SYSTEM ARCHITECTURES FOR AEC INTEROPERABILITY

Thomas Froese[1], **Kevin Yu** [2], **Kathleen Liston** [3], **Martin Fischer** [4]

[1] Associate Professor, Department of Civil Engineering, University of British Columbia, Vancouver, BC, Canada, V6T 1Z4, tfroese@civil.ubc.ca, http://www.civil.ubc.ca/~tfroese/

[2] Ph.D. candidate, Department of Civil Engineering, University of British Columbia; Software Engineer, Timberline Software Corp., 15195 NW Greenbrier Parkway, Beaverton, Oregon 97006, http://www.timberline.com

[3] Ph.D. Candidate, Department of Civil and Environmental Engineering, Stanford University, Stanford, CA 94305-4020, USA, kathleen@listons.net

[4] Associate Professor, Department of Civil and Environmental Engineering, Stanford University, Stanford, CA 94305-4020, USA, http://www.stanford.edu/~fischer

ABSTRACT: This paper addresses architectures for distributed, model-based, integrated information systems for AEC/FM. The paper discusses the motivating factors that hinder the ability of current information techniques to support effective information sharing and collaborative decision-making. It describes the results of a prototype implementation of a model-based integrated system, which integrated a 4D-planning tool with commercial scheduling and estimating applications through a shared, model-based project database. The paper then presents a typical set of components for integrated systems: a reference architecture. Finally, the paper describes the benefits of supporting multi-modal architectures (i.e., the ability to support a wide variety of data exchange paradigms through the distributed system) and transaction-based services.

KEYWORDS: Distributed Systems, Visualization, Integration, Project Management, 4D, System Architectures

1. INTRODUCTION

Recent trends in information technologies move us closer to the goal of integrating the computing tools used throughout architecture, engineering, construction, and facilities management (AEC/FM)—a long-standing topic of research. These trends include the explosion of interest and technologies for business-to-business Internet applications and an on-going maturation of industry data standards such as the Industry Foundation Classes (IFCs) (Froese et al. 1999). In our research into integrated systems for project management, we have previously focused on data representation and standards for project information (Froese et al. 1997) and on visualization environments that allow project stakeholders to interact with the integrated project information [Schwegler et al., 2000]. These environments, however, do not support multi-disciplinary interaction and decision-making.

Construction planning decisions usually affect many project stakeholders, from workers to various subcontractors to project managers to facility users, etc. Hence, coordination of all these perspectives is critical to maintain smooth progress on a project. Decisions must be based on many kinds and sources of information (3D CAD models from architects, cost estimating spreadsheets and construction schedules from the contractor, milestones from the owner, etc.). Typically, decisions must be made quickly to maintain the pace of a project. We have observed that, today, no tools help practitioners construct mental models of the focus of

the discussion in a meeting and to develop an understanding of the pertinent problems. Initial tests have shown that even simple highlighting and overlay mechanisms can help project stakeholders to focus the attention of meeting participants and explain an issue under discussion more effectively. Current software tools may support the sharing and representing of the project information, but do not enable users to relate many kinds and sources of information to each other.

We contend that integrative environments that enable teams to visually and interactively relate project information require a variety of data exchange modes and database architecture components. In recent work, we have begun to address the issue of system architectures for distributed, model-based, integrated systems for AEC/FM (Gorlick and Froese 1999). System architecture addresses issues of how an array of software components distributed across multiple computers can work together to support generation of, visualization of, and interaction with relationships between project information that are necessary for multi-disciplinary decision-making. This paper introduces a typical or reference architecture for such systems. It describes the results from a prototype implementation of this architecture. Finally, it introduces two specific concepts that we believe to be important to the further development of such architectures: multi-modal architectures and transaction-based services.

2. MOTIVATION

Construction project teams must consider a wide variety of information when making project decisions (Figure 1). Much of this information is produced electronically and is visual in nature, yet teams primarily use paper-based views of project information. These views often do not communicate critical relationships between project information or adequately highlight the important and critical information. Consequently, project teams spend far too much time trying to understand and describe project information to one another and are unable to leverage existing information to support decision-making and solve problems. Consider the following schedule review meeting for a major construction project in Southern California we observed:

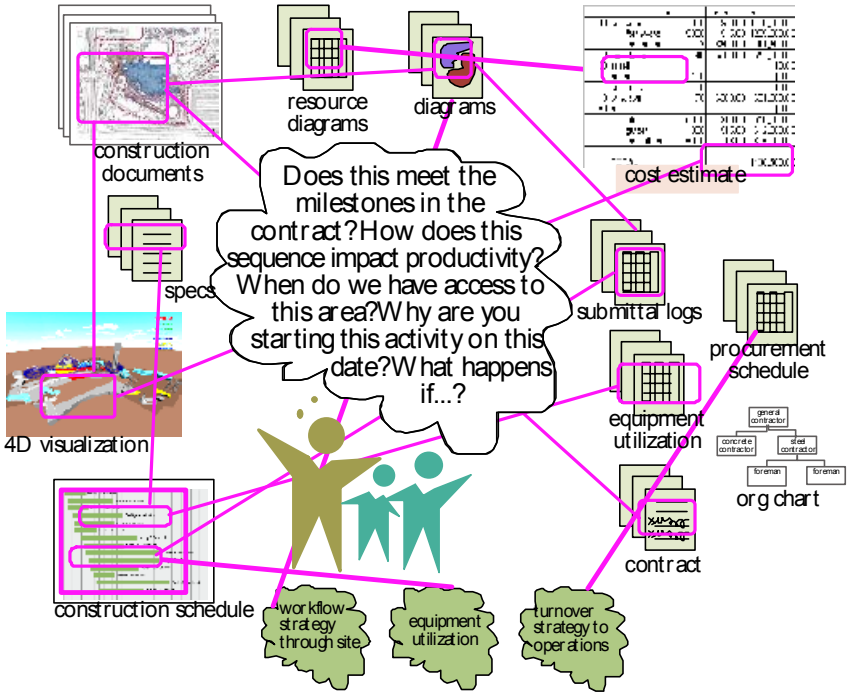

Figure 1. Even electronic documents require users to relate project information for decision-making.

On the walls of the conference room are 2D construction drawings and the project Gantt chart. Each meeting participant has handouts consisting of the schedule, which contains 8,000 activities, and the meeting agenda. Participants have brought other types of documents to the meeting such as 'marked-up' schedules, some contract documents, and construction drawings. The meeting begins with the first agenda item, 'Schedule Comments.' This discussion involves the owner asking questions such as: Does the schedule meet contractual milestones? Do these activities adhere to project specifications? Why are you finishing this facility on this date? What if we change this milestone date? What if the equipment is late? Throughout the meeting, project participants are distracted as they shuffle through the schedule sheets searching for activities or as they scan the walls searching for relevant information, trying to understand the schedule and the issues at hand. Meeting participants come and go. Some leave to get information such as project specifications or to get updated information. In some cases, a document is passed around for participants to review. By the end of the meeting, twenty types of documents have been referred to or used as participants try to describe, understand, explain, and evaluate the schedule. Although several problems are noted, and issues or potential solutions discussed, no problems are resolved during this meeting nor during the successive three meetings.

All of the information used was available online. Did it support the project team's ability to make decisions, and did the team effectively utilize the information they produced? No, since the team was unable to make any decisions and they spent more time trying to describe, explain, and evaluate the information than using the information to support decision-making. Our observations of project meetings show that teams spend most of their time on descriptive, explanative, and evaluative tasks - tasks that support the decision-making process - and approximately only 10% of the time performing predictive tasks or the critical "what-if" tasks that lead to better decisions. More importantly, teams rarely completed these predictive tasks in group settings. When they did, our observations showed that the reliability and accuracy of those tasks was low since team members often commented that the information was out of date or that they were uncertain of potential impacts to project activities and objectives.

Further analysis of such meetings indicates that most of the information that teams need to perform descriptive, explanative, evaluative, and predictive tasks is currently produced by project teams. However, project teams cannot leverage existing project information for the following reasons:

- Focus of information is not shared and project participants are easily distracted and forced to identify relevant information on their own.
- Views are inappropriate for group use, hindering multi-stakeholder participation.
- Views don't visually communicate critical relationships between project information, requiring project participants to manually integrate project information.

To address these issues and improve the effectiveness of using information to support communication and decision making, we are investigating approaches to visual presentation and integration of AEC/FM project information. The following section describes a prototype system developed as part of this research.

3. A PROTOTYPE INTEGRATED SYSTEM

A prototype integrated AEC system was implemented for a project dealing with interactive workspaces and integrated construction systems. This work was carried out by the

authors and other researchers lead by Martin Fischer at Stanford University's Center for Integrated Facility Engineering (CIFE) collaborating with the Stanford Interactive Workspaces Project (Winograd and Hanrahan 2000). A major objective of this project is to explore the potential for using advanced, interactive workspaces (as illustrated in Figure 2—essentially, meeting rooms equipped with numerous interactive computer devices and the accompanying infrastructure to coordinate their use) to support construction team activities (Fischer et al. 2000, Liston et al. 2000). This involved the simultaneous use of several construction-related software tools to analyze the effects of changes to construction schedules made during construction site meetings. This approach requires the ability to share project information among the different software programs. Exploring this construction system integration was the second major objective of project. Although the integrated systems approach is not dependent upon the interactive workspace environment, the workspace technologies research requires a solution for information integration and this research provides an excellent test bed.

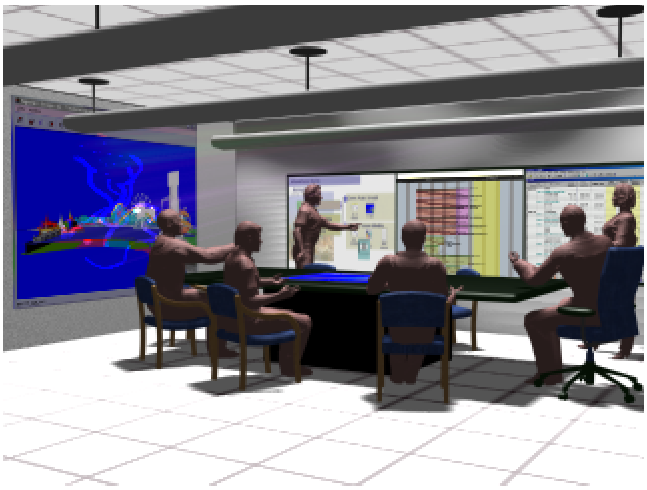

Figure 2. An illustration of the interactive workspace used for the prototype.

The integrated system developed for this project is illustrated in Figure 3. The system included three applications: a custom 4D tool (3D CAD plus time visualization), Primavera Project Planner for scheduling and resource management, and Timberline Precision Estimating for cost analysis. The data exchanged and stored in a shared database is structured according to a common schema loosely based on the IFC Release 2.0 model, but greatly simplified to provide only the data representation required for this prototype. It contained approximately 10 classes for representing building elements (without geometry), work plans, work tasks, resources, and costs.

The system uses a three-tiered architecture (tiered architectures and the components that make up the system are discussed in greater detail in Section 4). The data tier contains components that read and write, which data structured according to the common schema, in both XML files and relational databases. The middle tier contains "local model proxy components" that implement the common schema objects and expose them to application adapter components through a COM interface. The application tier contains the applications themselves and adaptors that map the applications' data to the local model proxy

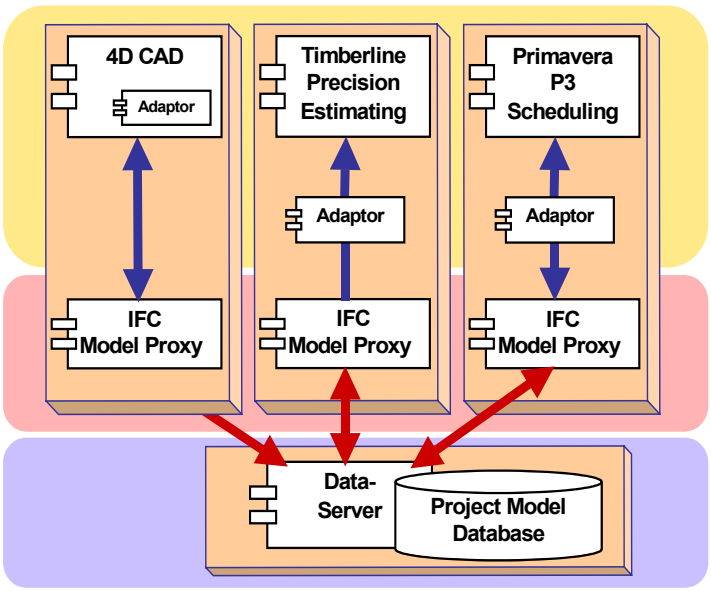

Figure 3: Components of a prototype distributed project management system.

components. The adaptor component for the 4D tool is implemented within the source code of the application, and allows import and export of building objects, scheduling tasks, and relationships from building objects to tasks. The adaptor component for Primavera is implemented as a stand-alone utility that maps schedule task and resource information from the shared project model in and out of Primavera's COM API (application programming interface). The adaptor for Timberline is a stand-alone utility that maps building component, schedule task, and resource information from the shared project model into Timberline cost estimating assemblies, through a Timberline COM API, to create an estimate within the Precision Estimating application.

The prototype was fully implemented and used to conduct a test-case scenario of a hypothetic schedule review meeting based on an actual large construction project in Southern California,. A contractor realizes that the original construction schedule does not provide adequate time for equipment testing after a lagoon area is filled with water. Therefore, the construction operations for the lagoon base will have to be accelerated. During construction, however, the congested site necessitates that the lagoon area be used for material laydown and crane working areas. Thus, the acceleration of the lagoon construction had to be carefully planned and coordinated among numerous work groups.

At the beginning of the scenario, the three construction planning applications (4D tool, scheduling, and estimating) and the common project database were all synchronized to the current construction plan. The analysis of the proposed change began with changes made inside the scheduling application to the number of crews assigned to the lagoon base construction activities. This resulted in a revised and accelerated construction schedule. The revised schedule dates and resource assignments where then exported from the scheduling tool to the project database. Next, the new project information was read out of the project database and used to create an updated cost estimate, which could be compared with the initial estimate (although the building components and activities were the same, the addition of extra crews had been accompanied by a slight decrease in productivity, so the overall cost was somewhat higher). Finally, the new schedule information was read from the project database into the 4D-planning tool, so that the new construction sequence could be carefully analyzed for adverse work interactions.

The results of this prototype showed that the basic approach worked well, but that many specific system design and implementation issues must be addressed in detail before the system could approach a practical solution. To further pursue this work, we have defined a reference architecture model for distributed systems.

4. A REFERENCE ARCHITECTURE FOR DISTRIBUTED, MODEL-BASED INTEGRATED SYSTEMS

4.1. A Tiered Reference Architecture

As a foundation for discussions of system architecture issues, Figure 4 illustrates a set of software components for a distributed, model-based, integrated system for AEC/FM. These components describe a typical—or *reference*—architecture, yet a wide range of variations is both possible and required, as discussed later in this paper. This particular architecture is our interpretation of emerging standard practice for distributed systems as we believe that it should be applied within the AEC/FM industry (i.e., it is intended to reflect a “best practice” architecture rather than an innovative one). The architecture is organized into three logical tiers: the application or presentation tier contains application programs and related user-centric components; the business objects or middle tier has two main roles—bringing the data and services of the distributed system to the local applications and implementing business logic processes; and the data tier handles the persistence of project model data. Figure 4

shows these logical layers residing on two or more physical computer systems: a user's workstation and one or more central servers. This is but one example of numerous possible configurations (For example, all components could be implemented on a single workstation; web-based applications could be implemented on central servers with only a web browser on a user's workstation; or a local model proxy could be implemented on a local area network server for shared use throughout a small workgroup—alternative configurations are discussed further in Section 5).

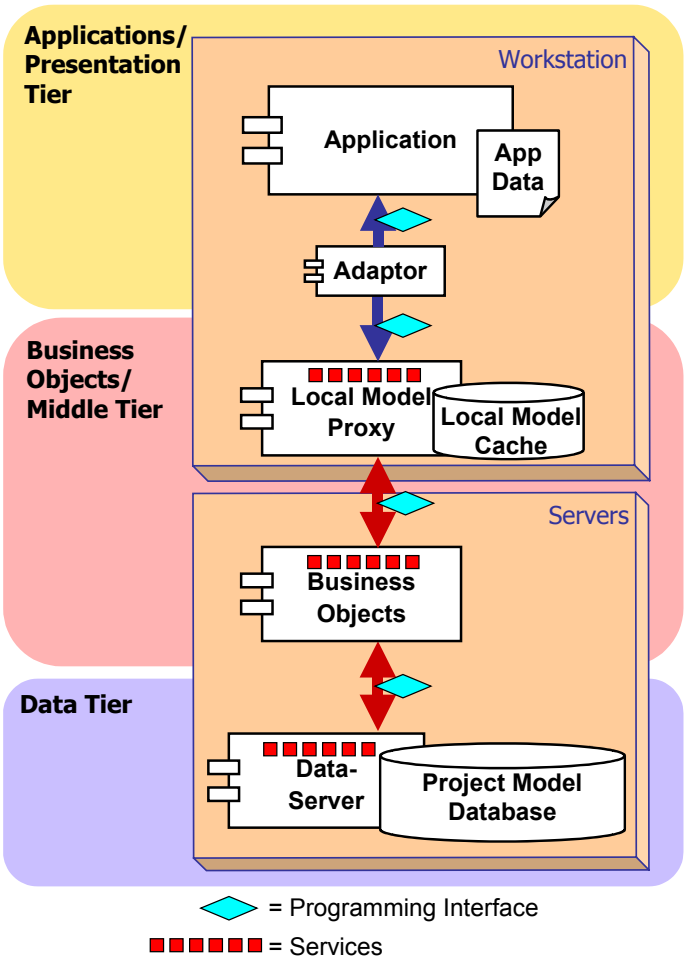

Figure 4. A reference architecture for a distributed, model-based, integrated system

4.2. System Components

Typical components are as follows:

- **Applications:** Users interact with the system through application programs: typically legacy applications for AEC/FM such as CAD, estimating, or scheduling applications, etc. These applications typically maintain their own data sets in addition to the information shared through the integrated system. Additionally, they can be new application classes specifically designed to work with the integrated system, or lightweight client interfaces to server-based application (such as browser interfaces to AEC services offered by Internet portal sites).
- **Adaptors:** The data and operations contained within any application must be mapped to the shared data and operations available within the integrated system. This mapping is carried out by an application-specific adaptor software component. Depending upon the nature of the application, the adaptor may be encoded within the application itself, may be an add-on or macro within the application, or may be a separate program that interacts with the application through an application programming interface (API) or operates on the application's data files.
- **Local Model Proxies:** The functionality or services available to participating applications through the distributed system are typically implemented on remote servers and made available to local applications through software components on users' workstations. This can be thought of as a local copy or proxy for the shared project model (see the description of "Webtop servers" in Lottaz et al, 2000). The local model proxy component, then, exposes the distributed information services to client applications, and handles the work of communicating and coordinating the local data with the distributed servers.
- **Business Object Components:** An important concept of distributed systems technology is the role of *business objects*: centrally deployed software components that implement fundamental logic or processing common to all of the applications that use the data. For example, this logic can handle a great deal of the data validation checking and the

propagation of the effects of changes to one data object to other related data objects. We know of no AEC-specific, currently available systems that make business object services available to outside applications, and we believe that the vast majority of AEC/FM business logic will continue to reside within applications for the near future. However, we are aware of systems currently under development that could provide AEC business object services, and generic e-commerce-support products (such as Microsoft's Biztalk server[1]) or standards from other industries (such as the RossettaNet Networked Application Protocols[2]), can be described as business object components. As business object approaches emerge, they will fit well within the reference architecture. We have shown them here as a distinct component that resides on servers, in-between the data components and the local model proxies. However, business objects may well be implemented within any of the other components by adding additional logic processing to those component's objects.

- **Data Components:** Various data components are responsible for maintaining the shared project model. A typical configuration, as in Figure 4, involves a central database and data server component that interacts with model proxy components across an Internet/Intranet connection. Alternative configurations include distributed data sources, where the central database is replaced with numerous, distributed, heterogeneous data sources; and approaches that rely on file exchange between model proxies with no central database.

4.3. System Features

Some key features of these components include the following:

- **Services:** The distributed system's functionality can be described as various services that are offered to the participating applications. Examples include basic access to shared data, data set management (e.g., checking data sets in and out of central repositories, object versioning support, data security, etc. Each service is implemented in one or more of the components, and each component implement one or more services.
- **Programming Interfaces:** An important issue in the system design is the specification of the programming interfaces or protocols used between components. Some of the critical interface points for distributed, model-based systems are indicated in Figure 4 with diamond symbols. Numerous technologies exist to provide partial solutions, such as COM, CORBA, XML, SDAI, etc. These generic technologies, where used, must have AEC/FM-specific solutions implemented on top of them.
- **Schema:** Each of the software components shown in Figure 4, as well as each of the interfaces between components, requires an underlying schema or data structures for representing project information. Generally, the middleware and data layer components use industry standard schema, such as the IFCs, while the application layer uses application-specific schema. Various components, and particularly the Adaptor, must map information between different, but related, schemas.

4.4. Distinctive Characteristics

None of the requirements or characteristics for distributed systems may be unique to the AEC/FM industry, but a number of characteristics differ from "typical" distributed systems, and this combination of distinctive characteristics may well be unique to the AEC/FM industry. The following are a few of these characteristics:

- **Complex Data:** AEC/FM projects are complex, and so too are the data structures (schema) required to represent these projects. This is particularly true for the technical

[1] http://www.microsoft.com/biztalk/
[2] http://www.rosettanet.org/

project information (e.g., CAD and other building product data), but also applies to AEC/FM project management data (e.g., schedules, estimates, quality plans, etc.) and commerce data (e.g., purchasing transactions, contracts, etc).

- **Large Data Sets:** Project data sets such as building product models are very large. This fact alone means that an approach which involves applications accessing all required data from remote repositories on demand (i.e., as in typical Web applications) is likely to be infeasible. Rather, a more advanced combination of coordinated centralized and decentralized data sources is required.
- **Reliance on Legacy Applications:** The previous two characteristics and other factors are likely to lead to an ongoing reliance on current, desktop applications (which we'll refer to legacy applications), rather than a rapid adoption of new, web-based tools for most functions.
- **Emerging Data Standards:** Although the nature of the Internet revolution was not commonly anticipated even five or ten years ago, AEC/FM researchers and developers have maintained a long-standing recognition of the need for data exchange standards to support system interoperability. A number of significant efforts have been under development for some time (e.g., STEP, IFC's, aecXML). To date, these standards have had little impact on either traditional or Internet-based integration, but they will become increasingly important as the scope of Internet-driven interoperability increases.
- **Organizational Challenges:** AEC/FM projects involve large numbers of small companies collaborating for short project lifecycles. For distributed collaborative systems, this places a very high premium on minimizing systems' "overhead" and in developing industry-wide solutions.

5. MULTI-MODAL ARCHITECTURES

As discussed, many variations and alternatives to the reference architecture are possible for the overall configuration of components. Not only are they possible—we contend that it is necessary for a distributed AEC/FM system to support a wide variety of data exchange modes. This section presents a series of typical AEC/FM data exchange scenarios (use cases) that we have developed to illustrate the need for a multi-modal solution. For each mode, we discuss briefly the variations made to the reference architecture to support this mode.

5.1. File-Based Data Exchange

Use Case 1: A building product model is exported as an IFC file from the CAD system in which it was created.

Use Case 2: An IFC file representing a building product model is read as input to an energy analysis package or to a 4D construction-planning tool.

Data Exchange Mode: Although the notion of distributed integrated systems nominally involves "on-line" access to a shared project database, the most common mechanisms for importing and exporting IFC data today involves file-based data exchange. This mode can be fully supported by treating the data source component as a file, the contents of which are exposed to an application adaptor through a model proxy component (see Figure 5). While it's important to support this mode because of its prevalence among today's limited set of IFC-enabled tools, file-based data exchange has many shortcomings compared to database approaches.

This mode was implemented in the prototype system described above since each of the applications, working through the model proxy component, could import and export data sets as XML files. This work well for many purposes, but it provides no mechanisms for the output of one application to be "merged into" the output from another application. This

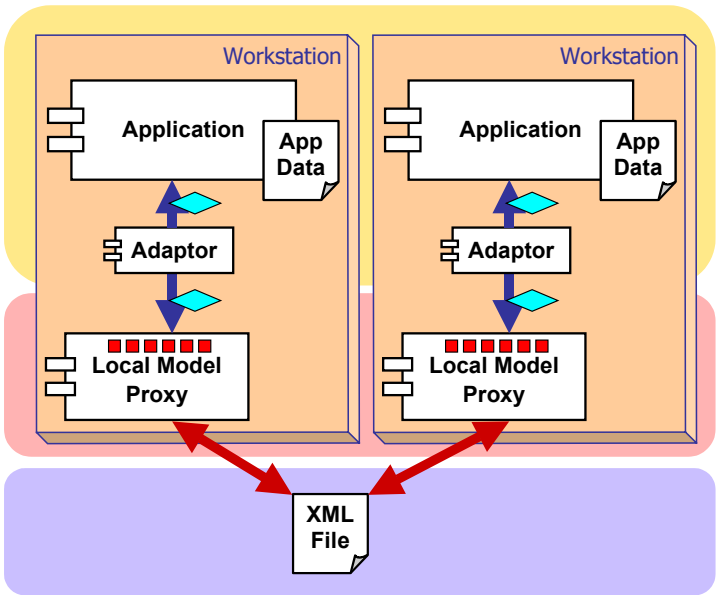

Figure 5. Architecture for file-based data exchange

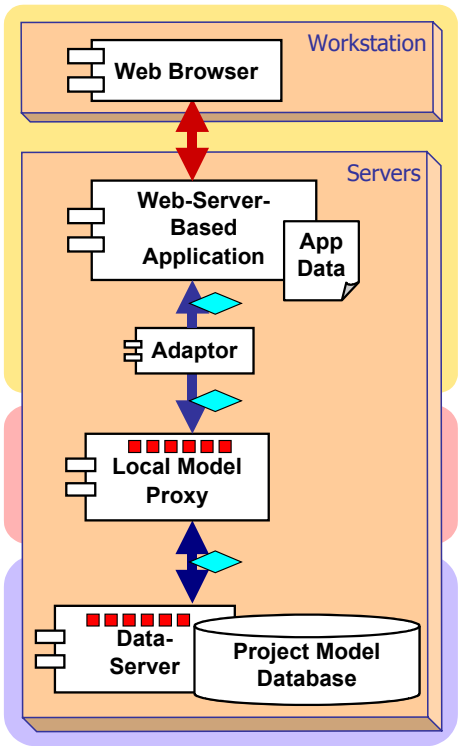

Figure 6. Architecture for server-based applications

was a severe limitation since the cost estimating required data from both the 4D viewer and the schedule applications.

5.2. Coarse-Grained Data Exchange

Use Case 1: A complete building product model is exported from the CAD system in which it was created, across an Internet connection, into a shared project database, where it is available for other use by other applications.

Data Exchange Mode: Applications access data in "large chunks", or collections of many objects. These collections may correspond to complete models, user-defined objects collections, or the results of queries or stores procedures. This mode is associated with relatively large data sets and/or slow connections.

The prototype implemented this mode since each application, working again through the model proxy component, could import and export its entire data set to a central relational database. This approach offered very good functionality, but our implementation (an initial "rapid prototype") exhibited very poor performance for the scenario (the scenario involved approximately 2,000 building components and a total of approximately 9,000 objects). None of the other exchange modes described below were implemented in the prototype.

5.3. Fine-Grained Data Exchange

Use Case 1: A change-order management tool reviews the status of change orders stored within a shared project database.

Use Case 2: When an object is selected in a CAD tool, it can show the object's cost data drawn from a remote project database.

Use Case 3: A "project explorer" interface allows a construction manager to browse and selectively edit all information stored in a shared, model-based project database.

Data Exchange Mode: Applications access data in "small chunks" (individual objects or small collections of objects). This mode is suitable for relatively small datasets and/or fast

connections. It supports both traditional applications and "thin-clients", or applications that provide little more than user interface functionality, as in the third use case above.

5.4. Off-Line/Message-Based Data Exchange

Use Case 1: An engineer sends a set of design changes to the architect as an e-mail (or similar type of) message.

Use Case 2: An inspector records inspection results into a tool on a palm-based computer, and the results are incorporated into the project database once the device is docked on the inspector's desktop.

Data Exchange Mode: Applications send datasets as messages to be delivered to a recipient at some future time. The connection from the local model proxy to the data layer is through a queued message system.

5.5. Server-Based Applications

Use Case 1: A construction manager uses a web browser to review the status of a project's subcontract packages on an Internet portal site's (i.e., a construction ".com" company's) project management system, which maintains a model-based project database.

Data Exchange Mode: Although this mode provides a marked departure from traditional desktop-based applications, it still fits well within the general distributed-system architecture. The primary difference is that all of the components shown in the typical architecture reside on servers, with the exception of the user interface elements of the application, which are separated out and delivered to the user's computer through Internet HTTP capabilities (see Figure 6).

5.6. Application-To-Application Data Exchange

Use Case 1: An estimating system interacts directly with a CAD system, without going through a project database.

Data Exchange Mode: This variation, which can be applied to all of the above alternatives, involves applications communicating directly with one another rather than with a common project database. This mode fits within the reference architecture by simply allowing individual applications to be treated as data sources as well as data clients, and implementing the data source API's against the applications.

5.7. Multiple Data Repositories

Use Case 1: A "project management desktop" applications, which gives users access to a wide array of project information, maintains a building's 3D geometry in a local file, accesses cost data across a local area network from a repository on a corporate intranet, and gets information about the project status directly from a remote project repository. All of the data sources are synchronized as needed.

Data Exchange Mode: data repositories can be arranged in "chains", where an application acts as a client of a "satellite" repository which, in turn, acts as a client of a centralized repository. Data management and transaction services (such as loading, locking, caching, and synchronization of data sets) operate in a similar manner between each link in the chain.

6. TRANSACTION-BASED SERVICES

Currently, data exchange among computer applications within the construction industry relies almost exclusively on document exchange, where datasets are exported as some type of document file from one application and imported into another. In contrast, our research is focused on model-based data exchange, wherein collaborative applications read and write

objects to a shared project-model repository. Although model-based integration continues to be the cornerstone of our approach to data exchange among construction applications, it exhibits many shortcomings (Eastman 1999). For example, numerous model-management issues remain unresolved (such as extensibility of models, supporting different application models and perspectives, version management, etc.), model-based exchange has little to offer at the level of integrating user interaction issues, and there are many situations where "shared models" are simply not desired. A simple example is a bank transfer, where a message requesting the transfer is sent from one account to another—there is no sharing of complete account information between the two accounts. This approach of collaboration through transactions among applications addresses many shortcomings of model-based data exchange.

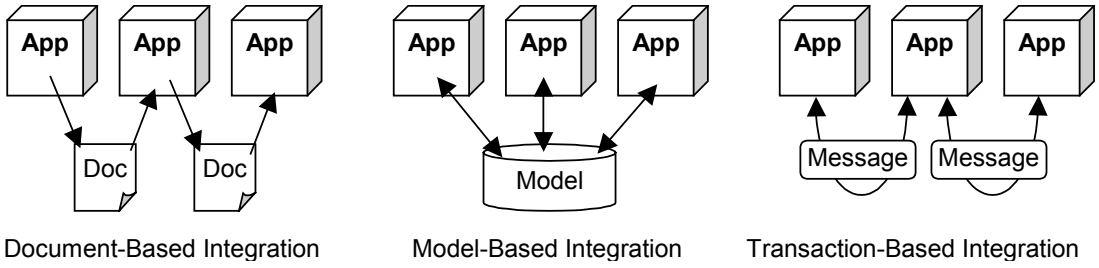

Figure 3: Alternative integration paradigms.

Transaction-based collaboration involves the broadcasting of messages from one application to other applications. Messages include some type of instruction or some notification that an event has occurred. Messages may include datasets and may result in a response from receiving applications. Examples of transaction messages include notification that data has changed, exchange of specific datasets, requests to select and highlight objects in target application, requests to switch to certain views or perspectives, etc.

Transaction-based integration also plays an important role with respect to emerging trends in the structure of construction information technology. Traditionally, individual computer applications encompassed three main responsibilities or "layers": representation and management of domain data, application functionality and logic, and user interface/information presentation. Model-based integrated systems move much of the responsibility for the data layer away from individual applications and into distributed model-based project repository mechanisms, yet this provides little support for integration of the logic or user-interface layers. Transaction-based integration has no such limitation; transactions can relate to data, logic or user interface issues. Thus, transaction-based integration provides support for the interactive workspace issues (Fischer et al. 2000, Liston et al. 2000) that are missing from model-based approaches. Moreover, current trends in information technology systems indicate greater separation of the application logic and user interface layers, as in the systems offered by emerging e-commerce providers for the construction industry, where data and application logic components reside on service-providers' central systems while user interaction occurs through web-based interfaces. As application logic and user interface layers are split apart, the need for standard mechanisms of communicating both data and messages between components increases: a need that is addressed by transaction-based integration.

Transaction-based integration doesn't replace model-based integration, but could provide an important additional service offered by a distributed system architecture. Like model-based exchange, however, its adoption requires formalization of the role that it should play in systems integration, industry-level standardization of transaction messages, and various other design issues. We intend to investigate these issues further in on-going work.

7. CONCLUSIONS

This paper discussed architectures for distributed, model-based, integrated systems for AEC/FM. It described the results of a prototype implementation, presented an overall reference architecture, and introduced the role of multi-modal architectures and transaction-based services.

Acknowledgements: We gratefully acknowledge support for this work from the Natural Sciences and Engineering Research Council of Canada, the Center for Facility Engineering and UPS Fellowship at Stanford University, and Timberline Software Corporation.

REFERENCES

Eastman, C.M., "Information Exchange Architectures For Building Models, Information Exchange Architectures", Durability of Building Materials and Components 8, Vancouver, May 1999. pp. 2139-2156.

Fischer, M., Liston, K., Kunz, J. (2000) "Requirements and Benefits of Interactive Workspaces in Construction," submitted to the 8th International Conference on Computing in Civil and Building Engineering, Stanford, USA.

Froese, T., et al. (1999) "Industry Foundation Classes For Project Management—A Trial Implementation", Electronic J. of Information Technology in Construction, Vol.4, 1999, pp.17-36. Available online at http://itcon.org/1999/2/

Froese, T., Rankin, J.,and Yu, K. (1997). "Project Management Application Models And Computer-Assisted Construction Planning In Total Project Systems," Int. J. of Construction Information Technology, Vol. 5, No. 1, Summer 1997, pp.39-62.

Gorlick, A. L., and Froese, T. (1999). "A Prototype Distributed CIC System Based On IAI Standards", Durability Of Building Materials And Components 8. Vancouver, May 1999. Vol. 4, pp. 2171-2179.

Liston, K., Kunz, J., and Fischer, M., (2000). "Advanced Human-Computer Interfaces for Construction Planning," submitted to the 8th International Conference on Computing in Civil and Building Engineering, Stanford, USA.

Lottaz,C., Stouffs, R. and Smith, I. (2000). "Increasing Understanding During Collaboration Through Advanced Representations", Electronic J. of Information Technology in Construction, Vol.5, pp.1-24. Available online at http://www.itcon.org/2000/1/

Schwegler, B., M. Fischer, and K. Liston. New Information Technology Tools Enable Productivity Improvements.in North American Steel Construction Conference, AISC,. 2000. Las Vegas.

Winograd, T. and Hanrahan, P., (2000). Stanford Interactive Workspaces Project. Web site available at http://www.graphics.stanford.edu/projects/iwork/ Accessed Feb. 2000.