initiated the decision, authorizations and other project-related data. The ADS system builds a network of such decision-version "chains" as the users work on the project, effectively linking decisions together and allowing the dependencies between decisions to be analysed during or after the design process.

5. ACTORS, ROLES AND RIGHTS

5.1 5.1. The Underlying Concept

As well as recording which actor has created each version of each object, the ADS system can control which actions (or "Operations") may be performed by a particular actor on each object in the system.

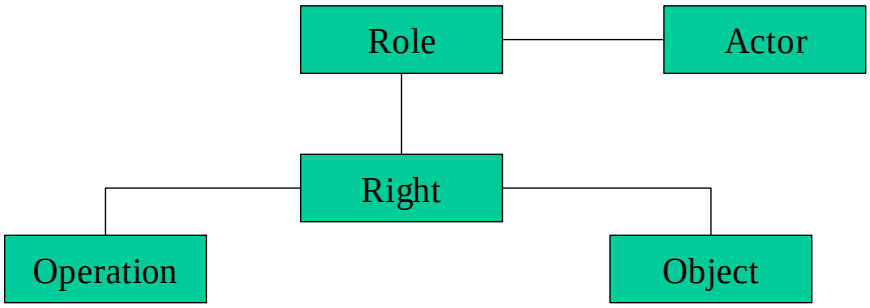

FIG. 5: Basic model of roles and rights in ADS

The basic model is shown in Figure 5. An actor can take one or more roles in a project. A role represents a set of responsibilities that have to be discharged by the actor(s) taking that role. In order to discharge those responsibilities, the actor must be given a number of rights, each of which allows the actor to carry out a certain

operation on a particular object. The operations that are available to be carried out on an object are determined by the type (or class) of the object in the normal object-oriented way.

There are potentially a large number of rights to be recorded in the system for various permutations of roles, operations and objects. For this reason, it makes sense to introduce approaches to defining default rights for whole collections of objects. These default rights will be applied in the absence of any relevant specific rights.

In the current implementation of ADS, the concept of a *TypeRight* exists, which is a right that is conferred on a particular role for all objects of a particular *ObjectType*. This is one form of default right and is the one mechanism currently available for default rights in the ADS system. However, this should not be thought of as the only approach possible, or necessarily the most useful, and it has been clear from the ADS workshops that other approaches need to be introduced to augment this one.

5.2 The Object Model for Actors, Roles and Rights

The object model implemented in the ADS prototype is somewhat more complicated than described above because of the overlaps with other aspects of ADS, particularly versioning. This model is shown in Figure 6. Here, the concepts of *SpecificRight* and *TypeRight* may be clearly seen, along with the relationships to *Operation* and *Role*. In the figure, the *Actor* concept has been left out to avoid complicating the diagram further.

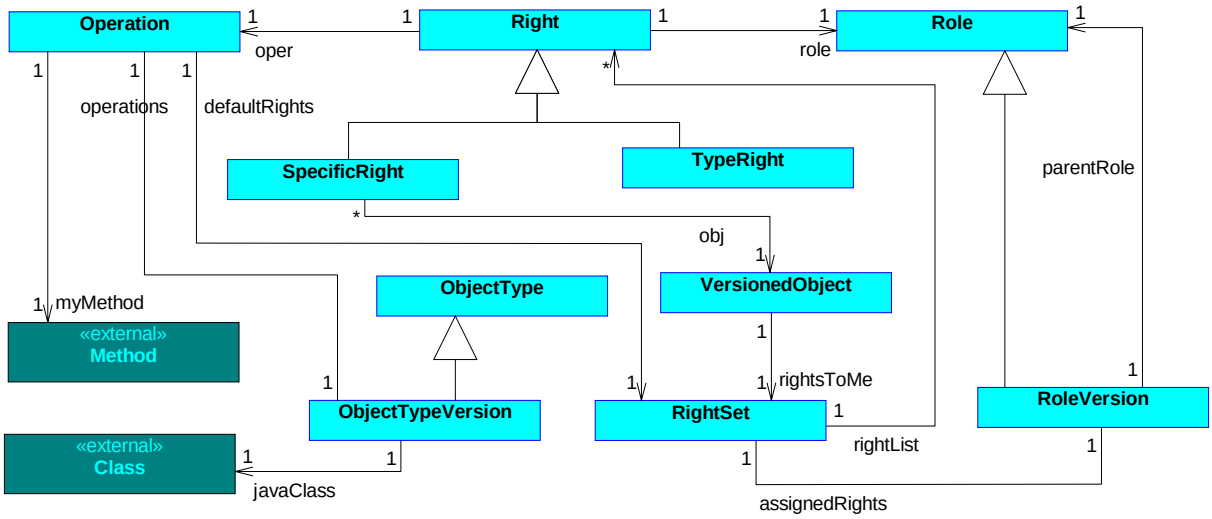

FIG. 6: The class model for Actors, Roles and Rights in ADS

Within the ADS model, the concepts of *Role* and *ObjectType* are also versioned, which adds some complications to the model. Since a change may be made to a role, such as adding or removing rights, it may be important to know what the role comprised at some time in the past, when a particular transaction was carried out. It is for this reason, and for controlling which role is allowed to make changes to roles, that the *Role* class is versioned. The *ObjectType* class is versioned for the purpose of schema evolution, since it may be important to know the nature of an *ObjectType* at the time an old version of an object was current if historical data is to be understood.

In the Java implementation, the *Operation* concept is implemented as an abstraction of the Java Method, and this is shown in the diagram. Similarly, the *ObjectTypeVersion* corresponds to a Java class.

The implementation of rights in ADS has involved a high degree of abstraction of the basic concepts, and this is described the section on software implementation. In order to encapsulate the storage and management of rights, the concept of a *RightSet* was introduced into the model as shown in the diagram.

6. NOTIFICATION AND CHANGE PROPAGATION

Due to a number of technical constraints and limitations in the way users have applied ADS, notification and change propagation were not extensively exercised during the ADS project – the emphasis being primarily placed on the relationship between decision-making and change tracking. However, the underlying framework

for this is present in the ADS model and software. Figure 7 shows the Java class model implemented in the ADS prototype in respect of this feature.

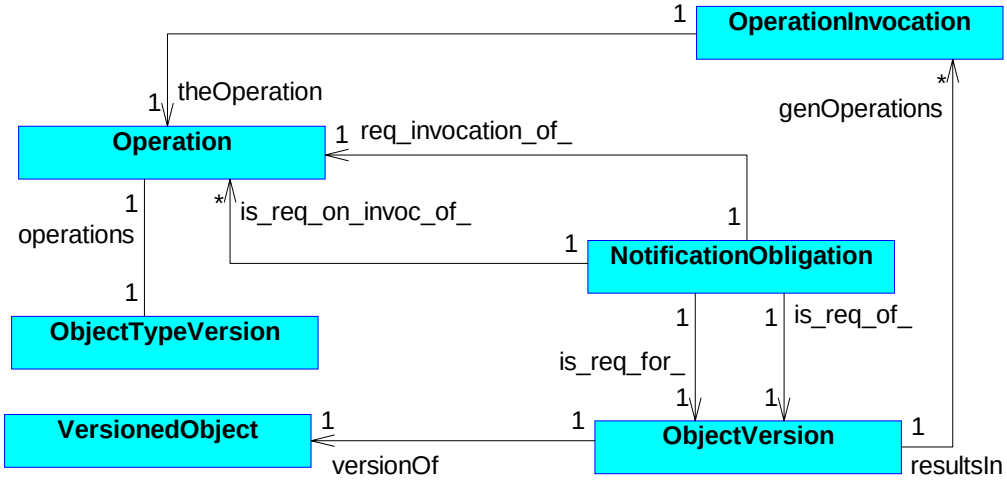

FIG. 7: The class model for notification and change propagation in ADS

The model revolves around the concept of a *NotificationObligation*. The existence of a *NotificationObligation* indicates that when a particular *Operation* is carried out on a particular *Object* (or *ObjectVersion*), then some *Operation* must invoked on another *Object*. This second operation invocation represents a notification to the second object that the first has been changed in some particular way.

One way of making use of this feature is to require that a particular actor (or role) is informed when certain objects are changed in the system. The user-interface for this functionality, which has been demonstrated in an earlier version of the model following the COMMIT project, is being rebuilt as a separate effort outside of the ADS project. This will be reported in detail in a subsequent paper.

7. THE DECISION BROWSER

Figure 8 shows the basic user interface of the prototype decision browser, which is the main user interface tool for tracking the relationships between decisions. This tool may be accessed by the user at any time as they are working on a design. On the left of the diagram is a list of transactions that have been carried out on the design information contained in the CAD tool, labelled by the date and time at which they were committed (completed). For the end user, the distinction between a transaction and a decision is not made since this could be confusing, and only the decision concept is visible. The topmost transaction in the list is the one that is currently being carried out, which relates to some changes in a drawing to move various parts of the design, including a services riser, in order to accommodate a requirement of the client. The tabbed area on the right displays information about the decision that relates to the transaction that is selected on the left.

Below the current transaction in the list are shown previous transactions in chronological order. Two of these, committed at 15:15:03 and 15:03:59, are highlighted red to indicate that during each of these transactions, some operations were carried out on drawing elements that are also affected by the current transaction. By selecting one of these, it can be seen that the reason for that previous change was to make additional room in the service riser to satisfy some need, indicating how the previous decision is related to the one currently being expedited. In the example, it can also be seen who was responsible for the change, so that this person may be contacted for further details if necessary.

In the example shown, the transactions of interest are separated by only minutes. However, in a real-life situation, they could be separated by weeks or months. In these circumstances the ability to extract dependent decisions out of the thousands that would exist on a project could reduce the need to rely on the serendipitous application of tacit knowledge in the decision-making process.

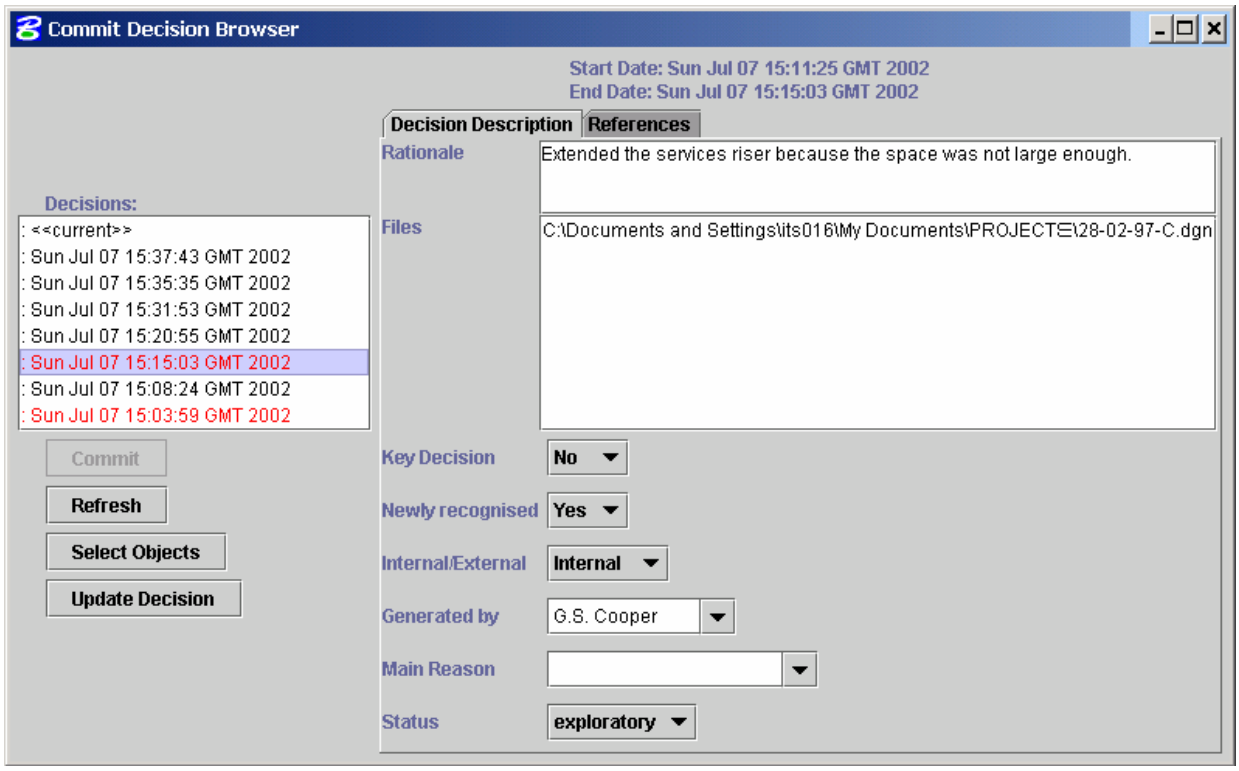

FIG. 8: The ADS Decision Browser.

8. ADS SOFTWARE ARCHITECTURE

An important goal in the design of the ADS software is to maintain independence from any particular type of information storage or modelling tool, in order to support integration of heterogeneous information from arbitrary sources. This was achieved by careful attention to the software architecture.

Figure 9 shows the overall architecture of the ADS prototype. This is shown as three separate layers.

At the top of the diagram is the ADS software itself, which is written entirely in pure Java and consists of three main parts. The first is a library of Java objects that implement the underlying concepts of ADS (the ADS API), along with a repository that provides a container and simple object persistence. The second is a User/Rights Manager, which is used to set up an ADS database including actors, roles and rights. The third part is the Decision Browser, which allows the user to explore decisions and associated information. The Decision Browser also provides the facilities to create and commit transactions and manage the information in the associated decisions. There is also a browser that provides access to a tree representation of all the objects managed under the ADS system, including all old versions. The latter is not intended for end-user use, but is mainly used in the development of the ADS system itself.

At the bottom of the diagram is the application software, in this case Microstation/J. This is a popular, advanced CAD system that provides a Java-compatible application programming language called JMDL. This language has been used to provide the integration between Microstation and ADS, which is implemented in the Adapter layer.

For the purpose of the investigations carried out in the ADS project, the prototype was linked exclusively into a CAD tool (Microstation/J). However, the information that is relevant to decision-making in a project is held in many forms, and it is important that all those forms can be brought together within the ADS framework. Because of this, care was taken to maintain a logical separation between the implementation of the ADS model and the realization of versions and related concepts within a particular application. The adapter layer is where this realization takes place in classes that are specializations of the *ObjectVersion* class. By creating adapters for various different forms of information storage, such as word-processed documents, spreadsheets, and also object

databases supporting standards such as IFC (the Industry Foundation Classes) [IAI, 2001], it is possible to bring together these different forms of information within the ADS framework.

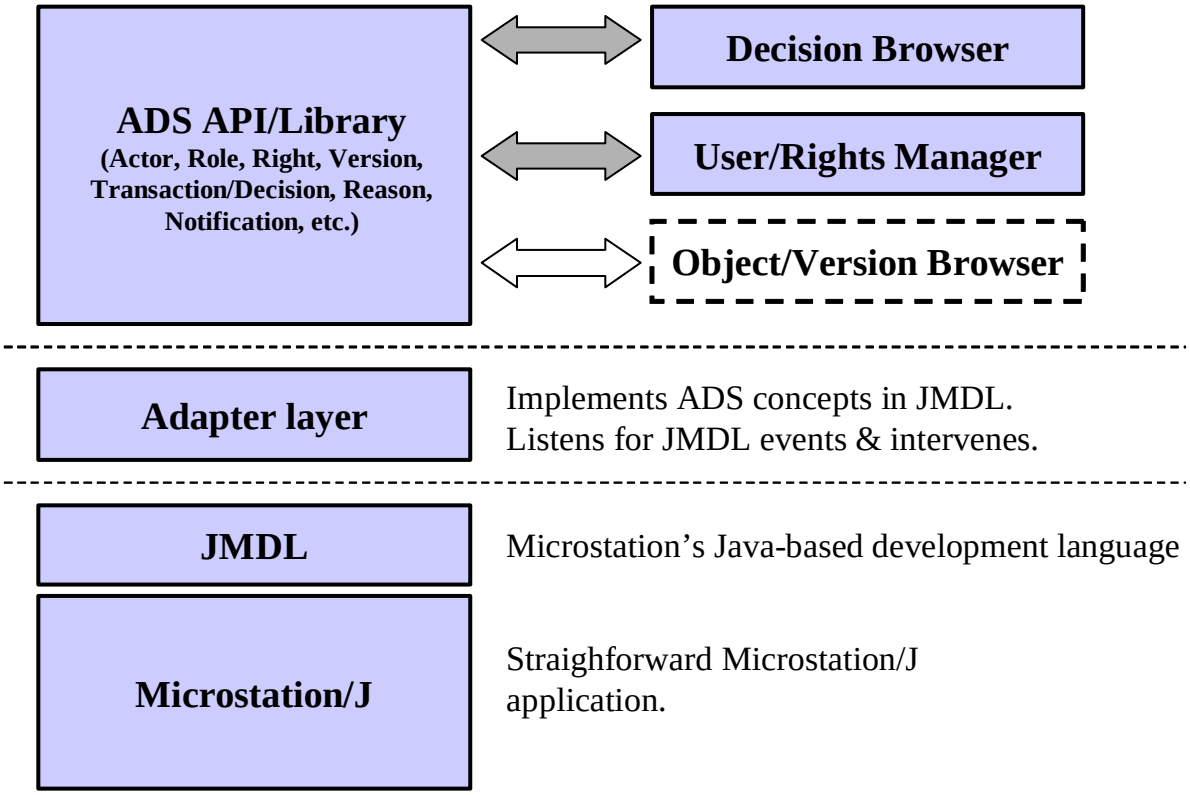

FIG. 9: The Architecture of the ADS Prototype

Figure 10 shows the underlying model used to separate the software implementing the ADS core (known in the model as "Commit") from the domain objects that represent the functionality of the domain applications. The dotted rectangles denote the separate packages that make up the software, and some of the classes from those packages are shown.

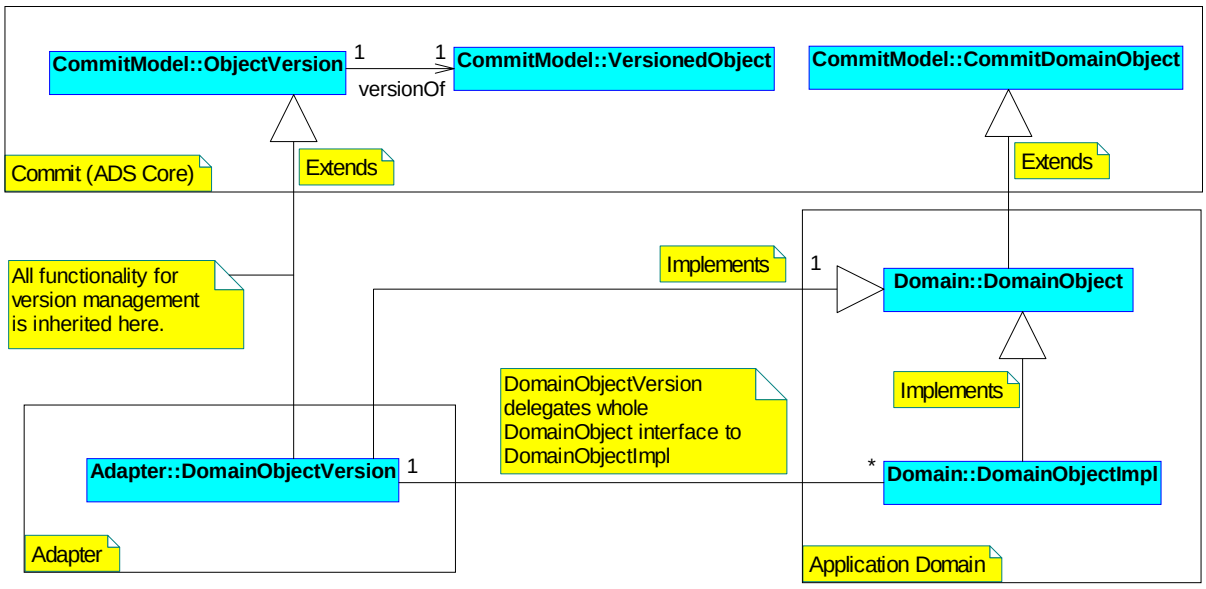

FIG. 10: Conceptual model of ADS/Application Domain separation

The package labelled Commit (ADS Core) contains the classes that make up the underlying ADS model, which are implemented in pure Java and are intended to be independent of any particular domain application.

The package labelled Application Domain represents classes that are specific to the particular application being integrated with ADS. This model is defined for conceptual purposes – the actual application domain classes, and their structure, being dependent on the particular application technology involved. Unless the domain application has been written specifically with ADS in mind, this part of the model will simply be replaced by some conceptual abstraction of the concepts implemented in the application, which may or may not be implemented directly.

The package labelled Adapter is where the ADS Core model and the Application Domain model come together. In this package, the application-specific elements of the abstract concepts represented by the ADS Core model concepts are implemented in concrete form. Particular issues that are addressed in this layer concern the way in which the versioning and storage of individual domain objects is interpreted in relation to the information storage of the application, and the way in which access controls may be applied at the level of the application itself. The latter issue can be quite tricky as it requires some way for the ADS software to intervene in the operations that the user carries out within the application.

In the model shown above, the class responsibilities are as follows:

- ***VersionedObject*:** participates in the (Commit) collaboration regarding roles, rights, etc., and provides a service to instances of *DomainObjectVersion* to check access rights whenever a method is called. It also records which instance of *ObjectVersion* is the current version.
- ***ObjectVersion*:** implements all the functionality required for recording versions from the point of view of the ADS system. To maintain the independence of ADS from any particular domain software, this class shouldn't know anything about Domain concepts.
- ***DomainObjectVersion*:** provides to ADS an interface for a particular domain object, which should be based on the appropriate *DomainObject* Java interface. An example of this in the ADS system is class *DgnObjectVersion*, which implements the interface *DgnObject*. The implementations of the *DomainObject* methods should check the rights of the user as required, then implement the required functionality with appropriate use of delegation to the domain application. In the simplest case, the whole interface functionality would be delegated to an appropriate *DomainObjectImpl*.
- ***DomainObject* (interface):** defines those methods that are used to access a particular domain object which should come under the control of ADS. This interface is the source of information on methods for defining access rights and notification obligations, etc. It may be an interface that is provided by the domain application, in which case its position in the architecture is as shown in the diagram. Alternatively, it may be an interface that is created to model the behaviour of the domain application specifically for the purposes of Commit, in which case it forms part of the wrapper area of the model.
- ***DomainObjectImpl:*** is purely a conceptual device in the model to represent whatever functionality is provided by the domain application to support the implementation of DomainObjects. Any shortfall in the responsibilities it accepts will need to be made up by the relevant DomainObjectVersion class, which acts as a wrapper around the functionality of the domain application. In the Microstation integration used for ADS, this is replaced by an object that listens for events reported by Microstation regarding changes to drawing elements, and maps these into changes in the ADS object representation: DGNObject and DGNObjectVersion.

In the actual ADS implementation, the Microstation/J elements had to be written using Microstation/J's JMDL language, which uses its own compiler, whilst the ADS model part was written in pure Java and compiled using Sun Microsystems' Java compiler. This helped to ensure that no domain-specific concepts crept into the core ADS software itself.

9. EVALUATION/DEVELOPMENT METHODOLOGY: FEEDBACK SUPPORTED CONTINUOUS DEVELOPMENT

Throughout the project, feedback from users and members of the construction Industry was used as a tool for refining the system, generating several development cycles. Several mechanisms were used at different stages to collect feedback about the system, its usability and its appropriateness for construction design practice: ***(a)*** Retrospective case studies: two case studies were carried out using project historical data to populate the ADS system; ***(b)*** Workshops: three workshops were conducted inviting practitioners, academics and various construction industry professionals; ***(c)*** Field Study: one live case study was carried out testing the ADS prototype over a 4-month period on a real ongoing project in which feedback from the users was recorded throughout the experiment; ***(d)*** Interviews with practitioners: a series of interviews were conducted with members of leading practices to disseminate objectives and results of the ADS Project as well as collecting system feedback and broadening the scope of user requirements gathering.

Throughout these processes, frequent enhancements were made to the prototype, taking on board issues raised through the evaluative studies. In cases where changes would be too extensive to implement, they were recorded for future development.

A number of design decisions were recorded into the system. Figure 11 shows one of the decisions recorded during Field Trials - Phase I committed by the actor *Garrett, S.* in her role of *Architect*, to which the *rights* of creating/deleting/modifying elements in the model had been assigned. The user was left totally free to determine at what point to commit a decision, the amount of information to insert and the number of design changes to be included in a single decision or transaction.

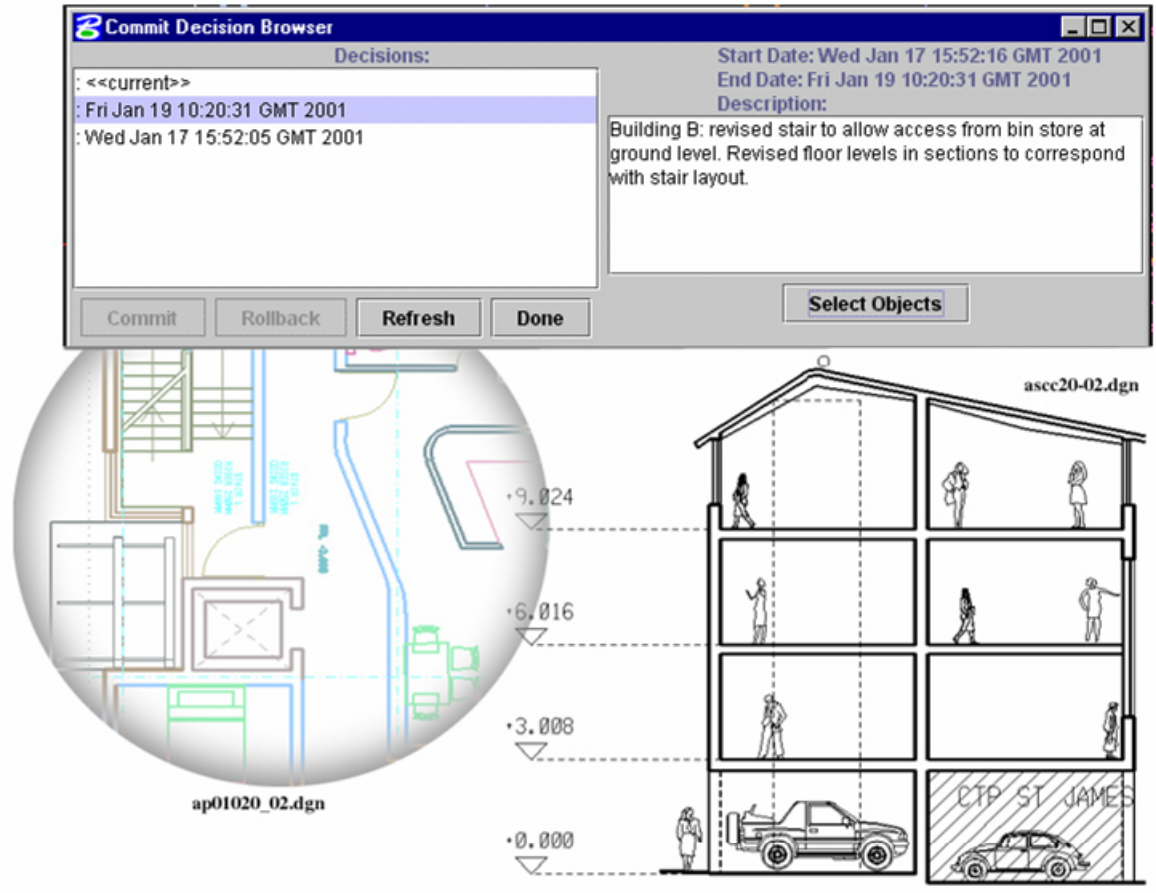

FIG. 11: Illustration of a design decision

The rationale for that decision was input in an unstructured form in a free-text box, and, for the decision in examination, reads as follows: “Building B: revised stair to allow access from bin store at ground level. Revised floor levels in sections to correspond with stair layout”. The *Select Objects* button allows highlighting the CAD objects involved in the decisions when one of the files containing them is open.

9.1 Feedback – Dissemination Interviews

A number of interviews with members of the industry were carried out both at intermediate and final stages of the project. The interviewees were holding intermediate or senior positions within some of the leading architecture and engineering practices.

The aims of the interviews were threefold:

1. Dissemination: to present the ADS prototype, raising awareness about the issues it aims to address and disseminating the research carried out within the project.
2. User Requirements: to broaden the user requirements gathering spectrum to a number of practices known for being centres of excellence, other than the project’s industrial partners.
3. Feedback: to gather feedback on the ADS prototype, both at system and interface level.

9.2 Outcomes of Evaluations/Trials

Much of the feedback and analysis from the evaluation has been used to enhance the user-interface and functionality of the prototype. Work has also been carried out on a classification of the types of design decisions according to their origin, status and relevance in the project.

At the time of the field studies the ADS system as a recording tool was still under testing and under development. The data structure proved to be versatile, easily accommodating changes and developments in the software architecture.

Minimum intrusiveness is crucial to the success of any decision support and design rationale-gathering tool and to avoid any interference with the process with ADS the granularity of decisions is determined entirely by the system user.

The benefits deriving from recording design rationale are proportional to the quantity of data gathered and, possibly, inversely proportional to the granularity of events, decisions and transactions. A potential impediment or deterrent to the data gathering is the fact that it is likely that the main beneficiary of such an activity is not going to be the very person that is requested to input the data into the system. But there is also a cultural dimension of the construction design process that determines the success of design rationale and project information capturing. It is possible to envision a gradual increase in the amount of information recorded into the system as the designers become more aware of the real potential benefits of recording design rationale.

The user response was very sympathetic towards the overall objectives of the system. Frustration was occasionally expressed towards the limitations of specific implementations. In particular limitations in processing speed were pointed out as disruptive and intrusive.

During the field trials, the need to fine-tuning the system data recording functionalities to the user needs necessarily shifted the focus on the ADS system as a data gathering tool rather than a design aid tool. Future developments of the ADS system will need to place more emphasis on improving the retrieval of information and to implement extra functionalities like the notification of changes to potentially affected objects, mechanisms for mapping relationships between decision (affected and pending decisions) and the nesting of design decisions.

10. FUTURE RESEARCH AND DEVELOPMENT

Although the architecture of the ADS prototype has been designed to separate the implementation of ADS itself from the integration into particular applications, it has been difficult at times to achieve this. For efficiency reasons, it has been necessary to run the ADS software within Microstation/J itself (under Microstation/J’s own Java Virtual Machine). This is clearly undesirable if we wish to integrate information of many different type under ADS, and also to implement it in a fully multi-user, networked environment, as is the ultimate intention.

To achieve this with reasonable performance, it may be necessary to work at a coarser level of granularity than the drawing element level.

Certain user demands have led to some compromises in the software architecture. In particular, the desire to have lists of drawing files in the Decision Browser is a perfectly reasonable request. However, it does mean that the Decision Browser, which is part of the ADS subsystem, contains elements that are specific to file based information such as that managed by Microstation/J. However, the approach and the software are not tightly coupled to the CAD package nor to CAD in general. Indeed, the greatest benefit from the ADS approach may be gained by it use in integrating heterogeneous information from many applications around the decision-making process.

In its application, the ADS project has concentrated primarily on the decision tracking elements of the approach. However, the system is also able to enforce obligations for notifying other, dependent objects (including Actors and Roles) whenever certain operations are performed on certain elements. This could further help to ensure that design imperatives and other dependencies are not overlooked when changes are made in a project, allowing the ADS software to take a more pro-active role.

The ADS approach and prototype do not directly address issues of branching and merging of design paths. These are important both for the support of long transactions, which may be expected to be fairly typical in design work. They are also important for the support of parallel lines of design activity, supporting the exploration of alternative designs. Branching and merging support a process whereby different actors may make independent changes to the same parts of the project information (branching), and the resulting alternative versions are later merged back together, with any conflicts being resolved (merging). This concept is very well-known to software developers using software such as CVS (Concurrent Versions System (CVS, 2005)), and has been addressed by Bentley Systems themselves in the Microstation CAD context through the development of ProjectBank (see for example: Bentley, 2000). The ADS approach would be enhanced considerably by providing a way to manage branching and merging across many different types of information.

The ADS system is capable of gathering very large amounts of data. One particular, relatively unsophisticated approach to applying this data in the design process has been illustrated here. Further experience with the use of the tool will enable more complex ways of analysing the data to be developed, making access to the information simpler and more powerful for users. More sophisticated analysis could be facilitated through the use of a database to store meta-information rather than the simple file-based approach currently used. This, coupled with the integration of different forms of information could make the ADS approach a powerful tool in improving decision-making in collaborative projects in construction and other areas.

11. ACKNOWLEDGEMENT

The authors would like to acknowledge the support of the EPSRC (Engineering and Physical Sciences Research Council) in funding the ADS project under contracts GR/M42381, GR/M42398 and GR/R53463. We would also like to acknowledge the support of Bentley Systems in providing software and support, BDP for participating in field trials and evaluation activities, and members of the ADS advisory committee for providing valuable feedback during the project.

12. REFERENCES

Aspin R, DaDalto I, Fernando T, Gobbetti E, Marache M, Shelbourn M, Soubra S. (2001) A conceptual framework for multi-modal interactive virtual workspace, Electronic Journal of IT in Construction (ITcon), Vol. 6, Special Issue Information and Communication Technology Advances in the European Construction Industry, 149-162, http://www.itcon.org/.

Bentley (2000) Banking on a bright future. bentleyuser.org. http://www.bentleyuser.org/features/2000/march/f00005.htm (as of 29th May 2005).

Bouchlaghem D, Rezgui Y, Hassanen M, Rose D, Cooper G, Barrett P, Austin S. (2000) IT Tools and Support for Improved Briefing, CIB W78 - IABSE - EGSEA AI International Conference on Construction Information Technology 2000 (CIT2000), Reykjavik, Iceland, 28-30. The Iceland Building Research Institute.

Bourdeau M, Giraud-Carrier F, Rezgui Y, Zarli (2001) A. Knowledge Management in the Construction Industry: the e-COGNOS Approach, Proceedings of eBusiness and eWork 2001, (Venice, Italy, 17-19 October 2001). IOS Press.

CVS (2005), Concurrent Versions System. http://www.cvshome.org/ (as of 29th May 2005).

IAI (2005), International Alliance for Interoperability. http://www.iai-international.org/ (as of 29th May 2005).

Kazi, AS, Hannus M, Laitinen J, Nummelin O. (2001) Distributed engineering in construction: findings from the IMS GLOBEMEN project , Electronic Journal of IT in Construction (ITcon), Vol. 6, Special Issue Information and Communication Technology Advances in the European Construction Industry; 129-148, http://www.itcon.org/.

Kohli R, Devaraj S. (2004) Contribution of institutional DSS to organizational performance: evidence from a longitudinal study, Decision Support Systems;Vol.37, 103– 118, Elsevier.

Krishnan GV, Taylor JE. (1999) Harnessing MicroStation J, Thomson Delmar Learning.

Latham M. (1994) Constructing the team, UK Government Commissioned Report.

Lawson BR, Roberts S. (1991) Modes and features: the organization of data in CAD supporting the early phases of design. Design Studies 1991; Vol.12 (2); 102-108. Elsevier.

Lawson BR. (1994) Design in Mind. Oxford, Butterworth Architecture.

Lawson BR. (1997) How Designers Think. Oxford, Architectural Press.

Liston K, Fischer M, and Winograd T. (2001) Focused Sharing Of Information For Multi-Disciplinary Decision Making By Project Teams, Electronic Journal of IT in Construction (ITcon), Vol 6, 69-82, http://www.itcon.org/.

Marir F, Aouad G, Cooper GS, (1998) Osconcad: A Model-Based Cad System Integrated With Computer Applications, Electronic Journal of IT in Construction (ITcon), Vol 3, 25-44, http://www.itcon.org/.

Rezgui Y, Brown A, Cooper G, Brandon P. (1998:2) Intelligent Models for Collaborative Construction Engineering, Micro Computers in Civil Engineering; Vol.13 (2); 151-161.

Rezgui Y, Cooper G, Brandon P. (1998:1) Information Management in a Collaborative Multi-Actor Environment, Journal of Computing in Civil Engineering 1998; Vol.12 (3); 136-145, ASCE.

Rezgui Y, Cooper GS, Zarli A, Marache M, Kazi AS. (2001) Web-based generic services for the construction virtual enterprises in the OSMOS project. Proceedings of the CIB W78 International Conference of IT in Construction. 29 May-1 June 2001, Mpumalanga, South Africa. CSIR, Pretoria, South Africa..