A Simulation Access Language and Framework For Project Management Applications

Jinxing Cheng, Stanford University, Stanford, CA 94305, USA (cjx@stanford.edu)

Gloria T. Lau, Stanford University, Stanford, CA 94305, USA (glau@stanford.edu)

Kincho H. Law, Stanford University, Stanford, CA 94305, USA (law@stanford.edu)

Summary

As computer programs become ever more complex, software development has shifted from focusing on programming towards focusing on integration. This paper describes a simulation access language (SimAL) that can be used to access and compose software applications over the Internet. Specifically, the framework is developed for the integration of tools for project management applications. The infrastructure allows users to specify and to use existing heterogeneous tools (e.g., Microsoft Project, Microsoft Excel, Primavera Project Planner, and AutoCAD) for simulation of project scenarios. This paper describes the components of the SimAL language and the implementation efforts required in the development of the SimAL framework. An illustration example bringing on-line weather forecasting service for project scheduling and management applications is provided to demonstrate the use of the simulation language and the infrastructure framework.

1 Introduction

As computer programs become ever more complex, software development has shifted from focusing on programming towards focusing on integration, as illustrated in Figure 1 (Beringer et al. 1998). In parallel to this trend, there is a shift from standalone applications toward distributed, Web-based or Web-enabled services. As a result, future software development will be based more and more on the composition and integration of existing application components.

To facilitate software integration, Wiederhold et al. (1998) discussed the need of a simulation language that can define a simulation scenario utilizing multiple application tools and has the potential to improve the reusability of simulation tools. The simulation language mirrors the Structured Query Language (SQL) for databases in that SQL enables access of database information through a database-independent language, while the simulation language would allow access of simulation results through an application-independent language. A simulation language has potential applications in many domains. One example is to coordinate workflow processes in project management. Many application tools are now available to perform project management tasks. To manage the information flow among the tools beomes an essential part of workflow management. A simulation language that can support the workflow processes and coordinate the information flows among the tools will greatly facilitate on-line project management applications.

Many languages have been proposed to facilitate the reuse of Web services or software components. Examples include Web Services Flow Language (WSFL) (Leymann 2001), Business Process Execution Language for Web Services (BPEL4WS) (Andrews et al. 2003), and DAML-based Web Service Ontology (DAML-S) (Ankolekar et al. 2001). However, few of these languages are designed for managing and reusing information to support engineering applications. In construction, many standalone applications (e.g., Microsoft Project, Microsoft Excel, Primavera Project Planner, and AutoCAD) are widely used to assist in project management. These tools are designed for specific application and generate large volumes of information that are not easily shared among the applications, partially due to their proprietary data formats.

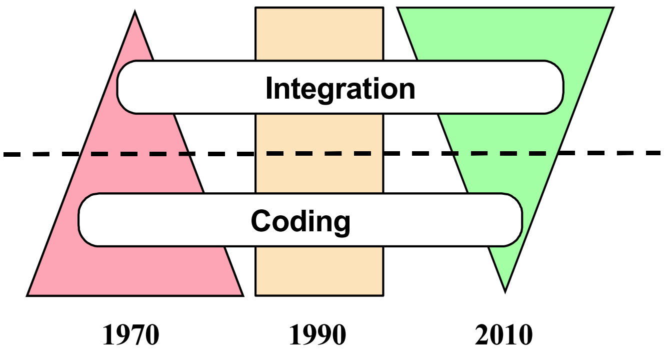


Figure 1: The Trend of Software Development

To illustrate, weather forecasting services are now available on-line and can provide important weather information for planning construction operations. However, this on-line information need to be translated and input to management tools, such as Primavera Project Planner or Microsoft Project, to schedule projects. When unexpected weather conditions occur, it could be difficult and take some efforts to quickly evaluate their impact and to dynamically adjust the project schedule using the tools. Furthermore, not only the data resided in these tools are not well structured and often represented in proprietary formats, these tools may reside at different locations and run on different platforms.

In this paper we propose a simulation access language (SimAL) which, with the necessary infrastructure, would allow users to simulate scenarios utilizing existing project management applications. Speicifically, the simulation access language (SimAL) can be used to access and compose software applications resided in different remote sites. This paper describes the components of the SimAL language and the implementation efforts required in the development of the SimAL framework. An illustration example bringing on-line weather forecasting service for project scheduling and management applications is provided to demonstrate the use of the simulation language and the infrastructure framework.

2 The SimAL Language

2.1 The SimAL Language Specification

The purpose of SimAL is to provide a simple, easy-to-use language to coordinate application tools and to simulate scenarios in assisting decision making. In general, three key factors are involved in decision-making: alternatives, information, and preferences. Alternatives imply that more than one option should be available. Information refers to the knowledge available to users about different options. Preferences specify the aspects that users want to optimize. To support these functions, SimAL includes operational statements for Invocation, Operation, Control, and Decision-Support. The SimAL statements and their syntax are given as shown in Table 1.

Detailed description of the SimAL statements are discussed by Cheng (2004). We use the statements for QUERY and UPDATE to briefly illustrate the features of SimAL. The QUERY and UPDATE statements allow users to query specific project information and to manipulate project models. The first parameter in the statements, *param1*, is a string used to specify the query or update operation. For example, the statement *QUERY("select startTime where activityID = ID100", arho, %%)* can be used to query the start date of the activity *ID100* from the simulated results. To update project data and re-simulate, an UPDATE statement can be used. For instance, the statement *UPDATE("set duration = 4 where activityID = ID110",arho, %%)* re-sets the duration of the activity *ID110*. It can be noted that the syntax of the parameter strings within the QUERY and UPDATE statements is similar to SQL. The operations that are currently supported are tabulated as shown in Table 2. The SELECT operation applies only to the QUERY statement, while the other operations are valid only for the UPDATE statement.

Table 1: SimAL Statements and Syntax

Statements		**Descriptions**
Invocation Statements	SETUP	*ServiceHandle = SETUP("ServiceName")* The SETUP statement is used to establish a communication with a simulation tool through its service name.
	INVOKE	*Variable = ServiceHandle.INVOKE(param1, param2, ...)* The INVOKE statement invokes a simulation service through the handle returned from the SETUP statement.
Operation Statements	QUERY	*QueryHandle = ServiceHandle.QUERY(param1, param2, ...)* The QUERY statement allows users to query specific information from simulation results.
	UPDATE	*UpdateHandle = ServiceHandle.UPDATE(param1, param2, ...)* The UPDATE statement enables users to modify project models.
Control Statements	IF-THEN-ELSE	*IF (expression) THEN {Statement_List} [ELSE {Statement_List}]* The branching statement is used for making dynamic branching decisions based on the value of a Boolean expression following the keyword IF.
	WHILE	*WHILE (expression) { Statement_List}* The while loop statement is used to provide looping operations. The *Statement_List* will be executed continuously as long as the *expression* is evaluated to be true.
Decision-Support Statements	SCENARIO CREATION	*ScenarioHandle = SCENARIO("Scenario Description") { Statement_List }* The SCENARIO statement is used to create a scenario. Here, a scenario refers to a set of actions to accomplish a specific task.
	SCENARIO INSTANTIATION	*ScenarioHandle.SETOBJECTIVES(variable1, variable2,...)* The SETOBJECTIVES statement sets the objectives of a scenario.
	SCENARIO COMPARISION	*CompareHandle = COMPARE(ScenarioHandle1, ScenarioHandle2, ...)* The COMPARE statement compares different scenarios. It is assumed that the objectives of different scenarios on the list are the same, whether they are cost, productivity, duration, or a combination of those attributes.
	RESULT DISPLAY	*Variable = DISPLAY(Variable | CompareHandle, "Description")* The DISPLAY statement is used to display the results of a variable or a comparison.

Table 2: Operations Supported in QUERY and UPDATE Statements

Operation	**Syntax and Description**	
SELECT	Syntax	*SELECT select-list* *[WHERE expression]*
	Description	The SELECT operation is used to query information from the simulation results. Users can specify search conditions in the expression.
SET	Syntax	*SET [variable = expression]+* *[WHERE expression]*
	Description	The SET operation allows users to update project models. For example, users can assign new values to selected attributes.
DELETE	Syntax	DELETE object-name *[WHERE expression]*
	Description	The DELETE operation enables users to delete elements (e.g., ACTIVITY and ACTOR) in project models.

INSERT	Syntax	*INSERT object-name*
		SET [variable=expression]+

	Description	The INSERT operation allows users to insert new elements (e.g., ACTIVITY and ACTOR) into project models, while the attributes of the elements are set by the expressions.

2.2 The SimAL Language Compiler

The SimAL compiler has been implemented using Java Compiler Compiler (JavaCC) (SUN 2004), a parser generator which reads a grammar specification and converts it to a Java program. The parsing and compiling process follows three basic steps:

- Lexical analysis: The token manager reads in a sequence of characters and produces corresponding objects called "tokens." The sequence of characters is broken into tokens according to the SimAL lexicon conventions.
- Syntax analysis: JavaCC uses the production rules in the SimAL specification (Cheng 2004) to generate a parser in Java.
- Building XML trees: The parser then generates an XML tree based on the events defined in the SimAL specification (Cheng 2004). The XML tree is used by the SimAL system to invoke and coordinate distributed tools at run-time.

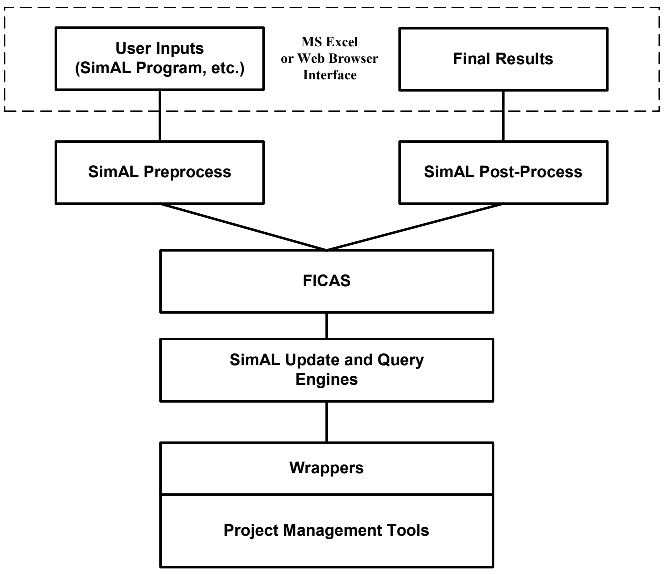

User Inputs (SimAL Program, etc.)
MS Excel or Web Browser Interface
Final Results
SimAL Preprocess
SimAL Post-Process
FICAS
SimAL Update and Query Engines
Wrappers
Project Management Tools

Figure 2: The SimAL Framework

3 The SimAL Framework and Implementation

3.1 The SimAL Framwork

Figure 2 shows the overall framework of the SimAL prototype system. The SimAL system includes a software composition infrastructure based on data and control flows (Liu et al. 2003) and software wrappers. The Flow-based Infrastructure for Composing Autonomous Services (FICAS) (Liu et al. 2003) is utilized to invoke distributed services and to direct data flow among different services. While there are other solutions available for distributed service invocation, include Remote Procedure Call (RPC), Common Object Request Broker (CORBA), and Simple Object Access Protocol (SOAP), FICAS is designed to handle applications that involve high volume of data typically found in engineering applications. Specifically, FICAS takes advantage of distributed data flows to efficiently route the data to designated applications (Liu et al. 2003). Wrappers for the project management application tools are developed based on the Process Specification Language (PSL) which is a standard interchange language for process data developed at NIST (ISO 2003, Cheng et al. 2003).

A SimAL program is first processed by the preprocessing engine. The preprocessor parses the SimAL program, produces instructions for FICAS to invoke relevant services, and generates necessary information to display the simulation results. FICAS then establishes connections to appropriate services and instructs them to simulate various project tasks. The update and query engine is employed to filter the results generated by the tools and to update project models. Once the simulation is completed by the different tools, the simulation results are processed by the post-processing engine and are displayed to the users in appropriate formats.

3.2 Implementation Efforts

Wrappers act as a bridge between FICAS and the project management tools in that FICAS invokes these tools and retrieves their simulation results through the wrappers. Depending on the invocation methods, project management tools can be categorized into two types: standalone services and embedded services. A standalone service is an application that can run independently and can be invoked directly through its wrapper. An embedded service is an application that has to run inside another tool and cannot be invoked directly by FICAS. Example of a standalone service is an independent application tool such as the Primavera

Project Planner, which can be accessed directly through its wrapper. An example of an embedded service is a tool built within a specific software such as a cost-estimating tool which is built and run inside Microsoft Excel.

A service directory is employed for the registration, discovery, and invocation of the application tools. The directory maps the name of the service to the information, such as the network location and the TCP/IP port number of the application, needed by the SimAL system to invoke the tool. The service directory is structured in XML formats, as shown in Figure 3. Each individual application is represented by an XML element, *SERVICE*, whose child elements specify the parameters of the application. Within the *SERVICE* elment, the *NAME* element specifies the name of the application, the *SERVER* element contains the IP address of the machine the application is running on, and the *PORT* element indicates the TCP/IP port to which the application listens.

```
<?xml version="1.0"?>

<SERVICEDIRECTORY>
  <SERVICE>
    <NAME>ServicePsl</NAME>
    <SERVER>psl.stanford.edu</SERVER>
    <PORT>2409</PORT>
  </SERVICE>
  <SERVICE>
    <NAME>ServiceP3</NAME>
    <SERVER>psl.stanford.edu</SERVER>
    <PORT>2410</PORT>
  </SERVICE>
  <SERVICE>
    <NAME>ServiceNotification</NAME>
    <SERVER>psl.stanford.edu</SERVER>
    <PORT>2412</PORT>
  </SERVICE>
  <SERVICE>
    <NAME>ServiceWeatherForecast</NAME>
    <SERVER>psl.stanford.edu</SERVER>
    <PORT>2413</PORT>
  </SERVICE>
   ......
</SERVICEDIRECTORY>
```

Figure 3: Service Directory

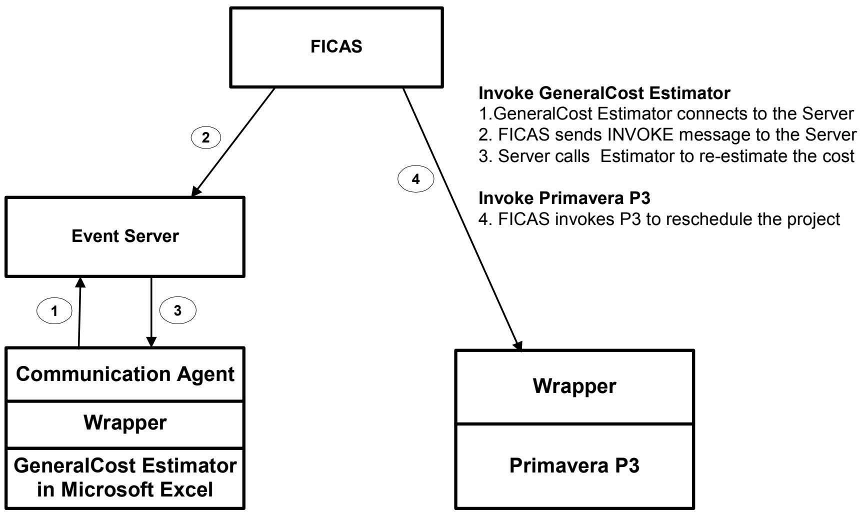

FICAS
Invoke GeneralCost Estimator
1.GeneralCost Estimator connects to the Server
2. FICAS sends INVOKE message to the Server
3. Server calls Estimator to re-estimate the cost
Invoke Primavera P3
4. FICAS invokes P3 to reschedule the project
2
4
Event Server
1
3
Communication Agent
Wrapper
GeneralCost Estimator in Microsoft Excel
Wrapper
Primavera P3

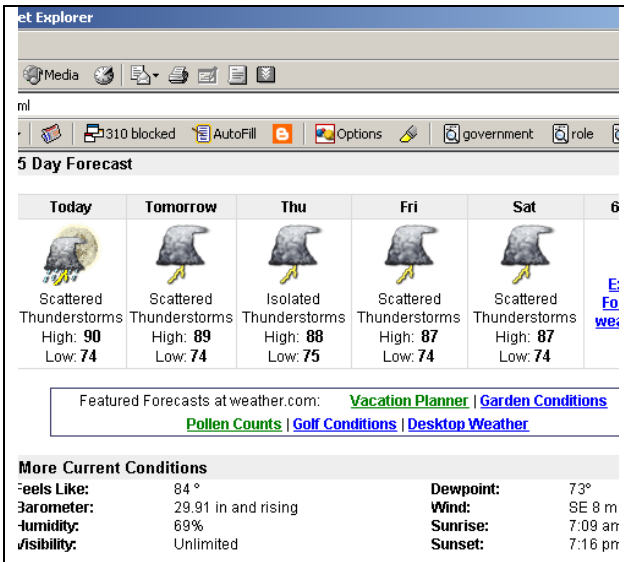
Figure 4: The Invocation of Embedded and Standalone Services

For standalone services, SimAL looks up the service directory, invokes the services through the wrappers, and accesses the generated results. For embedded services, which cannot be invoked directly by the SimAL program, an event server and a communication agent for each service is developed by dynamically opening the software before invoking the embedded service. Figure 4 illustrates the difference between invoking embedded and standalone services using SimAL.

4 Application Demonstration

To demonstrate the potential application of the prototype SimAL system and framework, we use the data from the Arnold's House project example as described in the tutorial of Vite's SimVision Software manual (EPM 2003). In particular, we demonstrate how the system framework can incorporate online weather information, coordinate various computer application tools, and allow users to review the impact on the project due to weather conditions.

We develop a wrapper/parser to extract the information from an online weather forecasting service (in HTML formats) and to convert the weather information into XML formats. Figure 5 shows the original weather information as shown from the Yahoo's forecasting service site (http://weather.yahoo.com) and the generated information in XML.



5 Day Forecast

Today	Tomorrow	Thu	Fri	Sat	6
Scattered Thunderstorms	Scattered Thunderstorms	Isolated Thunderstorms	Scattered Thunderstorms	Scattered Thunderstorms	
High: **90**	High: **89**	High: **88**	High: **87**	High: **87**	
Low: **74**	Low: **74**	Low: **75**	Low: **74**	Low: **74**	

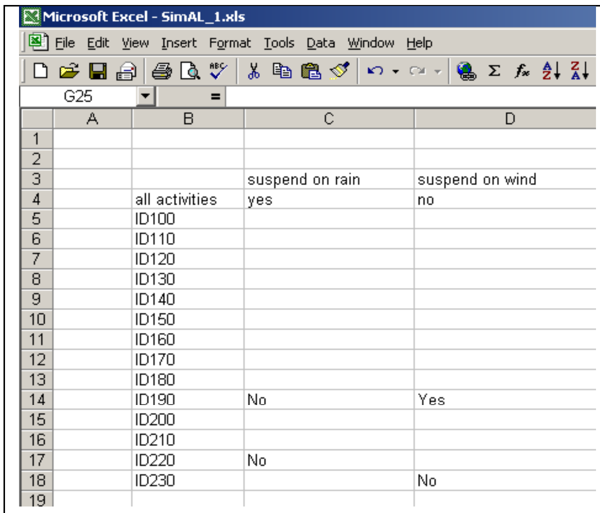
Featured Forecasts at weather.com: Vacation Planner | Garden Conditions
Pollen Counts | Golf Conditions | Desktop Weather

More Current Conditions

eels Like:	84 °	**Dewpoint:**	73°
Barometer:	29.91 in and rising	**Wind:**	SE 8 m
Humidity:	69%	**Sunrise:**	7:09 am
Visibility:	Unlimited	**Sunset:**	7:16 pm

```
<?xml version="1.0"?>
<WeatherReport>
<weather date="2003-9-23">
<location>
<zipcode value="33410" />
</location>
<conditions value=" Isolated thunderstorms
early, mainly cloudy overnight with a few
showers" />
<temperature>
<templow c="23.3" f="74.0" />
<temphigh c="32.2" f="90.0" />
</temperature>
……
</weather>
……
```

Figure 5: Translating Weather Information from an Online Service into XML Format


Microsoft Excel - SimAL_1.xls
File Edit View Insert Format Tools Data Window Help
G25
=

	A	B	C	D
1				
2				
3			suspend on rain	suspend on wind
4		all activities	yes	no
5		ID100		
6		ID110		
7		ID120		
8		ID130		
9		ID140		
10		ID150		
11		ID160		
12		ID170		
13		ID180		
14		ID190	No	Yes
15		ID200		
16		ID210		
17		ID220	No	
18		ID230		No
19				

```
<?xml version = "1.0"?>
<weatherScheduling>
<actImpact actid = "ALL" suspendOnRain
= "yes" suspendOnWind= "no" />
<actImpact actid = "ID190"
suspendOnRain = "No" suspendOnWind =
"Yes" />
<actImpact actid = "ID220"
suspendOnRain = "No" />
<actImpact actid = "ID230"
suspendOnWind = "No" />
</weatherScheduling>
```

Figure 6: Encoding Domain Knowledge in XML via a Excel Spreadsheet

Weather can have significant impacts on scheduling activities in construction projects. Heavy rain as well as strong winds may cause many construction activities to be suspended. The exact influence of weather conditions varies from activity to activity. For example, exterior concrete works may need to be suspended due to rain conditions, while interior works can be performed as usual. In this demonstration, the domain knowledge relating weather conditions to activities can be specified by the users in a tabulated format using Microsoft's Excel. Figure 6 shows the table recording the domain knowledge and the corresponding expressions in an XML file.

Let's consider a hypothetical scenario and assume that the project encounters prolonged rainy season. Within the Excel spreadsheet, the user can specify the instructions in SimAL statements as shown in Figure 7. As shown in the figure, a SimAL program starts with the keyword *SimAL* followed by the program name, *WeatherDemo*. The keyword *SimAL* indicates that the program follows the syntax of the SimAL language.

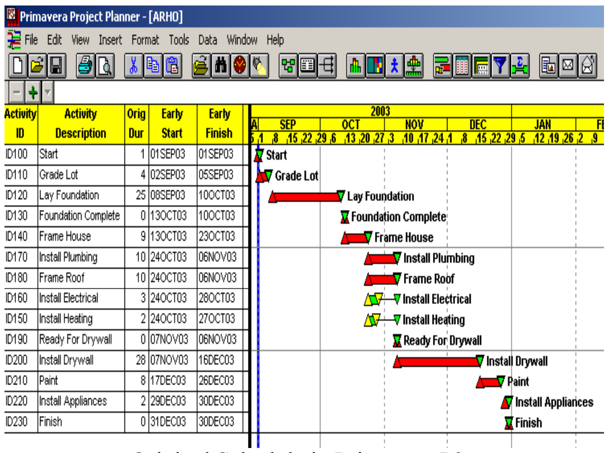
```
  SimAL WeatherDemo
  {
        p3_svc = SETUP("ServiceP3")
        psl_svc = SETUP("ServicePsl")
        vite_svc = SETUP("ServiceVite")
        notification_svc = SETUP("ServiceNotification")
        wforecast_svc = SETUP("ServiceWeatherForecast")
        wprocess_svc = SETUP("ServiceWeatherProcess")

        psl = psl_svc.INVOKE("to-psl", %%)
        wf = wforecast_svc.INVOKE("RetrieveForecast", %%)
        wp = wprocess_svc.INVOKE("ProcessForecast", wf_arho, arho, %%)
        p3 = p3_svc.INVOKE("reschedule", wp_arho, %%)
        vite = vite_svc.INVOKE("simulate", arho1, %%)

        notif = notification_svc.INVOKE("psl.stanford.edu", 8250, status)

}
```

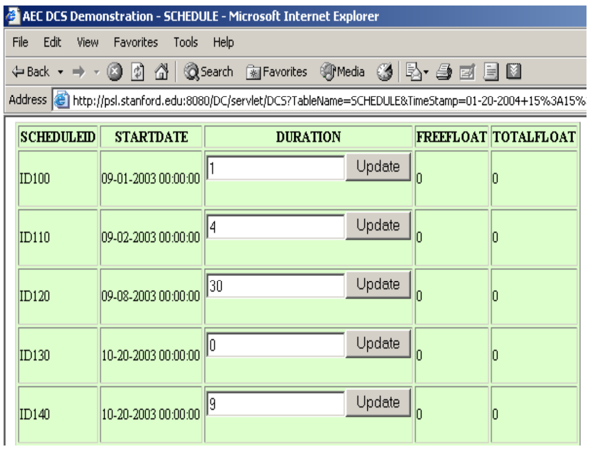
Figure 7: An Example SimAL Program to Simulate the Impact of Weather Conditions



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

Activity ID	Activity Description	Orig Dur	Early Start	Early Finish
ID100	Start	1	01SEP03	01SEP03
ID110	Grade Lot	4	02SEP03	05SEP03
ID120	Lay Foundation	25	08SEP03	10OCT03
ID130	Foundation Complete	0	13OCT03	10OCT03
ID140	Frame House	9	13OCT03	23OCT03
ID170	Install Plumbing	10	24OCT03	06NOV03
ID180	Frame Roof	10	24OCT03	06NOV03
ID160	Install Electrical	3	24OCT03	28OCT03
ID150	Install Heating	2	24OCT03	27OCT03
ID190	Ready For Drywall	0	07NOV03	06NOV03
ID200	Install Drywall	28	07NOV03	16DEC03
ID210	Paint	8	17DEC03	26DEC03
ID220	Install Appliances	2	29DEC03	30DEC03
ID230	Finish	0	31DEC03	30DEC03

Original Schedule in Primavera P3



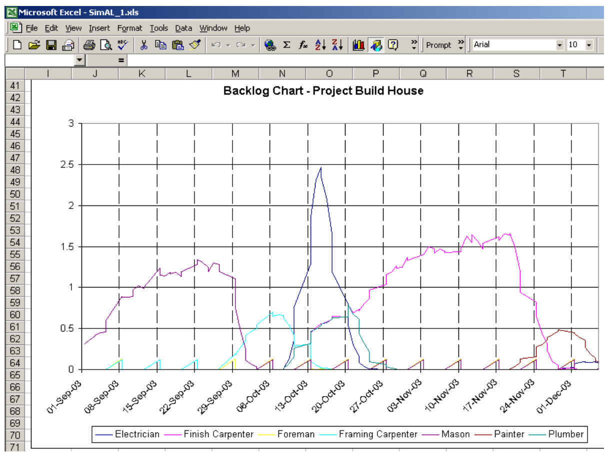
AEC DCS Demonstration - SCHEDULE - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Media
Address

http://psl.stanford.edu:8080/DC/servlet/DCS?TableName=SCHEDULE&TimeStamp=01-20-2004+15%3A15%

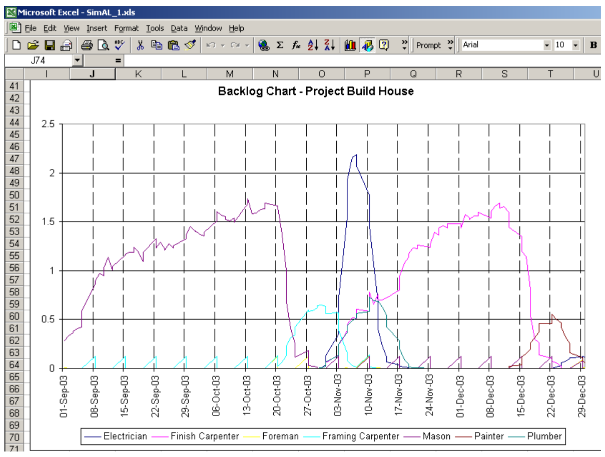
SCHEDULEID	STARTDATE	DURATION	FREEFLOAT	TOTALFLOAT
ID100	09-01-2003 00:00:00	1 Update	0	0
ID110	09-02-2003 00:00:00	4 Update	0	0
ID120	09-08-2003 00:00:00	30 Update	0	0
ID130	10-20-2003 00:00:00	0 Update	0	0
ID140	10-20-2003 00:00:00	9 Update	0	0

Review Updated Schedule Online

Figure 8: Review the Impact of Weather Conditions on Project Schedule using a Web Browser



Original Backlogs in Chart



Updated Backlogs in Chart

Figure 9: Review the Impact of Weather Conditions in MS Excel Charts

As shown in Figure 7, the program *WeatherDemo* first includes the statements for setting up the services needed for the simulation. The program then invokes a weather forecast service, extracts the results, and encodes them in XML format. A scheduling software, Primavera Project Planner, is then invoked to adjust the schedule. The updated schedule is stored in PSL format and transferred to Vite's SimVision for further analysis (for example, to determine task backlogs). Figures 8 and 9 show the impact of the weather on the project schedule and on the task backlogs, respectively. As shown in Figure 8, the activity "*Lay Foundation*" has been prolonged from 25 (as displayed in Primavera P3) to 30 days (as displayed in a web browser). Although not shown here, the delay will, in turn, cause delays in the other activities. Figure 9 shows the pattern of task backlogs as analysed using Vite's SimVision and displayed in Micrsoft Excel; it can be seen that the peak values of task backlogs and the associated dates have been altered.

5 Summary

In this paper, we have presented a simulation access language (SimAL) and a prototype framework to enable simulation using heterogeneous application tools. SimAL is a simple and high-level language that can be used to specify the tasks and the tools involved in a simulation.

The SimAL framework allows the user to take advantage of the features and the strengths of individual application tools and to coordinate and integrate them to perform the simulation. Specifically, we have demonstrated the potential use of SimAL for project scheduleing and management applications.

We have illustrated the use of SimAL in bringing online weather information to project management. It should be noted that, in the demonstration scenario, the entire simulation of weather impact on the project schedule is automated. The simulation involves a number of very sophisticated application tools such as Primavera P3 and Vite's SimVision. The results can be displayed using software packages commonly available at a typical office such as Microsoft Excel, Microsoft Project, and a web browser. A project manager in an office can quickly review the impact and make decisions to adjust project activities and schedule accordingly whenever there is a concern on weather conditions. It should also be noted that the project management services can reside at different locations and run on different platforms. We have also conducted a number of demonstrations between Glasgow Caledonian University in Scotland and Stanford University. In the demonstrations, the Oracle database server and the Vite's SimVison are located at Stanford University, the scheduling services reside at Glasgow Caledonian University, and the Yahoo's weather forecasting service is an outside service available online. In summary, the potential value of a simulation access language and framework has been illustrated from the example scenario described in this paper and the demonstrations of the prototype system to run distributed project management application tools at multiple sites.

6 Acknowledgement

This work is partially sponsored by a Stanford Graduate Fellowship 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 authors would like to acknowledge the contributions by Dr. David Liu to the system framework described in the paper. The authors would like to thank Prof. Bimal Kumar of Glasgow Caledonian University for his help in demonstrating the framework for supporting distributed applications.

7 References

Andrews, T., Curbera, F., et al. (2003). "BPEL4WS Specification: Business Process Execution Language for Web Services Version 1.1." BEA, IBM, Microsoft, SAP AG, and Siebel Systems, http://ifr.sap.com/bpel4ws/ (30 March 2004).

Ankolekar, A., Burstein, M., et al. (2001). "DAML-S: Semantic Markup for Web Services." *The International Semantic Web Working Symposium*, Stanford, CA.

Beringer, D., Tornabene, C., et al. (1998). "A Language and System for Composing Autonomous, Heterogeneous and Distributed Megamodules." *DEXA International Workshop on Large-Scale Software Composition*, Vienna, Austria.

Cheng, J. (2004). "A Simulation Access Language and Framework with Applications to Project Management." Ph.D. Dissertation, Under Preparation, Stanford University, Stanford, CA.

Cheng, J., Gruninger, M., Sriram, R. D., and Law, K. H. (2003). "Process Specification Language For Project Scheduling Information Exchange." *International Journal of IT in Architecture, Engineering and Construction*, vol. 1(4), pp. 307-328.

EPM (2003). "SimVision Tutorial: Building Basic Models." EPM, Revision 5.

ISO (2003). "Industrial Automation System and Integration -- Process Specification Language." No. 18629-11, International Organization for Standardization.

Leymann, F. (2001). "Web Services Flow Language (WSFL 1.0)." IBM Corporation, http://www.ibm.com/software/solutions/webservices/pdf/WSFL.pdf (30 March 2004).

Liu, D., Peng, J., Law, K.H., Wiederhold, G., and Sriram, R.D. (2003). “Composition of Autonomous Services with Distributed Data Flows and Computations.” *ACM Transactions on Internet Technology*, Submitted for Publication.

SUN (2004). “ Java Compiler Compiler(JavaCC) - The Java Parser Generator.” Sun Microsystems, https://javacc.dev.java.net/ (30 March 2004).

Wiederhold, G., Jiang, R., and Garcia-Molina, H. (1998). “An Interface for Projecting CoAs in Support of C2.” *Proceeding of the Command & Control Research & Technology Symposium*, Naval Postgraduate School, Monterey CA, pp. 549-558.