PSL QUARTERLY PROGRESS REPORT

Project Title: Process Specification and Simulation Access Languages
Date: March 31, 2003
Principal Investigator: Kincho H. Law, Stanford University
Duration: September 1, 2002 – August 31, 2005

PROJECT OBJECTIVES

The objective of the proposed study is to evaluate the process specification language (PSL) and to design and implement a simulation access language (SimAL) with application to project and workflow management. This proposed joint research effort with NIST includes:

- To analyze the concepts of PSL for process information exchange with selected project management applications
- To investigate the reasoning power of PSL for consistency checking and conflict resolution of project information from various sources
- To design and implement a simulation access language for integration of engineering services
- To integrate PSL with SimAL, and develop a demonstration prototype for reusing and integrating results from a variety of application services

Our long-term goal is to develop a distributed network-based framework to integrate process modeling, process specification and virtual simulations of project activities.

PROGRESS AND RESULTS

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. We have investigated the necessary technologies 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 (ISO 2003). 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. 2003).

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 report, we describe a prototype infrastructure to integrate project management applications as Web services.

Project Scheduling Information Exchange Using PSL

We have investigated the PSL ontology and compared it with concepts in project management applications (Cheng et al. 2003). 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.

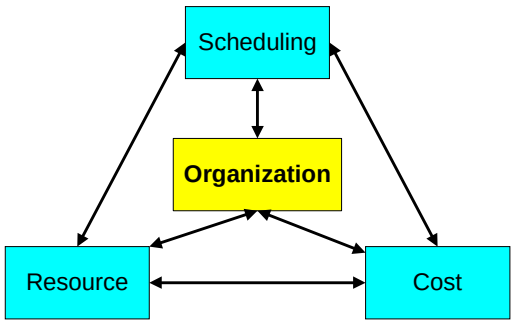

Figure 1: Information in a Construction Project

In a typical construction project, a project schedule consists of a set of activities and the dependency relationships among the activities. The mapping of scheduling information includes mapping activity and dependency relationship information. Table 1 summarizes the major terms used in Primavera P3 and PSL to describe activities and activity dependency relationships.

Table 1: Mapping of Activities and Activity Dependency Relationships

Concepts in Primavera P3	**PSL terms**
Activity	activity occurrence
Predecessor, Successor	before-start, before-finish, after-start, after-finish
Start to Start	before-start
Start to Finish	before-finish
Finish to Start	after-start
Finish to Finish	after-finish
Dependency Lag	before-start-delay, before-finish-delay, after-start-delay, after-finish-delay

Semantic mapping between PSL and project management applications is not always straightforward. For example, the *total float* concept in Primavera P3 cannot be directly mapped to a corresponding PSL term. In Primavera P3, *total float* indicates the maximum amount of time a task can be delayed without postponing the whole project. To express the *total float* concept, we need a set of PSL expressions. For example, the activity *A* is scheduled to finish on October 7, 2002 with a total float of 3 days, while the project *proj1* is scheduled to last 180 days in Primavera P3. To express the *total float* concept in the above example, we need to use the following PSL expressions:

(=> (beforeEQ (endof A) 10/10/2002) (equal (duration-of proj1) 180))

(=> (before 10/10/2002 (endof A)) (lesser 180 (duration-of proj1))

Wrappers have been developed for data exchange among various project management applications, as shown in Figure 2. For example, the Primavera Automation Engine (RA)

is employed to develop a PSL wrapper for Primavera Project Planner™ (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™ and Microsoft Excel™, 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™. In addition, a translator has been built to translate the project data between PSL files and a relational database.

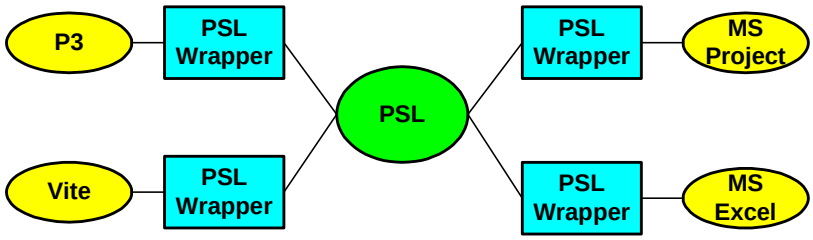

Figure 2: PSL in the Information Exchange

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.

We must note that only the information that is common to all applications can be exchanged through PSL. As shown in Figure 3, Primavera Project Planner™ (P3) includes scheduling, resource and cost information, while Vite SimVision™ provides scheduling, resource, communication, and organizational information. Scheduling and resource information, which is common to both applications, can be exchanged through PSL. Nonetheless, not all scheduling and resource information is exchanged between these two applications, since the granularity of such information may be different. For example, Primavera P3 includes much more detail scheduling information than Vite SimVision™; in other words, not all scheduling information in Primavera P3 is needed by and transferred to Vite SimVision™.

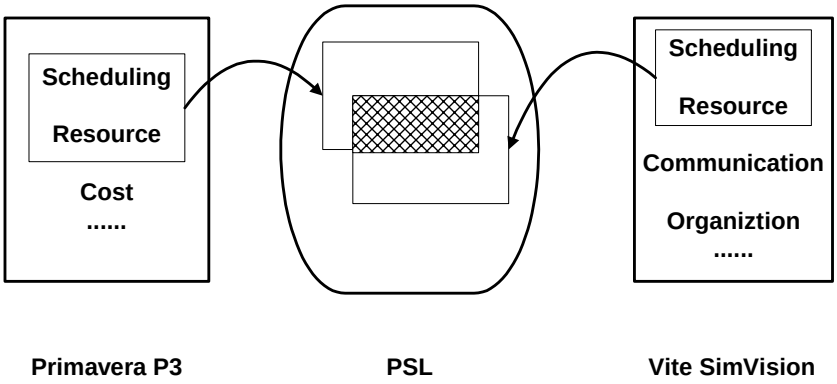

Figure 3: Exchange Information between Primavera P3 and Vite SimVision Through PSL

Integrating Project Management Applications As Web Services

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 4, 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.

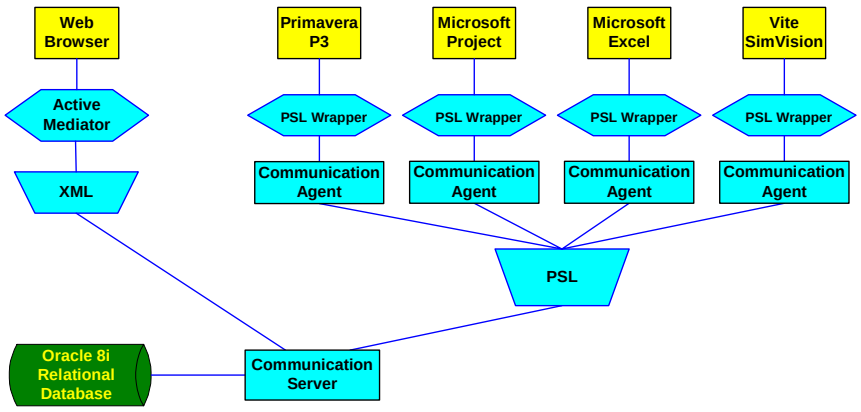

Figure 4: A Distributed Integration Infrastructure

The network communication mechanism for the distributed Web service infrastructure is illustrated in Figure 5. 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 6 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.

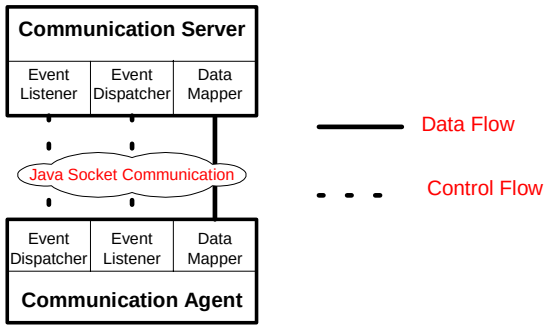

Figure 5: A Network Communication Framework

```
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 6: The Code Segment of an Event Listener

Demonstration of the current results

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 7 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 8 presents the schedule regenerated in Primavera Project Planner™.

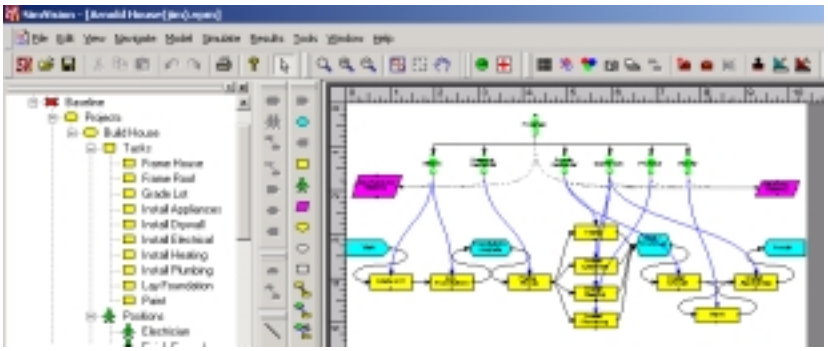

Figure 7: The Arnold's House Project In Vite

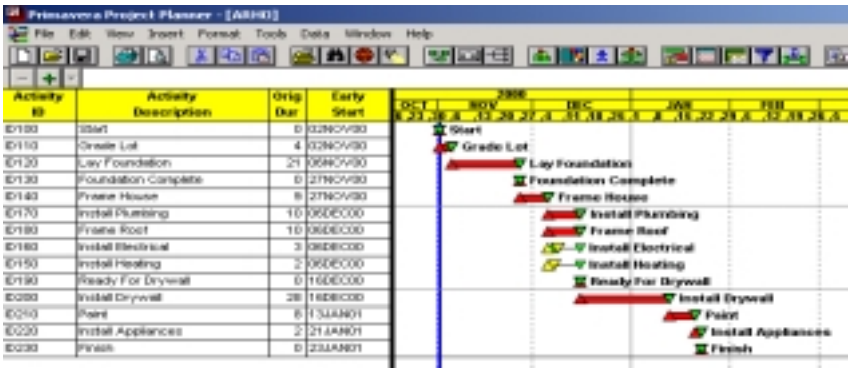

Figure 8: Project Schedule in Primavera P3

Using the distributed integration infrastructure, we can also view and update the project on a Web browser, as shown in Figure 9. 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 10 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 11 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 12. 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.

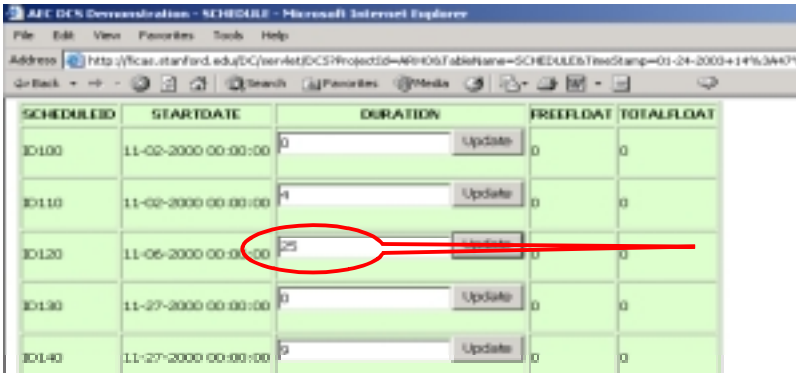

Figure 9: Change Activity Duration on a Web Browser

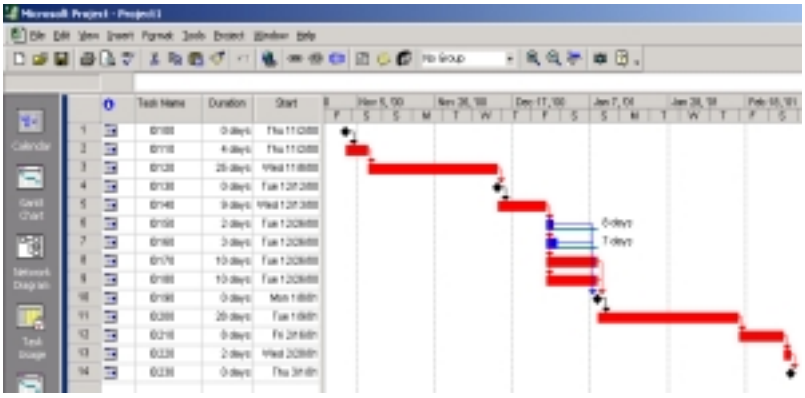

Figure 10: Updated Project Schedule in Microsoft Project

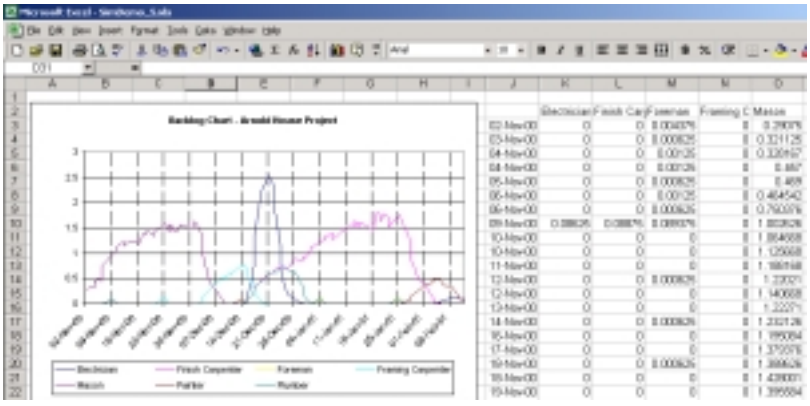

Figure 11: Re-simulated Results in Microsoft Excel

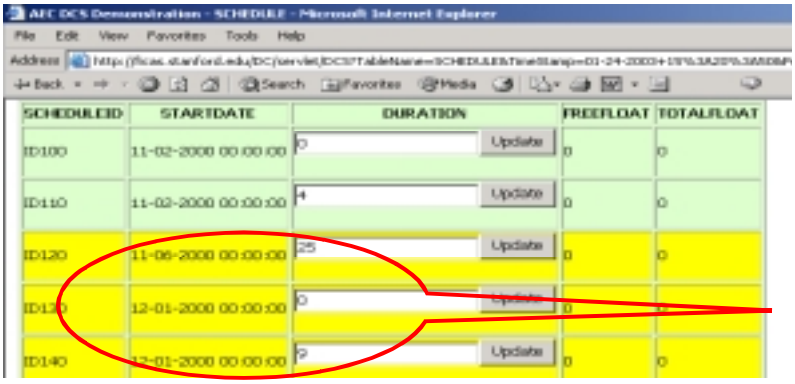

Figure 12: Re-simulated Results on a Web Browser

FUTURE TASKS

We have analyzed the concepts in selected project management applications, including Microsoft Project, Primavera P3 and Vite SimVision, and compared these concepts with the PSL ontology. PSL Wrappers have also been developed to achieve interoperability among these applications. We have also proposed a prototype framework to integrate these project management applications as Web services. Our future tasks include:

- To continue analyzing the concepts of PSL ontology for product and process information exchange, evaluating the use of PSL (and XML-based standards) for selected project management application services, and building PSL wrappers to validate the results;
- To evaluate the use of PSL for consistency checking and conflict resolution on engineering projects.
- To design and implement a simulation access language (SimAL);
- To integrate PSL with SimAL to support product design activities, process simulation and project management applications.

REFERENCE:

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. (2003), *"Process Specification Language For Project Scheduling Information Exchange."* Submitted for publication.

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.

ISO (2003). “Industrial automation system and integration -- Process specification language.” No. 18629-11, 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.