# Internet-Enabled Distributed Engineering (Web) Services

J. Peng[1], D. Liu[1], J. Cheng[1], C.S. Han[1] and KH. Law[1*]

[1] Engineering Informatics Group, Department of Civil and Environmental Engineering, Stanford University, Stanford, CA, USA

## ABSTRACT

The emergence of the Internet and communication technologies will have significant impacts in the life cycle project development and management of civil infrastructures. This paper presents the basic concepts of web services technology and its potential applications in Civil Engineering. The web services model is becoming a popular approach for integrating software applications and to improve the flexibility and extend the functionalities of a software application by making it interoperable with other software services. Three example applications are presented to demonstrate that the web service approach is a promising paradigm for integrating large engineering software applications.

**Keywords:** Internet computing, engineering web services, megaservice, engineering simulation

## 1. INTRODUCTION

The world of Internet computing is evolving very quickly. Simply defined, a web service is a combination of software applications and data that can be accessed from any web-enabled devices. Using a set of standardized protocols, web services can be universally accessed and deployed, independent of the underlying operating environment. The basic idea behind web services is to adopt the web programming model for developing generic software applications that are not necessarily browser-based. The goal is to provide a platform for building distributed applications that utilize software components developed using different programming languages (possibly by different developers or vendors) and running on different operating systems, computer platforms and devices. Acting as adapters for complicated process components and legacy applications, web services allow disparate systems to work together.

The web services model is also becoming a favorite approach for integrating engineering applications in that the model can improve the flexibility and extend the functionalities of a software application by making it interoperable with other software services. An engineering simulation may now involve a number of software applications that run on geographically distributed computers. For example, the architects, structural engineers, and construction team of a project may reside in different locations and use separate computer systems and software packages for engineering analysis and design. A project management application may take advantage of and dynamically integrated with on-line services. The simplicity of the web services model makes it possible to build a complex software system incrementally. This paper presents a few examples to illustrate the potential applications of web services in civil and building engineering.

## 2. EMERGENCE OF WEB SERVICES

As software becomes ever more complex, there is a shift from coding as the focus of programming to a focus on integration[4], as illustrated in Figure 1. Traditionally, large programs are partitioned into subtasks of manageable sizes. The subtasks are assigned to programmers who code the instructions in a programming language. The resulting program segments are subsequently incorporated into the software package. As more and more program segments are pre-constructed and packaged, increasingly, a large portion of the overall software engineering effort is being spent on integration.

* Corresponding author. Email: law@stanford.edu.

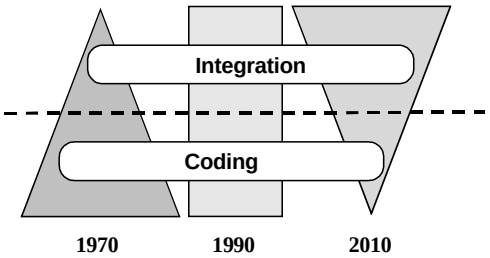



**Figure 1. Trend of Software Development (Courtesy of Prof. Gio Wiederhold, Stanford University)**

### 2.1. Software Integration

Software integration takes place in many forms. Early attempts are primarily based on code reuse. The simplest method is to copy the source code to wherever the desired functionality is needed. There are many drawbacks to this approach, ranging from compiler incompatibility to difficulty in maintaining duplicate copies of code. To deal with these drawbacks, shared libraries are used in place of copied code. Software components written using the same programming language are compiled into shared libraries. The shared libraries have public interfaces, through which the users can invoke the functions contained in the libraries. Generally speaking, software integration based on code reuse assumes that the ownership of the reused software components belongs to the users of the software components. In addition, the software components are executed on the same machine as the invoker (client) of the components.

Advances in network computing, especially the emergence of the Internet, allow software components to be distributed to multiple machines. Each software component runs as a separate process, communicating with each other by exchanging messages. Distributed components require to have well-defined interfaces and constraints, and are normally managed in a decentralized manner. The distributed components whose interfaces are published on the web are generally referred to as web services[20].

Web services can be regarded as the atomic building blocks for service integration. The functionalities provided by the web services are composed together to form an integrated megaservice. Although the web services are distributed and heterogeneous, they are utilized as if they were locally available to the megaservice. Communication messages are exchanged among the web services to coordinate the execution of the web services and to exchange data among the web services. To achieve interoperability, a set of communication protocols is needed for exchanging data among the web services. Various standardized communication protocols that can be used to build web services have been proposed. Examples of standard communication protocols include Common Object Request Broker Architecture (CORBA)[19], Microsoft's Distributed Component Object Model (DCOM)[8], Java Remote Method Invocation (RMI)[18], and Simple Object Access Protocol (SOAP)[5]. Distributed components may be written in different languages, and can be compiled by different compilers, while they communicate with each other via standardized protocols.

### 2.2. Integration Models

Models for integrating software components can be roughly categorized into two groups, namely tightly coupled and loosely coupled. For tightly coupled model, software components are typically managed under a single administrative domain. The software components follow a set of proprietary rules that allow access to software components located in other machines. For software components managed under multiple administrative domains, loosely coupled component model is preferred for integration. The software components in the loosely coupled model exist as autonomous services, each of which is controlled by its own service provider. A typical web service application uses the loosely coupled component model. Unlike in the tightly coupled distributed object systems where all the pieces of an application are deployed at once, the web services model allows services to be added as needed. The connections to a new web service can be established during the system runtime.

Web service applications range from comprehensive services such as storage management and customer relationship management to more specific services such as travel reservation, book purchasing, weather forecasts, financial data summaries, and newsgathering. To coordinate the execution of a number of individual services together, forming a megaservice, many standard languages, such as Web Services Flow Language (WSFL)[12], Business Process Execution Language for Web Services (BPEL4WS)[2], and Web Service Ontology based on DARPA Agent Markup Language (DAML-S)[3], have been proposed. These service description languages, however, are mainly targeted for business oriented applications and are not designed for managing and reusing information to support engineering applications. For example, engineering simulation typically involves large volume of data and it is more beneficial to separate

execution controls and (distributed) data (message) communication. In addition to service description languages, data representation and exchange standards, such as Industry Foundation Classes (IFC)[10], Standards for the Exchange of Product Data (STEP)[11], Extendible Markup Languages (XML)[22], Process Specification Language (PSL)[21], play a significant role in facilitating the interoperability of engineering applications.

The objective of our research in web services is to develop methodologies that can effectively wrap legacy applications and make them accessible over the network. Different applications would require different models for integration and coordination. The following describe a number of demonstrative examples in civil and building engineering to illustrate the methodologies and potential applications of engineering web services.

## 3. APPLICATION EXAMPLES

### 3.1. A Web Service Example for Building Design.

The first example application is a distributed service architecture that allows building design web services to be incorporated into a modular network-enabled infrastructure[9]. Building designs are often required to comply with design codes and guidelines. Figure 2 shows a prototype framework for on-line review with a partially automated on-line compliance process using Internet and web-based technologies. At any point in the design process, the user can send the design to a code-checking program that resides on a remote server. The code-checking program examines the design data and summarizes the results in a generated web page. The web page contains a graphical representation of the building model along with "redline" information with hyperlinks to specific comments. When applicable, the comments have hyperlinks to the actual building code document provisions.

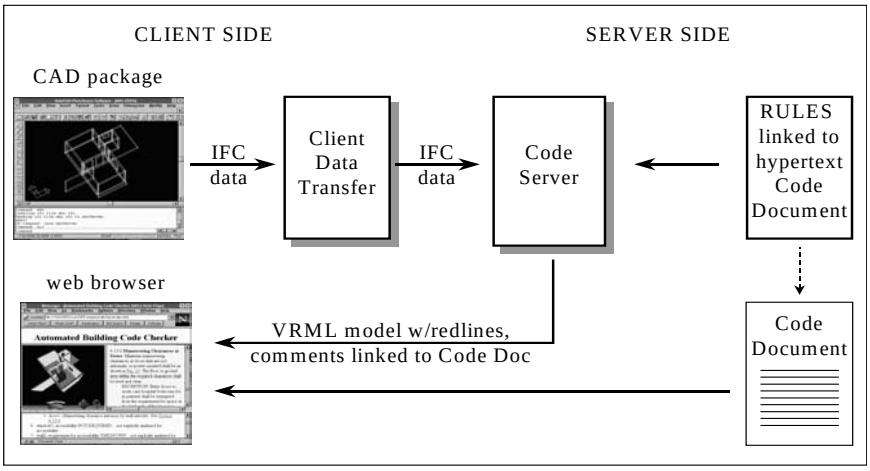



**Figure 2: An Online Compliance Assistance Framework**

In the US, among the numerous provisions governing a facility design, the two issues that have been identified by facility managers and building inspectors as most significant are *accessibility* and safe *egress*. This pilot study focuses on design compliance with regulations governing disabled access[1] and includes a path-planner to simulate wheelchair access. Figure 3 shows the conceptual Internet-enabled distributed framework with web services and a broker. Each individual service adheres to a three-tiered architecture. The first tier, a communication protocol interface developed using CORBA, gives the application services a common means to send and receive design data over the Internet. The middle tier, the common product model interface following the IFC standard, is a standard protocol that describes the design data. The third tier is the core of the design service – the design service extracts the appropriate information of the building design through the common product model interface and either modifies the design data or generates a report based on the analysis of the data. The broker does not need the product model interface that is present in the services. An application package needs simply register itself with the broker to advertise its services in the infrastructure. Another service will query the broker for the existence of services in the distributed service architecture. The registration and query service is based on a predefined syntax, but the broker does not have to be aware of the underlying product model that is being used to exchange design data between services. As an example, Figure 4 shows the results for the compliance checking of a floor plan and the simulation of wheelchair access of the facility.

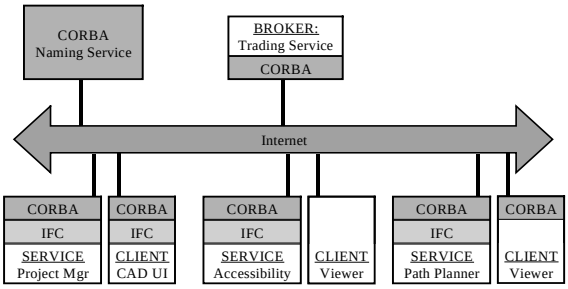



**Figure 3. A Web Service Architecture for Building Design Compliance Analysis**


**(a) Compliance Code Checking Result** **(b) Performance Simulation using a Path-Planner Program**

**Figure 4. Accessibility Analysis Service and Generated Report**

### 3.2. A Web Service Example for Structural Analysis

The second example is a web service framework designed to facilitate the utilization of a collaborative finite element structural analysis program[16,17]. A finite element analysis program is constructed as a web service to allow easy access to the analysis program and the analysis results by using a web-browser or other application programs. Although the finite element analysis program can be viewed as a single web service from the users' perspective, the analysis program can actually be running on disparate computers. In this study, the core service is built upon an object-oriented finite element program, OpenSees[15], and it serves as the entry point for users' requests. Elements, materials, and solution strategies can be constructed as web services and run on distributed computers to facilitate structural analyses. The modules of the collaborative software framework are schematically depicted as shown in Figure 5.

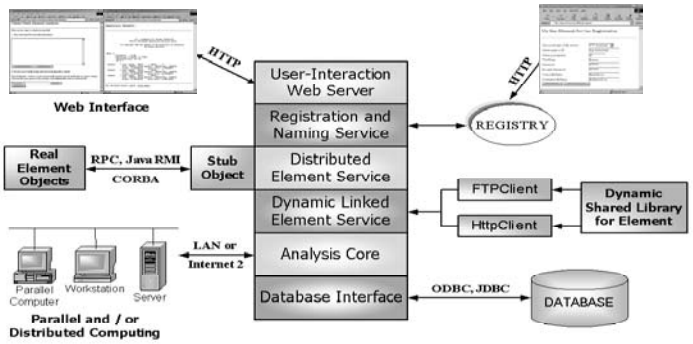



**Figure 5: System Modules for a Collaborative Finite Element Structural Analysis Software**

In the prototype implementation, Java RMI is chosen to handle the network communication for distributed element services. As an example, a "specialized" *ElasticBeamColumn* element, which is not available in the analysis core, is built as an on-line web service. Figure 6 illustrates the interaction among the web services during a simulation of the model. The analysis core service is running on a server computer called *opensees.stanford.edu*. The developed *ElasticBeamColumn* element service is running on a computer named *galerkin.stanford.edu*. A MATLAB-based post-processing web service is deployed on *epic21.Stanford.edu*. Since it is the core server that handles the communication with the element service, users only need to know the location of the central server (*opensees.Stanford.edu*) without the awareness of the underlying distributed model and the physical locations of the services.

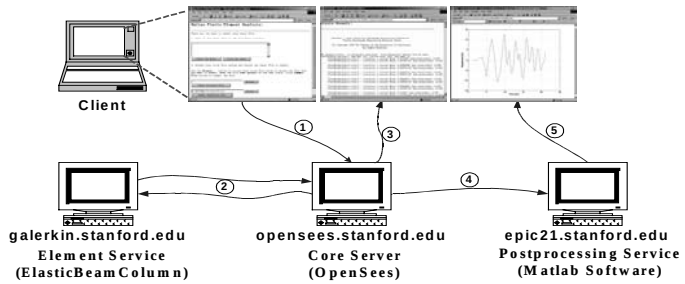



**Figure 6. Interaction of Distributed Web Services for a collaborative structural analysis**

### 3.3. A Web Service Example for Project Management

The third example is a web service application for project management and scheduling. In this work, web services are linked together through an integration framework, which is designed to efficiently handle large volume of engineering data communication, to compose software applications and to form a megaservice[13,14]. As shown in Figure 7, commercial software applications, such as Microsoft Project, Excel, Primavera Project Planner, Vite SimVision, and 4D Viewer are wrapped as web services that export their functionalities. Applications run on heterogeneous platforms and can be accessed homogeneously through standard web services interfaces. Functionalities from various software applications can be brought together to complete a specific engineering task. The prototype also incorporates a variety of devices ranging from PDA, web browsers, desktop computers, and server computers to support ubiquitous access to the information and simulation applications. In short, by using the web services model to develop the integration framework, project management applications can interoperate regardless of locations and platforms.

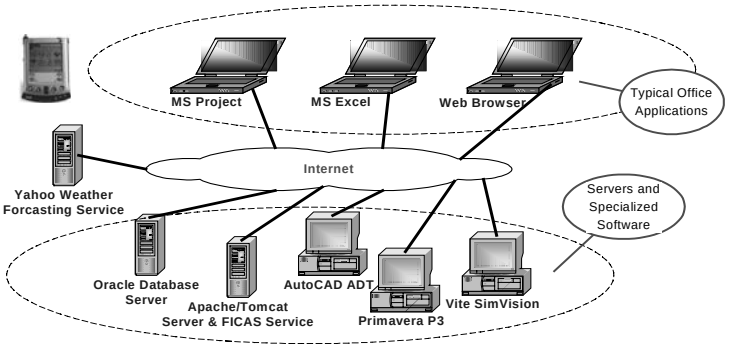
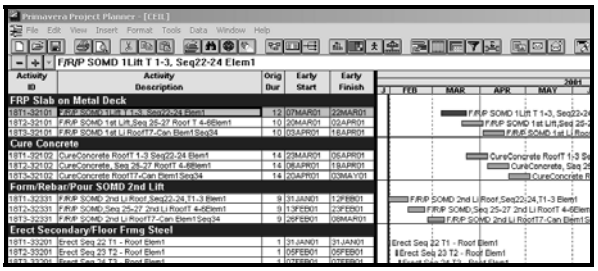
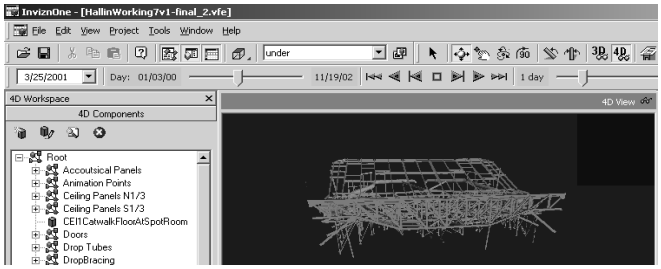



**Figure 7. Web Service Infrastructure for Project Management Applications**

Let's first look at a sample scenario to demonstrate how the engineering service infrastructure may help facilitate personnel from different groups conduct collaborations. The test case model and the scheduling information as displayed using Primavera are shown in Figure 8. The project data is shared between the relational data model and the Primavera's data model by using a standard data model expressed in PSL[6]. As shown in Figure 9, the project schedule can be reviewed using a handheld PDA device to access the information stored in a relational database. Suppose that the duration for the activity, 18T1-33201, is hypothetically changed from 1 day to 40 days using the PDA. The update will then trigger other application services to modify the project schedule. As shown in Figure 10, the updated schedule can be retrieved from the relational database using Microsoft's Project and the updated displayed with the 4D Viewer.




**(a) Project Schedule as displayed in Primavera P3**




**(b) Sample Project on 4D Viewer**

**Figure 8. An Example Test Model and Its Project Schedule**

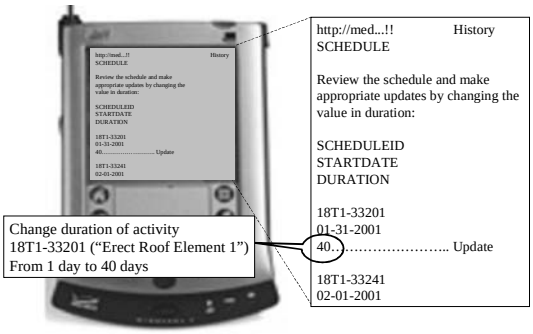
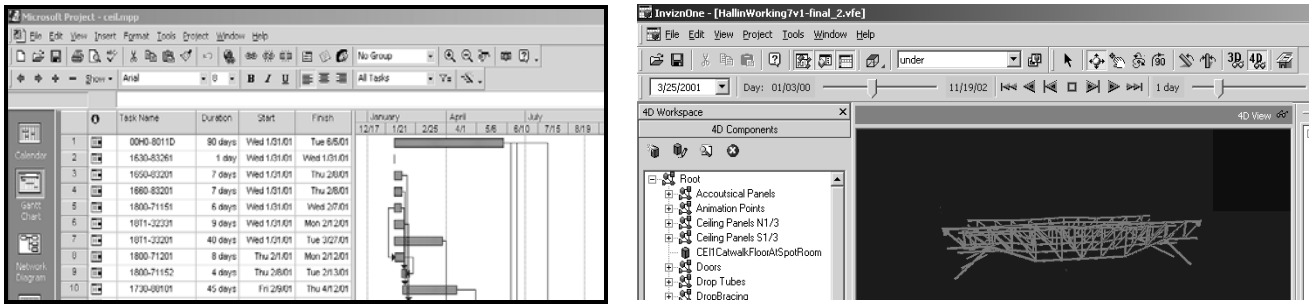



**Figure 9. Revising the Project Schedule via a PDA Device**



**(a) Reviewing the Updated Schedule in Microsoft Project** **(b) Reviewing Updated Project on 4D Viewer**

**Figure 10. Reviewing Updated Project Schedule and Project Model**

Another scenario example is to bring on-line services to engineering simulation. Figure 11 shows the example workflow to include weather conditions to project management applications. A parser is developed to convert the weather forecast service information into XML format as shown in Figure 12. In addition, a simulation access language[7] has been designed to allow an user to specify the workflow and to embed the program code in Microsoft Excel as depicted in Figure 13. Figures 14 and 15 illustrate the impacts of the weather conditions to the schedules displayed in Primavera P3 and the results of task and resource backlogs generated by Vite SimVision and displayed using Excel as charts.

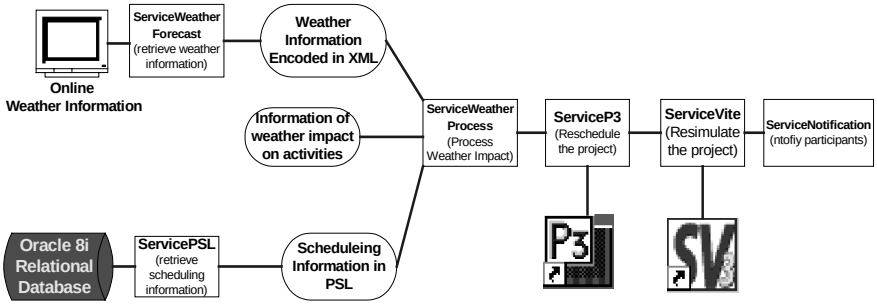
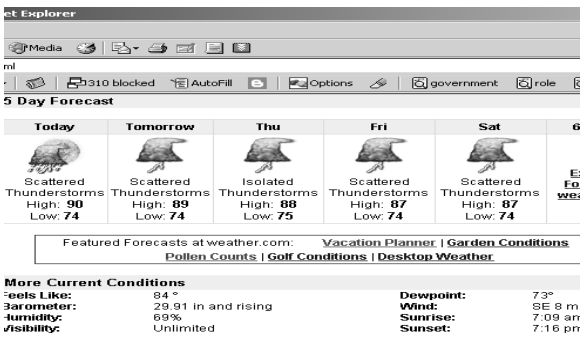
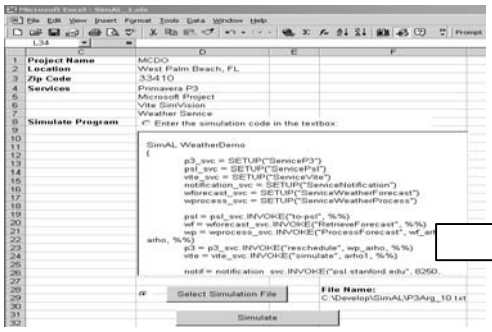



**Figure 11: The Workflow in the Weather Demonstration**



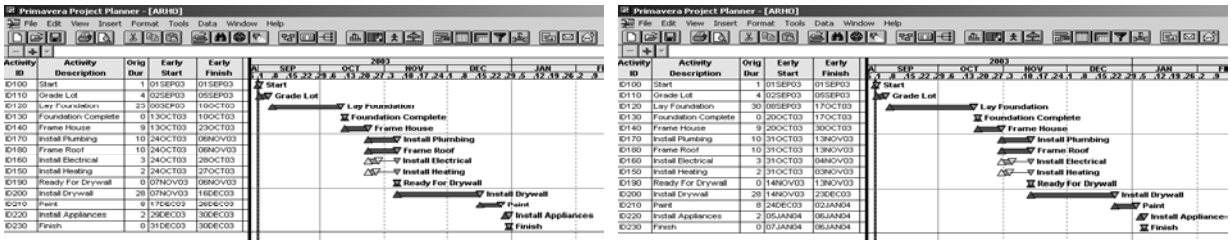

```
<?xml version="1.0"?>
<WeatherReport>
<weather date="2003-9-23">
<location>
<zipcode value="33410" />
</location>
<conditions value=" Isolated thunderstorms early, mainly cloudy overnight
with a few showers" />
<temperature>
<templow c="23.3" f="74.0" />
<temphigh c="32.2" f="90.0" />
</temperature>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

**Figure 12: Expressing Weather Information in XML files**



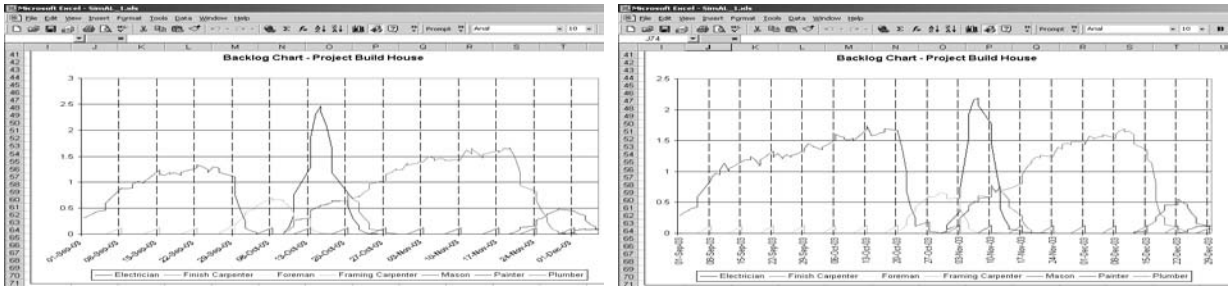
```
SimAL WeatherDemo
{ /* Establish Connections */
  p3_svc = SETUP("ServiceP3")
  psl_svc = SETUP("ServicePsl")
  vite_svc = SETUP("ServiceVite")
  notification_svc =   SETUP("ServiceNotification")
  wforecast_svc = SETUP("ServiceWeatherForecast")
  wprocess_svc = SETUP("ServiceWeatherProcess")
  /* Invoke Services */
  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 13: A Demonstration SIMAL Program**



(a) **Original Schedule in Primavera P3** **(b) Updated Schedule in Primavera P3**

**Figure 14: The Impact of Weather on the Schedule Displayed in Primavera P3**




**(a) Original Backlogs Displayed in Excel** **(b) Updated Backlogs Displayed in Excel**

**Figure 15: The Impact of Weather on Task Backlog Displayed in Charts**

## 4. SUMMARY AND DISCUSSIONS

This paper has presented three example applications that demonstrate the potential applicabilility and flexibility of the web services technology. Although the mechanisms and protocols that these applications employ to achieve distributed computing are different, the approach that each of them takes is more or less similar. Integrating distributed engineering applications as web services provides an effective mechanism to make legacy applications accessible to a broader group of users. Developers can collaborate to build sophisticated engineering applications by developing distributed modules that provide portions of the desired functionalities. With the flexibility and scalibility of the web services technology, the impact could have significant impact not only in project management but life cycle operations and management.

Web service is still an emerging technology, and many improvements need to be made. First, many general-purpose features, such as security, reliable message delivery, and transactional semantics, are needed to facilitate the development of web services. Second, specifications for the web services need to be standardized. The lifecycle of the specifications typically progresses from proposal to *de facto* standard to actual standard. While researchers are continuing to propose new specifications, standardization is required for adoption. Third, toolkits and frameworks for developing web services need to be improved. To promote scalability in the integration of web services, programming models need to shift from procedural call style services toward specification-centric services. As these new developments progress, the web services technology will be applied more widely for developing engineering software applications.

## ACKNOWLEDGEMENTS

This work has been supported in part by NSF Grant Number EIA-9988998, the Pacific Earthquake Engineering Research Center through the NSF under Award number EEC-9701568, the Center for Integrated Facility Engineering at Stanford University, and the Product Engineering Program at National Institute of Standards and Technology.

## REFERENCES

1. *Americans with Disabilities Act Accessibility Guide.* Access Board, U.S. Architectural and Transportation Barriers Compliance Board, Washington, D.C., 1997.
2. Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D. Thatte, S. Trickovic, I., and Weerawarana, S., *Specification: Business Process Execution Language for Web Services (BPEL4WS)*, Version 1.1., http://www-106.ibm.com/developerworks/library/ws-bpel/, 2003
3. Ankolekar A, Burstein M, Hobbs JR, Lassila O, Martin DL, McIlraith SA, et al., "DAML-S: Semantic Markup for Web services," *Proceedings of the International Semantic Web Working Symposium*, Stanford, CA, pp. 411-430, 2001.
4. Beringer, D., Tornabene, C., Jain, P., and Wiederhold, G., "A Language and System for Composing Autonomous, Heterogeneous and Distributed Megamodules," *DEXA International Workshop on Large-Scale Software Composition*, Vienna, Austria, 1998.
5. Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, H. F., Thatte, S., and Winer, D. *Simple Object Access Protocol (SOAP)*, Version 1.1, W3C Note, http://www.w3.org/TR/SOAP, 2000.
6. Cheng, J., Gruninger, M., Sriram, R.D., and Law, K.H., "Process Specification Language for Project Scheduling Information Exchange," *International Journal of IT in Architecture, Engineering and Construction*, 4:307-328, 2003.
7. Cheng, J.. *A Simulation Access Language and Framework with Applications to Project Management*, Ph.D. Thesis, Department of Civil and Environmental Engineering, Stanford University, Stanford, CA, 2004.
8. Eddon, G., and Eddon, H., *Inside Distributed COM*, 1st Ed., Microsoft Press, Redmond, WA, 1998.
9. Han, C. S., Kunz, J. C., and Law, K. H., "Building Design Services in a Distributed Architecture," *ASCE Journal of Computing in Civil Engineering*, 13(1):12-22, 1999.
10. International Alliance for Interoperability, *Industry Foundation Classes*. Specification Volumes 1-4, , Washington, DC, 1997.
11. International Organization for Standardization, *Product Data Representation and Exchange: Part 1: Overview and Fundamental Principles*, No. 10303-1, 1994.
12. Leymann, F., *Web Services Flow Language (WSFL)*, Version 1.0, IBM Corporation, http://www-306.ibm.com/software/solutions/webservices/pdf/WSFL.pdf , 2001.
13. Liu, D., Law, K. H., and Wiederhold, G., "Data-flow Distribution in FICAS Service Composition Infrastructure," *Proceedings of the 15th International Conference on Parallel and Distributed Computing Systems*, Louisville, KY., 2002.
14. Liu, D., Cheng, J., Law, K. H., Wiederhold, G., and Sriram, R. D., "An Engineering Information Service Infrastructure for Ubiquitous Computing," *ASCE Journal of Computing in Civil Engineering*, 17(4):219-229, 2003.
15. McKenna, F., *Object Oriented Finite Element Analysis: Frameworks for Analysis Algorithms and Parallel Computing*, Ph.D. Thesis, Department of Civil and Environmental Engineering, University of California, Berkeley, CA., 1997.
16. Peng, J., and Law, K. H. "A Prototype Software Framework for Internet-Enabled Collaborative Development of a Structural Analysis Program," *Engineering with Computers*, 18(1):38-49. 2002.
17. J. Peng and K. H. Law. "Building Finite Element Analysis Programs in Distributed Services Environment," *Computers & Structures*, 82(22):1813-1833, 2004.
18. Pitt, E. and McNiff, K., *Java™.RMI: The Remote Method Invocation Guide*, 1st Ed.., Addison-Wesley, 2001.
19. Pope, A., *The CORBA Reference Guide: Understanding the Common Object-Request Broker Architecture*, 1st Ed., Addison-Wesley, 1998.
20. Roy, J., and Ramanujan, A., "Understanding Web Services," *IT Professional*, 3(6):69-73, 2001.
21. Schlenoff, C., Gruninger, M., and Ciocoiu, M., "The Essence of the Process Specification Language." *Transactions of the Society for Computer Simulation*, 16(4):204-216, 1999.
22. Young, M.J., *Step by Step XML*. Microsoft Press, 2001.