INTEGRATING PROJECT MANAGEMENT APPLICATIONS AS WEB SERVICES

Jinxing Cheng[1], Kincho H. Law[2] and Bimal Kumar[3]

ABSTRACT

Many related software applications can be employed at various stages of a project, at different locations and for disparate purposes. Integrating these tools can help extend the capabilities of individual software applications. This paper addresses some of the issues related to the integration of distributed software applications as Web services. Specifically, information modeling for project management applications and communication mechanisms are examined. Process Specification Language (PSL) is employed as the information modeling language. The implementation efforts required towards the development of an integrated service architecture are discussed. A variety of project management tools, including Primavera Project Planner™, Microsoft Project™, Vite SimVison™ and Microsoft Excel™, are employed as a demonstration of the distributed integration infrastructure.

KEYWORDS

Distributed Applications, Process Specification Language, Web Services, Integration

[1] *PhD Student, Civil and Environmental Engineering Department, Stanford University, Stanford, CA 94305-4020, email: cjx@stanford.edu*

[2] *Professor, Civil and Environmental Engineering Department, Stanford University, Stanford, CA 94305-4020, email: law@cive.stanford.edu*

[3] *Professor, School of the Built and Natural Environment, Glasgow Caledonian University, UK, email: b.kumar@gcal.ac.uk*

1. INTRODUCTION

Many related software applications can be employed at various stages of a project, at different locations and for disparate purposes. Integrating these tools can help extend the capabilities of individual software applications. This paper addresses the issues of how to integrate distributed software applications as Web services. Specifically, information modeling for project management applications and communication mechanisms are examined.

The integration of distributed applications is often complicated by the diverse data structures and formats employed by the applications. To facilitate the exchange of information between applications, many efforts have been made with collaboration from industry, software vendors, academia and standard organizations to define data exchange standards, such as STEP (ISO 1994), IFC (IAI 1997), ifcXML (Liebich 2001), aecXML (IAI 2002) and others, over the last couple of decades. In this work, we use Process Specification Language (PSL) as an information modeling language. PSL was initiated by the National Institute of Standards and Technology (NIST) and is emerging as a standard exchange language for process information in the manufacturing industry. While most current data standards deal primarily with product information, PSL focuses on process information, which is essential for project management applications. Furthermore, based on first order logic, PSL can potentially support various reasoning mechanisms beyond data exchange, such as checking inconsistencies of project information from different sources (Cheng et al. 2002a).

With the proliferation of the Internet, companies are increasingly leveraging the Internet to achieve competitive advantage. For example, an interactive Web site was developed and used to disseminate bid packages to contractors and material suppliers at different locations (Runser et al. 2002). Construction projects are often performed and managed in a geographically distributed fashion, where a company's headquarter, the regional offices and the construction sites are located in different cities, states and countries. Each office may run its own in-house applications. Integrating software applications as Web services can potentially remove the barriers of geographic boundaries and expedite project delivery. An integration framework would require not only data integration but also network communication in order to achieve interoperability. In this paper, we describe a prototype infrastructure to integrate project management applications as Web services.

This paper is organized as follows. In Section 2 we discuss the applicability of PSL as an information interchange standard for project management applications. In Section 3 we describe a prototype of a distributed integration infrastructure and investigate the mechanism of how to provide interoperability among distributed project management tools. Section 4 provides a sample demonstration of the prototype system, and Section 5 discusses related works and future extensions.

2. PROCESS SPECIFICATION LANGUAGE

The motivation for the development of the Process Specification Language (PSL) is twofold. First, few existing standards focus on exchanging process information. Second, most current standards lack a formal logic to define relationships and constraints. PSL is based on Knowledge Interchange Format (KIF) (Genesereth and Fikes 1992), which can be used to define terms, relationships and constraints.

The overall organization of the PSL ontology (Schlenoff et al. 2000) includes a set of core theories, the PSL core, the PSL outer core and PSL Extensions.

- PSL core theories are used to define the terms in PSL, so that precise interpretation of the terms can be obtained.
- The PSL core is designed to describe fundamental concepts of manufacturing processes and includes four basic classes: Object, Activity, Activity-Occurrence and Timepoint. The expressions *(beginof LayFoundation)* and *(activity_occurrence LayFoundation)* are example function and relation, respectively, in the PSL core.
- The PSL outer core consists of a small set of generic extensions, which are Subactivity Extension, Activity-Occurrence Extension and States Extension. For example, the expression *(subactivity a1 a2)* defines a relation over the activities *a1* and *a2,* in which *a2* can be decomposed into *a1* and other activities.
- PSL extensions provide the basis for process modeling and information exchange among different applications within a specific domain. Currently, standard PSL extensions include various ontology modules, such as generic activities, ordering relations and schedules. For example, the expression *(before-start occ1 occ2 a3)* specifies the relationship among *occ1*, *occ2* and *a3*, in which both *occ1* and *occ2* are subactivity occurrences of the activity *a3*, while the beginning timepoint of *occ1* is earlier than the beginning timepoint of *occ2*.

We have investigated the PSL ontology and compared it with concepts in project management applications (Cheng et al. 2002a). As shown in Figure 1, a typical project includes not only scheduling information but also cost, resource, organization and other information. For example, MS Project™ and Primavera Project Planner™ (P3) include detailed scheduling information and some rudimentary resource and cost information. Vite SimVision™, a project and organization modeling system, provides detailed project organization information but rudimentary cost and scheduling information.

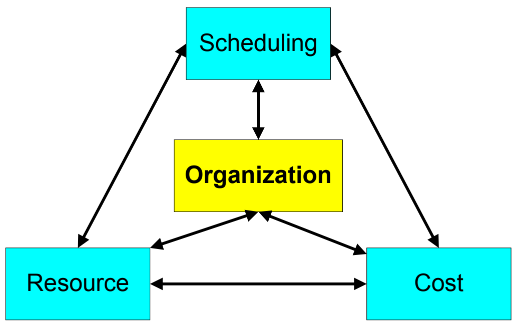


Figure 1: Information in a Construction Project

The current PSL ontology is sufficient for modeling scheduling information. For example, the function *endof* specifies the finish date of an activity occurrence, while the relations *before-start* and *before-start-delay* define the "Finish to Start" dependency relationship between two activity occurrences. The following PSL expressions indicate that, for the Arnold's House Project (ARHO), the activity occurrence *FrameRoof* starts on December 6, 2002 after *FrameHouse* is finished, and it lasts 10 days.

(beginof FrameRoof 12/06/2000)
(durationof FrameRoof 10)
(after-start FrameHouse FrameRoof ARHO)

The current PSL ontology also provides limited capabilities for modeling resource information. For example, the lexicon *resource* identifies an object required by an activity and the lexicon *demand* is used to specify that an activity requires a resource with nonzero quantity. For example, the concept that *InstallDrywall* requires 10 pieces of *Drywall* can be expressed as:

(resource Drywall)
(demand InstallDrywall Drywall 10)

The ontology about cost and organization information has been proposed and is currently under development.

PSL has been successfully used to exchange process information among manufacturing applications. For instance, in a pilot implementation at NIST, manufacturing process information was successfully exchanged using PSL information between the IDEF3-based ProCAP and the C++ based ILOG Scheduler (Schlenoff et al. 1999). To exchange project information among different applications, we first need to develop wrappers for each application (Figure 2). The PSL wrappers are built to retrieve information from applications, to convert the information into PSL formats, to parse PSL files and to transfer the information to applications.

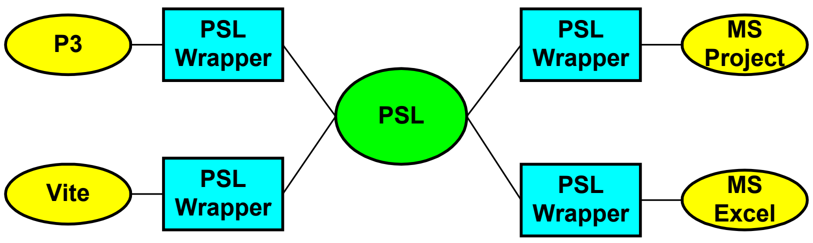


Figure 2: PSL in the Information Exchange

There are three basic steps involved in the translation process between the data files in PSL format and the applications. The first step is to retrieve the project information from an application and to update the project model. Semantic mapping is then performed to translate between the formal PSL ontology and the concepts in various project management tools. Finally, the project data is syntactically translated between PSL files and the applications.

Wrappers have been developed for data exchange among various project management applications. For example, the Primavera Automation Engine (RA) is employed to develop a PSL wrapper for Primavera Project Planner^{TM} (P3). The RA is a set of object-oriented, OLE 2.0-based API, which allows object-oriented programming access to the P3 scheduling engine. For Microsoft Project^{TM} and Microsoft Excel^{TM}, VBA (Visual Basic for Application) is utilized. SvEngine, a COM (Component Object Model) automation object, is used to access a simulation engine compatible with Vite SimVision^{TM}. In addition, a translator has been built to translate the project data between PSL files and a relational database.

3. DISTRIBUTED INTEGRATION INFRASTRUCTURE

Different architectures have also been proposed to achieve software integration, such as localized integration, client-server integration and distributed integration. Localized integration involves integrating various software tools on one machine, for example a desktop PC. In a client-server environment, software integration is often accomplished using a project repository, which is either a neutral file or a database, residing on a central server, to which all applications communicate to exchange information. In a distributed environment, applications are resided on different computers and are accessed over private or public, local or wide area network.

Web services are distributed services which consist of independent application components published on the Web (Roy and Ramanujan 2001). These application components are published in a way that they can be used by other Web applications. Examples of Web services include a financial market service that provides stock market information, a weather forecasting service that can be used for construction planning, and a price quoting service that can be utilized to optimize production plans. Conceptually, a typical Web service architecture consists of three entities: service providers, services brokers and service requesters (Roy and Ramanujan 2001).

- Service providers develop Web services, register them with service brokers, and publish them to the Web.
- Service brokers act as bridges between service providers and service requesters; they also maintain detailed lists of published Web services.
- Service requesters search the brokers' lists, find the required services, and send requests to the corresponding service providers.

To integrate Web services, a data standard needs to be employed, so that results can be reused by other applications. Network communication issues, such as asynchronous messaging, also need to be addressed (Bosworth 2001). Furthermore, mechanisms for invoking and terminating applications over the network need to be provided (Liu et al. 2002b).

The goal of a distributed integration infrastructure is to link application tools, to act collaboratively on a project, and to allow access to the results using a client's device, such as a PDA, a Web browser or a desktop computer, irrespective of time and space. We have prototyped a distributed integration infrastructure using PSL as the information interchange standard among different project management tools. As shown in Figure 3, a communication server and a database system are used to serve as the backbone of the system. The communication server is responsible for listening requests from clients, including various applications and client devices. When the server receives a request, it broadcasts the request to different communication agents. The communication agents then pick up the request and process it. In addition, an active mediator is built to act as an information broker between the client devices and the information sources (Liu et al. 2002a). For example, the user can send a request from a Web browser (in an XML format). The mediator then sends the request to the communication server and the database. The results retrieved from the database are then returned to the mediator, which filters and transforms the results suitable for display on the client device.

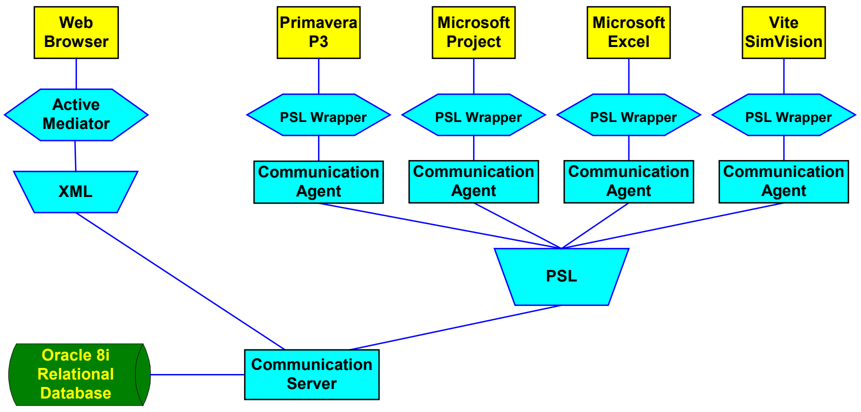


Figure 3: A Distributed Integration Infrastructure

The network communication mechanism for the distributed Web service infrastructure is illustrated in Figure 4. Java socket communication is used as the protocol between the communication server and agents. A communication agent includes an event listener, an event dispatcher, and a data mapper. The messages in the system include control messages and data messages. Control messages, such as invoking and termination requests, are typically small in size. Data messages, such as the project scheduling information and organization information, however, are usually bigger in size. The event listener receives control messages, while the event dispatcher sends out control messages. The data mapper is responsible for sending and receiving data messages.

Figure 5 shows a Java code segment of an event listener. The listener first creates two data streams: one input stream and one output stream. It then creates a Socket on a specific port. Finally, it keeps listening on the port to see if there are messages from the server.

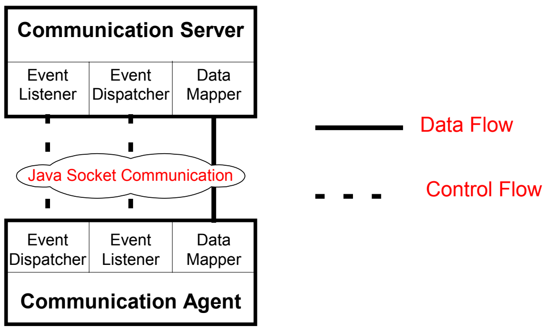


Figure 4: A Network Communication Framework

```
Public class ClientListener{
  protected DataInputStream i;  protected DataOutputStream o;
  public static void main (String args[]) throws IOException {
    Socket s = new Socket (args[0], Integer.parseInt (args[1]));
    ClientListener client = new ClientListener(" " + args[0] + ":"
+ args[1], s.getInputStream (), s.getOutputStream ());
    client.waitForEvent();
    s.close(); }
  public void waitForEvent () {
    try { String line = i.readUTF ();}
    ……}
  ......
}
```

Figure 5: The Code Segment of an Event Listener

4. DEMONSTRATION

We use the Arnold's House project from the tutorial example of Vite SimVision™ to demonstrate our prototype system. The goal of the project is to build a residential house on time and within budget. Vite SimVision™ is used to model the planned work process and identify major risks, such as task backlogs. Primavera Project Planner™ (P3) or Microsoft Project™ is used to schedule the project, while Microsoft Excel™ is used to display summary information. Figure 6 shows the initial Arnold's House project in Vite SimVision™, which includes a traditional CPM diagram and project personnel who are responsible for the activities. Figure 7 presents the schedule regenerated in Primavera Project Planner™.

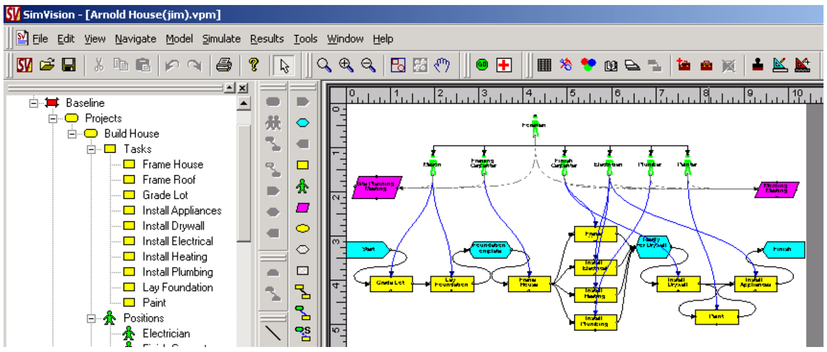
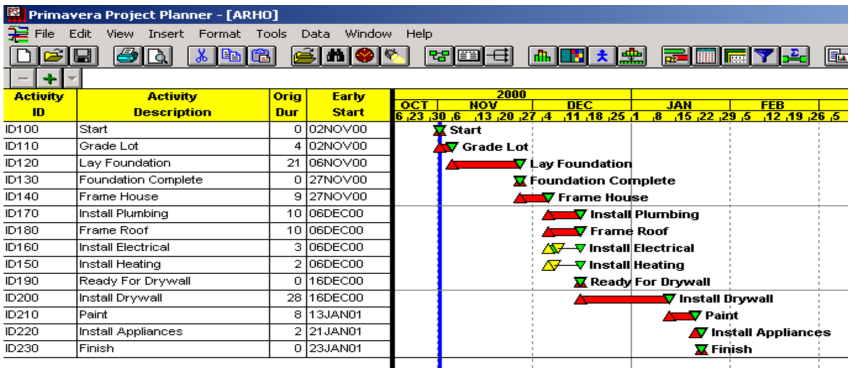


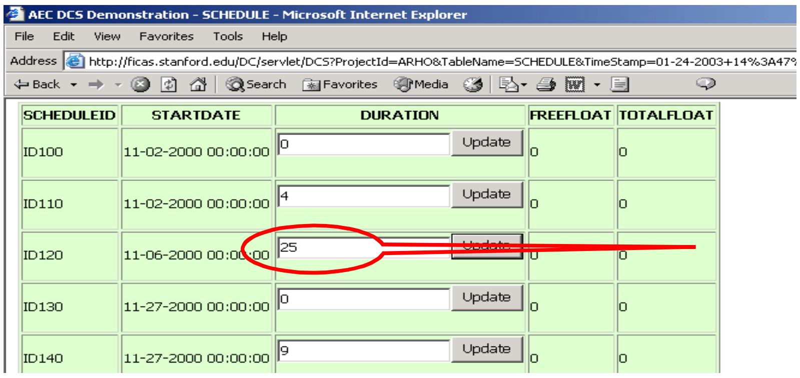
Figure 6: The Arnold's House Project In Vite


Primavera Project Planner - [ARHO]
File Edit View Insert Format Tools Data Window Help

Activity ID	Activity Description	Orig Dur	Early Start
ID100	Start	0	02NOV00
ID110	Grade Lot	4	02NOV00
ID120	Lay Foundation	21	06NOV00
ID130	Foundation Complete	0	27NOV00
ID140	Frame House	9	27NOV00
ID170	Install Plumbing	10	06DEC00
ID180	Frame Roof	10	06DEC00
ID160	Install Electrical	3	06DEC00
ID150	Install Heating	2	06DEC00
ID190	Ready For Drywall	0	16DEC00
ID200	Install Drywall	28	16DEC00
ID210	Paint	8	13JAN01
ID220	Install Appliances	2	21JAN01
ID230	Finish	0	23JAN01

Figure 7: Project Schedule in Primavera P3

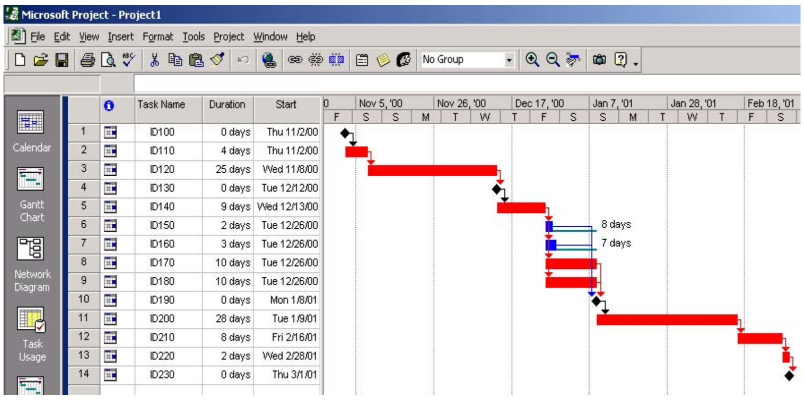
Using the distributed integration infrastructure, we can also view and update the project on a Web browser, as shown in Figure 8. For example, we can change the duration of the activity ID120 ("Lay Foundation") from the original 21 days to 25 days. The communication server then broadcasts the change to various applications, such as Primavera Project Planner™, Microsoft Project™ and Vite SimVision™. Figure 9 shows the rescheduled results in Microsoft Project™. The updated schedule can be sent to Vite SimVision™ for work process simulation. The re-simulated results can be retrieved and displayed in Microsoft Excel™. For example, Figure 10 shows the backlog information of different project personnel as displayed in Microsoft Excel™. In particular, we can examine the backlogs of the mason who is responsible for the delayed activity ID120 ("Lay Foundation"). The results of the updated project information can be retrieved using a Web browser where the delayed activity as well as the affected activities is highlighted, as shown in Figure 11. It should be noted here that, throughout the simulation, the software tools are resided on different computers with different URLs. Furthermore, the changes and updated project information are stored in the relational database for persistence storage and version control.



AEC DCS Demonstration - SCHEDULE - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://ficas.stanford.edu/DC/servlet/DCS?ProjectId=ARHO&TableName=SCHEDULE&TimeStamp=01-24-2003+14%3A47
Back Search Favorites Media

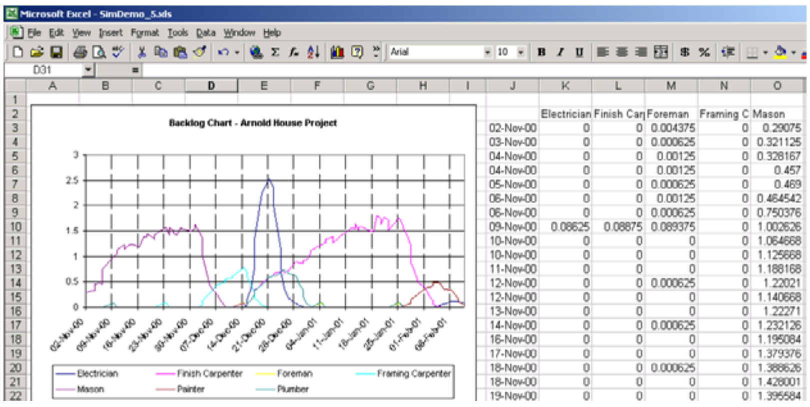
SCHEDULEID	STARTDATE	DURATION		FREEFLOAT	TOTALFLOAT
ID100	11-02-2000 00:00:00	0	Update	0	0
ID110	11-02-2000 00:00:00	4	Update	0	0
ID120	11-06-2000 00:00:00	25	Update	0	0
ID130	11-27-2000 00:00:00	0	Update	0	0
ID140	11-27-2000 00:00:00	9	Update	0	0

Figure 8: Change Activity Duration on a Web Browser



	Task Name	Duration	Start
1	ID100	0 days	Thu 11/2/00
2	ID110	4 days	Thu 11/2/00
3	ID120	25 days	Wed 11/8/00
4	ID130	0 days	Tue 12/12/00
5	ID140	9 days	Wed 12/13/00
6	ID150	2 days	Tue 12/26/00
7	ID160	3 days	Tue 12/26/00
8	ID170	10 days	Tue 12/26/00
9	ID180	10 days	Tue 12/26/00
10	ID190	0 days	Mon 1/8/01
11	ID200	28 days	Tue 1/9/01
12	ID210	8 days	Fri 2/16/01
13	ID220	2 days	Wed 2/28/01
14	ID230	0 days	Thu 3/1/01

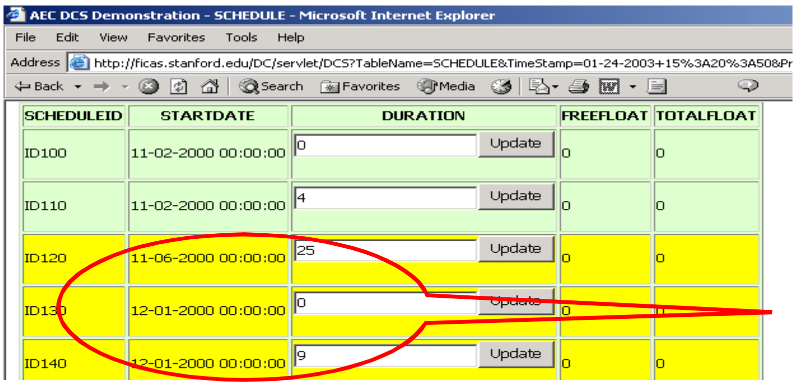
Figure 9: Updated Project Schedule in Microsoft Project



Microsoft Excel - SimDemo_5.xls
File Edit View Insert Format Tools Data Window Help
Backlog Chart - Arnold House Project
Electrician
Finish Carpenter
Foreman
Framing Carpenter
Mason
Painter
Plumber

	Electrician	Finish Car	Foreman	Framing C	Mason
02-Nov-00	0	0	0.004375	0	0.29075
03-Nov-00	0	0	0.000625	0	0.321125
04-Nov-00	0	0	0.00125	0	0.328167
04-Nov-00	0	0	0.00125	0	0.457
05-Nov-00	0	0	0.000625	0	0.469
06-Nov-00	0	0	0.00125	0	0.464542
06-Nov-00	0	0	0.000625	0	0.750376
09-Nov-00	0.08625	0.08875	0.089375	0	1.002626
10-Nov-00	0	0	0	0	1.064668
10-Nov-00	0	0	0	0	1.125668
11-Nov-00	0	0	0	0	1.188168
12-Nov-00	0	0	0.000625	0	1.22021
12-Nov-00	0	0	0	0	1.140668
13-Nov-00	0	0	0	0	1.22271
14-Nov-00	0	0	0.000625	0	1.232126
16-Nov-00	0	0	0	0	1.195084
17-Nov-00	0	0	0	0	1.379376
18-Nov-00	0	0	0.000625	0	1.388626
18-Nov-00	0	0	0	0	1.428001
19-Nov-00	0	0	0	0	1.395584

Figure 10: Re-simulated Results in Microsoft Excel


AEC DCS Demonstration - SCHEDULE - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://ficas.stanford.edu/DC/servlet/DCS?TableName=SCHEDULE&TimeStamp=01-24-2003+15%3A20%3A50&Pr
Back Search Favorites Media

SCHEDULEID	STARTDATE	DURATION	FREEFLOAT	TOTALFLOAT
ID100	11-02-2000 00:00:00	0 Update	0	0
ID110	11-02-2000 00:00:00	4 Update	0	0
ID120	11-06-2000 00:00:00	25 Update	0	0
ID130	12-01-2000 00:00:00	0 Update	0	0
ID140	12-01-2000 00:00:00	9 Update	0	0

Figure 11: Re-simulated Results on a Web Browser

5. SUMMARY AND DISCUSSIONS

In this paper, we have presented a distributed integration infrastructure, which enables the integration of distributed applications as Web services. In addition, we have investigated the use of PSL as an interchange standard and demonstrated the potential of PSL as an effective ontology standard for project management applications. By developing a communication agent for each application and a communication server, data can be exchanged among project management applications regardless of location and platform.

In this paper, we have focused the discussion of the implementation efforts required in developing the prototyped integration infrastructure. In other efforts, we have also investigated ontology mapping issues between PSL and XML-based schemas (Cheng et al. 2002b). Furthermore, we have begun to explore the possibility of using the facts expressed in PSL to reason about consistency and to detect conflicts among project information obtained from different sources (Cheng et al. 2002a). We have also implemented a mediation-based framework for ubiquitous computing and distributed engineering services (Liu et al. 2002a). Our current investigation is to propose a simulation access language and framework to extend the capabilities of existing project management tools.

6. ACKNOWLEDGEMENT

This work is partially sponsored by a Stanford Graduate Fellowship, the Center for Integrated Facility Engineering at Stanford University, and the Product Engineering Program headed by Dr. Ram D. Sriram at NIST. The Product Engineering Program gets its support from the NIST's SIMA (Systems Integration for manufacturing Applications) program and the DARPA's Radeo Program. The Vite SimVision^{TM} is provided by Dr. John Kunz at Stanford University. The authors would like to acknowledge the contributions by Mr. David Liu in the distributed infrastructure described in the paper.

7. REFERENCES

Bosworth, A. (2001), “Developing Web Services.” *Proceedings of the International Conference on Data Engineering,* Heidelberg, Germany, pp. 477-481.

Cheng, J., Law, K.H., Gruninger, M., and Sriram, R.D. (2002a), *"Process Specification Language For Project Information Exchange."* Submitted for publication.

Cheng, J., Trivedi, P., and Law, K.H. (2002b), “Ontology Mapping Between PSL and XML-Based Standards For Project Scheduling.” *3rd International Conference on Concurrent Engineering in Construction, Berkeley*, CA, pp. 143-156.

Genesereth M.R., and Fikes R. (1992). “Knowledge Interchange Format.” Version 3.0 Reference Manual, Computer Science Department, Stanford University, Stanford, CA.

IAI (1997). “Industry Foundation Classes.” Specification Volumes 1-4, International Alliance for Interoperability, Washington, DC.

IAI (2002). “AecXML.” International Alliance for Interoperability, http://www.aecxml.org (December 2002).

ISO (1994). “Product Data Representation and Exchange: Part 1: Overview and Fundamental Principles.” No. 10303-1, International Organization for Standardization.

Liebich, T. (2001). “XML Schema Language Binding of EXPRESS for IfcXML.” MSG-01-001(Rev 4), International Alliance of Interoperability.

Liu, D., Cheng, J., Law, K.H., and Wiederhold, G. (2002a), “Ubiquitous Computing Environment for Project Management Services.” *Proceedings of the International Workshop on Information Technology in Civil Engineering*, Washington, D.C, pp. 273-285.

Liu D., Law K.H., and Wiederhold G. (2002b), "FICAS: A Distributed Data-Flow Service Composition Infrastructure." Stanford University, Unpublished Report, http://mediator.stanford.edu/papers/FICAS.pdf.

Roy, J., and Ramanujan, A. (2001), “Understanding Web services.” *IT Professional*, Vol. 3, No. 6, pp. 69-73.

Runser D.J., and Runser J.A. (2002), “Dissemination of Preliminary Design Build Bid Packages through an Interactive Web Site – An Industry Case Study.” *Proceedings of the International Workshop on Information Technology in Civil Engineering*, Washington, D.C, pp. 357-366.

Schlenoff, C., Ciocoiu, M., Libes, D., and Gruninger, M. (1999). “Process Specification Language: Results of the First Pilot Implementation.” *Proceedings of the International Mechanical Engineering Congress and Exposition*, Vol. 10, pp. 529-539, Nashville, Tennessee.

Schlenoff, C., Gruninger M., Tissot, F., Valois, J., Lubell, J., and Lee, J. (2000). “The Process Specification Language (PSL): Overview and Version 1.0 Specification.” NISTIR 6459, National Institute of Standards and Technology, Gaithersburg, MD.