International Symposium on Innovation & Sustainability of Structures in Civil Engineering
Xiamen University, China, 2011

WEB SERVICES IN CIVIL AND STRUCTURAL ENGINEERING SIMULATIONS

Kincho H. Law
Department of Civil and Environmental Engineering, Stanford University, Stanford, CA 94305-4020, United States of America

Abstract: Civil and structural engineering have had a long and successful history in adopting computing technologies, from computer graphics, CAD, finite element analysis to virtual simulations. As computer hardware and software technologies continue to advance, demands for functionalities of engineering software have also grown. Computer programs have become ever more sophisticated and complex. The desire for integrated solution in the engineering industry had led to a fundamental shift in software development process from focusing on programming in the early days of computing towards focusing on software integration. The tremendous advancements in network and communication technologies, especially with the emergence of the Internet, have not only revolutionized our way of life but also significantly impacted engineering practices and business transactions. Serving as a communication infrastructure, the Internet allows easy access to distributed software services residing in multiple locations. The computing environment has been transformed from stand-alone desktop application to an interconnected web of autonomous services accessible inside and outside the boundary of an organization. In general terms, a web service can be described as a specific function that is delivered over the Internet to perform a service or to provide some specific information. The web service model has become a favorite approach for integrating software applications and extending the functionalities of an application by making it interoperable with other services. This paper presents selected examples, including finite element computing, project management and model-based CAD simulations, to illustrate the potential use of the Internet infrastructure to support interoperation between software applications.
Keywords: Internet Computing, Web Services, Finite Element Analysis, CAD, Engineering Simulation

1 INTRODUCTION

As demands for functionalities increase, software has become ever more complex. Traditionally, software projects are partitioned into subtasks or program modules which are then encoded in some programming languages. The program modules are subsequently grouped and integrated into a software package. Programming languages, utilities and software paradigms have been developed over the years to facilitate the integration process. As more and more software modules and application programs, each has its own functionalities, are developed and packaged, increasingly software development, as illustrated in Figure 1, has shifted from coding as the primary task to a focus on integration (Beringer et al. 1998). This trend is of no exception in civil engineering. As stand-alone engineering applications (such as finite element analysis, CAD, project management, etc..) of the 1960s and 1970s began to mature, the industry has been pushing for shared or integrated platforms, extending domain-specific tools and capabilities beyond an individual application to support collaborative project-wide activities, ranging from planning, design, construction to management.

Advances in network and communication technologies, especially with the emergence of the Internet, have not only revolutionized our way of life but also have significantly impacted engineering and business practices. Users of information systems now have tremendous sources of services available on the web, such as travel reservation, book purchasing, weather forecasts, financial data summaries, newsgathering, customer relation management, and many others. Serving as a communication infrastructure connecting computers and devices anywhere and anytime, services can be integrated over the Internet, thus supporting application-to-application and organization-to-organization cooperation. Internet allows easy access to software services residing in

geographically distributed locations. The computing environment has transformed from stand-alone desktop application to an interconnected web of autonomous services accessible inside and outside the boundary of an organization. The concept that distributed software services can be accessed and collaborate over the Internet has now become a reality (Han et al. 1998, 1999). In general terms, a web service can be described as a specific function that is delivered over the Internet to perform a service or to provide some specific information. The web service model has become a favorite approach for integrating software applications and extending the functionalities of an application by making it interoperable with other 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, design and project management. Additionally, online information can be dynamically integrated with the applications. The simplicity of the web service model makes it possible to build and integrate engineering simulation tools collaborating via the Internet. This paper discusses web service models for software integration and presents a few examples to illustrate the potential applications of web services in civil and structural engineering.

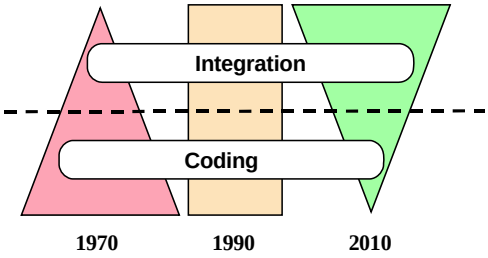


Figure 1. Trend of Software Development (Courtesy of Prof. Gio Wiederhold, Stanford University)

2 SOFTWARE INTEGRATION AND WEB SERVICES

Software integration can take many forms. In general terms, the integration can be categorized as tightly coupled or loosely coupled. For tightly coupled software integration, code reuse is the most common approach that the source code is simply inserted to wherever the desired functionality is needed. Another approach is to establish shared libraries of software components, which are typically written in the same programming language. Shared libraries typically have well-defined public interfaces, through which the users can invoke the functions contained in the libraries. Each software component or task is executed as a separate process and communicating with other tasks through exchanging data or messages. Tightly coupled software components are typically managed by a single administrative entity.

Integrating loosely coupled software applications, particularly those belong to multiple administrative authorities, would require sharing data common to the applications. Since 1980's, as domain specific applications, such as CAD and project management tools, become widely available, much efforts have been spent toward integrating the standalone desk-top applications through information sharing among engineering tools (Eastman 1999). One common approach is the use of neutral files or standard data exchange formats. Data exchange standards have evolved from graphic and geometric entities, such as IGES (1983) to product models, such as STEPS (ISO 1991). The early work of data exchange standards focused on exchange of data within similar domains. With the development of standards and technology for product and parametric CAD models, the industry began to push for interoperability among loosely coupled software applications, allowing the data from one application to be shared by another software. In the civil engineering domain, the development of industry foundation classes (IFC 1997) has now led to the establishment of building information modeling (NBIM 2007) that includes the description of a facility and its product components as well as the process and project information. Although the concept of building information modeling is not new, the recent advancement in computer hardware and software technology has made practical deployment of BIM a reality (Eastman et.al. 2008).

In parallel to the development of information models, there have also been tremendous advancements in network and communication technologies. Network computing has allowed software services to be distributed and executed on multiple locations. As communication protocols evolved from low level specifications, such as TCP/IP and RPC (Birrell and Nelson 1984), to the high level object-oriented schemes, such as COBRA (Pope 1998), SOAP (W3C 2000), Java RMI (Pitt and McNiff 2001), implementation of distributed services has been greatly facilitated. Web-based information standards, such as XML (Young 2001), RDF (Brickley and Guha 1998), and OWL (Dean et.al. 2002), and web-based description languages, such as WSDL (W3C 2004) and BPEL (Andrews et.al. 2003), have been developed to facilitate the description, reuse and integration of web services. The web service model has emerged as a favorite approach for integrating software applications and extending the functionalities of an application by making it interoperable with other services. An engineering simulation can now involve a number of geographically distributed software applications. Online information can be dynamically integrated with an application. In short, the Internet infrastructure can be leveraged to support interoperation between applications and collaboration between organizations.

Our research interest in web services has been focused on developing methodologies that can effectively wrap legacy applications and make them accessible and interoperable with each other via the Internet. Different applications require different models for integration and coordination. The following sections describe a number of demonstrative examples in civil and structural engineering to illustrate the methodologies in developing engineering web services and the integration of distributed web services for engineering simulations.

3 TIGHTLY COUPLED SOFTWARE INTEGRATION -- A WEB SERVICE FRAMEWORK FOR FINITE ELEMENT ANALYSIS

Current structural finite element analysis (FEA) programs are “monolithic” in that they are typically implemented and run on a dedicated computer. Typically, a FEA program bundles all the procedures and program kernels that are developed by an individual organization. As theories continue to advance, FEA programs need to be able to accommodate new developments in element formulation, solution algorithms, and analysis strategies. The desire to develop and maintain large complex software systems in a dynamic and constantly evolving environment has driven interest in new approaches to FEA software design and development. Object-oriented design principles and programming have been utilized in FEA software development to support better data encapsulation and to facilitate code reuse (Miller 1991, Mackie 1992, Zimmermann et al 1992, Archer 1996). However, most existing object-oriented FEA programs are still rigidly structured. Extending and upgrading these programs to incorporate new developments and legacy codes remain to be a tedious and difficult task. Moreover, there is no easy way to access computing resources and structural analysis services distributed in remote sites. This section discusses a research prototype implementation of an Internet-enabled software framework that allows integrating remotely available finite element services (Peng 2002, Peng and Law 2002).

We adopted the web service model for the development of an Internet-enabled framework for FEA to facilitate software integration. Following the object-oriented design paradigm, many modules of an FEA program can be encapsulated and developed as separate software services, which in turn are integrated with the framework through a set of pre-defined protocols (Plasil et al 1999). The design of the framework focused on the interactions among the services and the system was built to provide a *plug-and-play* character (Carney and Oberndorf 1997).

As shown in Figure 2, the prototype Internet-enabled finite element structural analysis framework consists of the following modules (Peng 2002):

- The **Analysis Core** module, which consists of a set of basic functional units of a finite element structural analysis program. Element and material models, solvers, as well as analysis strategies and solution procedures, are included in this module.

- The **User-Interaction Interface** module, which provides an interface to facilitate the access to the software platform by both the users and the developers. The platform can be accessed from either a web (browser) interface or other application programs such as Matlab and Excel.
- The **Registration and Naming Service**, which allows online application services to register to the core so that these services can be found and accessed. Users can obtain references to the online application services by querying the Registration and Naming Service.
- **Element Services**, which enable integration of external finite elements with the analysis core. Two approaches are provided for remote access to element services. The **Distributed Element Service** provides a communication link to remote element services where the element code is executed. The **Dynamic Linked Element Service** dynamically binds the element code with the core at runtime.
- The **Project and Data Service**, which is built to link with a database system, which provides persistent data storage and efficient data access, and to facilitate post-processing tasks. Project management and version control are also supported by the project and data service.

The component-based modular architecture is designed to support interactions and collaborations among users, developers and external applications.

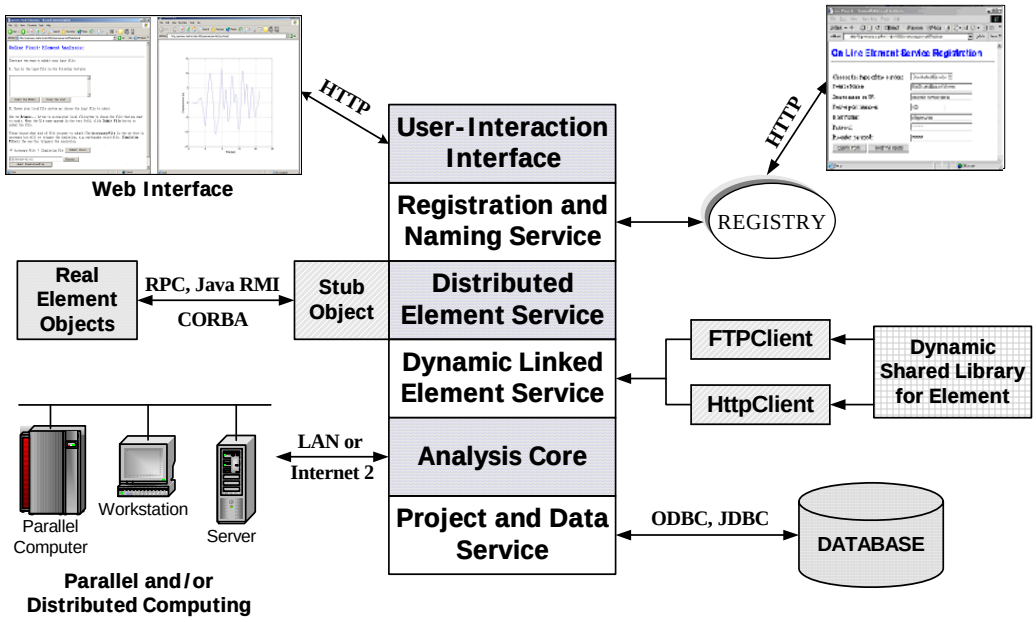


Figure 2. Services of the Internet-Enabled Finite Element Analysis Framework

The *Analysis Core* module is built upon an object-oriented finite element program, OpenSees (Open System for Earthquake Engineering Simulation) (McKenna 1997). In this framework, the core model, in this case OpenSees, runs on a central server as a compute engine to provide finite element analysis service. A compute engine is a remote software program that takes a set of tasks from clients, runs them, and returns the result. Parallel and distributed computing environment, if available, can be employed to improve the system performance and makes it more suitable for large-scale engineering simulations (Mackey and Law 1996, De Santiago and Law 2000). New element and material technologies as well as new analysis strategies and solution algorithms can be directly incorporated into the core to improve the capabilities and performance of the server. The user needs only know the location of the core server, which could be installed on his/her own desktop computer or in a remote site.

Our prototype implementation focused on the development of element services to demonstrate the web service model for finite element analysis. For a finite element program, element code is a natural candidate for building it as a service. First, a standard interface/wrapper is developed for communicating the element with

the object-oriented analysis core. Typical for an object-oriented FEA software, such as OpenSees, introducing new elements into the analysis core can be achieved by creating a subclass to the generic *Element* class whose common interface is defined in the analysis kernel (McKenna and Fenves 2000). Instead of directly encoding a new element into the finite element (library) core as traditionally done, the new element can be developed in the form of an online element service. As shown in Figure 2, developer can register the element service through an online registration and naming service. Since the registry keeps track of the modifications and versioning of the element services, an element service can be easily replaced or upgraded.

The most common style in the implementation of web services is either using remote procedure call by invoking the distributed functions (or methods) or treating software applications as services by communicating via messages. Remote procedure calls are particularly useful in implementing "tightly" coupled services (as, in this case, integrating element service with the finite element analysis core) where the services are directly mapped to language-specific functions or method calls. In the prototype implementation, Java's Remote Method Invocation (RMI) is selected to handle communication for the distributed element services over the Internet (Pitt and McNiff 2001). Additionally, since the original FEA core program is written in C++, and partly written in C and Fortran (referred to as *native language* in Java terminology), communication support between Java and other languages is enabled using the Java Native Interface (JNI) (Liang 1999).

As an alternative to the distributed service model, an element service can also be built as a dynamic shared library and placed on an FTP or an HTTP server on the element service provider's site. During the system runtime, the installed shared library can be automatically downloaded to the core server and linked with the FEA core. The shared element code in binary format is loaded at runtime, so that different services can be replaced at runtime without re-compilation and re-linking with the application. In order to support dynamic loading and binding, in most cases the shared library must be built using the same platform as the core server.

The mechanics of the Internet-enabled collaborative finite element analysis framework is illustrated in Figure 3. The user first builds the structural model using a web-based model-building service on the client site. The model is then sent to the analysis core. The core server authenticates the user's identity and starts the analysis on the received model. During the analysis, elements that are available in the core are accessed locally from the static element library, whereas other elements are obtained from online element services. For elements that are not available in the core, the *registry* is queried to find the on-line element services pre-registered by the element developer. Depending on the service model registered, finite element matrices are generated either through method invocation to a distributed element service or dynamically downloading and binding the shared element during runtime. After the analysis is completed, the result is returned to the user by generating a dynamic web page in the user's web browser and/or via other user-defined (such as Matlab) interface.

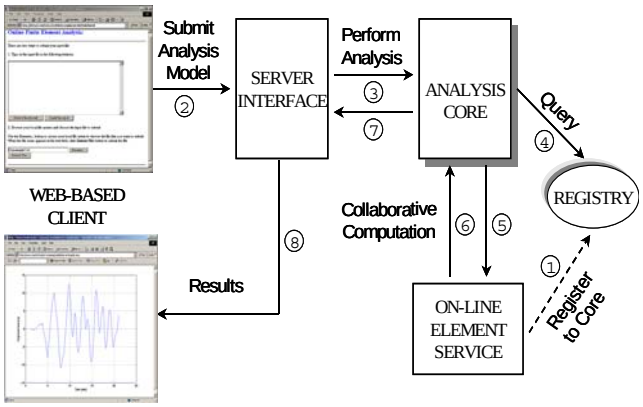
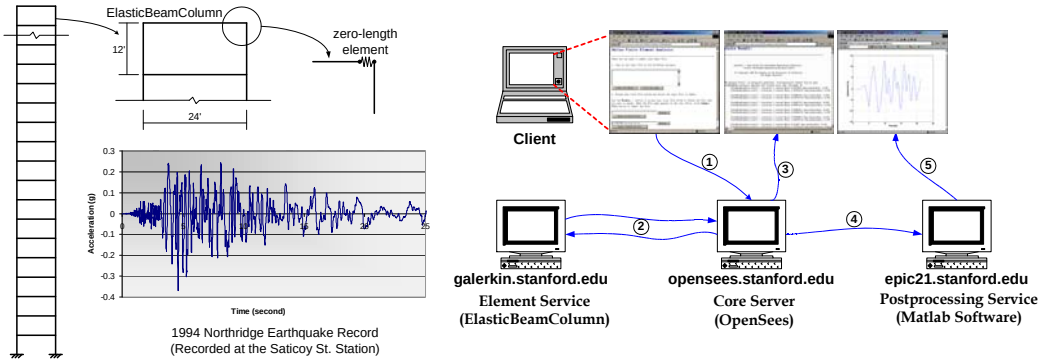


Figure 3. Mechanics of the Internet-Enabled Element Service Framework

To illustrate the element service models, Figure 4(a) shows a model of an 18-story one bay frame structure. All the beams and columns are modeled as *ElasticBeamColumn* elements where hinging is modeled with zero-length elasto-plastic rotational element. Nonlinear dynamic analysis is performed on the model subjected to the 1994 Northridge earthquake record using the Newton-Raphson analysis scheme. As depicted

in Figure 4(b), the *ElasticBeamColumn* element is built as an element service residing on galerkin.stanford.edu, a Sun's Ultra workstation. Furthermore, the analysis core server and the post-processing service are implemented on two separate Sun's Ultra workstations, named opensees.stanford.edu and epic21.stanford.edu, respectively. As noted earlier, the user only needs to know the location of the analysis core server for job submission. The element service will be automatically executed depending on the location and type of service. The results are delivered to the client's workstation and a post-processing service can be invoked to display the results using a web browser or a Matlab interface.



(a) Example Structural Model (b) Interaction of Distributed Services

Figure 4. Illustrative Example of Internet-Enabled Finite Element Service Model

To compare the performance of the different element service models, Table 1 lists the total time required for the simulation. Clearly invoking the distributed element service for each element calculation imposes significant overhead due to initialization cost for each invocation, network latency and the JNI interface. One possibility to improve the performance is to reduce remote method initialization and network latency by bundling the calculations for all elements for each time step and storing the element data in a buffer before transmitting to the analysis core. When comparing with the static element library where the element is hard coded with the analysis core, the shared library element service also incurs additional overhead associated with the downloading and the runtime binding of the library file from the remote element service. It should be pointed out that the percentage of overhead decreases as the number of time steps increases. Since the registry service keeps track of the versioning of the element library, one possibility is to pre-caching the library file so that, if there is no version change on the element service, the file can be just directly loaded and bound with the analysis core without any network overhead. Nonetheless, this Internet-enabled framework has demonstrated the flexibility for integrating third party element developments as a web service and its ability to expand the functionalities of a traditional finite element analysis program beyond its own capabilities.

Table 1. System Performance using Different Element Services

Time steps	Static Element Service	Distributed Element Service	Shared Element Service
50	4.25 s	108.06 s	8.91 s
300	38.24 s	856.95 s	51.53 s
1500	204.47 s	6245.6 s	284.82 s

4 LOOSELY COUPLED SOFTWARE INTEGRATION – APPLICATIONS AS WEB SERVICES

In the engineering and construction domain, many stand-alone applications (e.g., Microsoft Project, Microsoft Excel, Primavera Project Planner, and AutoCAD) are widely used. These tools are designed for specific functions and generate large volumes of information that are not easily shared among the applications. Even though many online services, such as weather forecasting, product catalogues, finance reports, etc., now exist,

these services are not readily integrated with traditional standalone applications. The functionalities of an standalone application can be greatly extended if it can be integrated and collaborating with other tools. As opposed to the tightly software integration model used for finite element software, besides treating the software application as a web-based service, the key issue for integrating and coordinating of loosely coupled applications for simulation is data and software interoperability.

4.1 A WEB-ENABLED PROJECT MANAGEMENT SIMULATION FRAMEWORK

The need for efficient data exchange in computer aided design has been a subject of research and development for quite some time (Eastman 1999). Current standards, such as STEP (ISO 1991) and IFC (1997), focus on the definition and exchange of product data. An engineering project, however, involves not only the descriptions of the component objects that make up the design but also the activities, organization, goals, and constraints that constitute the processes required to describe the workflow and enterprise models. In order to exchange process information, a standard model or language is needed to describe the processes and to support interoperability among the process oriented applications. Furthermore, to integrate and coordinate standalone applications and to conduct simulation of project scenarios without the detailed knowledge of the network communication and application software, a high level simulation language and a software composition framework are needed.

Our prototype web service simulation framework builds upon a software composition infrastructure based on data and control flows (Liu et al. 2002, Liu 2003). The Flow-based Infrastructure for Composing Autonomous Services (FICAS) (Liu 2003) is developed to invoke distributed services and to direct data flow among the different services. 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. Wrappers are developed with information exchange protocols based on Extensible Markup Language (XML) (Young 2001) and the Process Specification Language (PSL) (Menzel and Gruninger 2001, Schlenoff 1999). PSL, developed at the National Institute of Standards and Technology originally to facilitate correct and complete exchange of process information among manufacturing systems, is extended to handle the terminologies and functions that are commonly found in construction management (Cheng et al. 2003).

In order to describe the usage of web services, a simple, easy-to-use simulation access language, SimAL, is designed and implemented to coordinate application tools and to simulate scenarios in assisting decision making (Cheng 2004). In general, three key factors are involved in decision-making: alternatives, information, and preferences. Alternatives imply that more than one option may be available. Information refers to the knowledge available to users about the different options. Preferences specify the aspects that users want to optimize. To support these functions, operational statements for Invocation, Operation, Control, and Decision-Support are provided in SimAL (Cheng 2004).

A scenario example is shown here to demonstrate the potential applications of web services in project management simulation. Commerical application software, such as Microsoft Excel, Primavera Project Planner (P3) and Vite SimVision, are wrapped as web services. The web service applications run on heterogeneous, distributed platforms and are accessed homogeneously through standard web service interface provided by FICAS. Additionally, this scenario example illustrates the possibility in bringing on-line services to engineering simulation. Figure 5 shows an example workflow to include weather conditions in project management. A parser is developed to convert the weather forecast service information into XML format as shown in Figure 6. As depicted in Figure 7, a simulation program written using SimAL is embedded in Microsoft Excel to specify the workflow. Figures 8 and 9 show 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 as charts using Microsoft Excel. This example illustrated that, by using a web service model to develop an integration framework, loosely coupled engineering applications can interoperate regardless of locations and platforms.

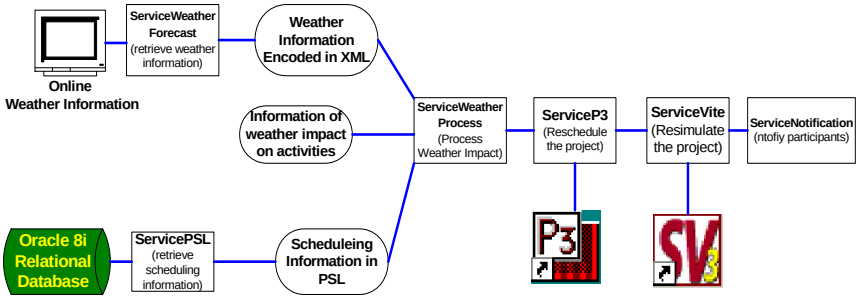
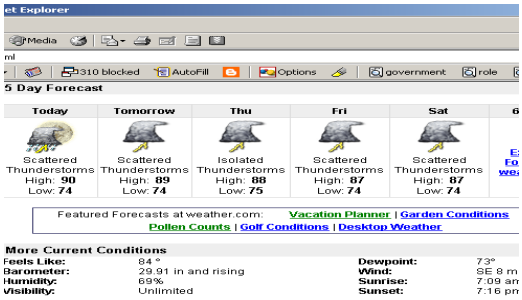
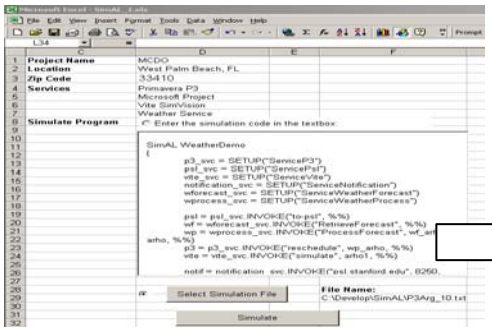


Figure 5. The Workflow in the Weather Demonstration



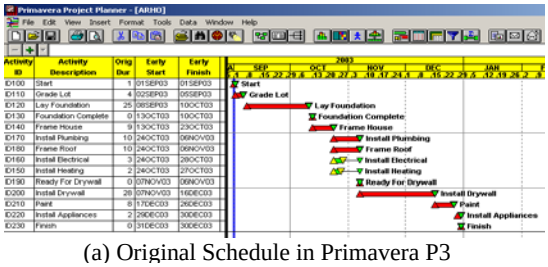

```
<?xml version="1.0"?>
<WeatherReport>
<weather date="2003-9-23">
<location>
<zipcode value="33410" />
</location>
<conditions value=" Isolated thunderstorms early, mainly cloudy overnight with a few showers" />
<temperature>
<templow c="23.3" f="74.0" />
<temphigh c="32.2" f="90.0" />
</temperature>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

Figure 6. Expressing Weather Information in XML

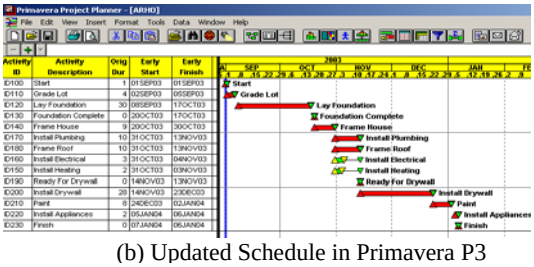



```
SimAL WeatherDemo
{ /* Establish Connections */
  p3_svc = SETUP("ServiceP3")
  psl_svc = SETUP("ServicePsl")
  vite_svc = SETUP("ServiceVite")
  notification_svc =      SETUP("ServiceNotification")
  wforecast_svc = SETUP("ServiceWeatherForecast")
  wprocess_svc = SETUP("ServiceWeatherProcess")
  /* 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 7. A Demonstration SimAL Program Embedded in Excel



(a) Original Schedule in Primavera P3



(b) Updated Schedule in Primavera P3

Figure 8. The Impact of Weather on the Schedule Displayed in Primavera P3

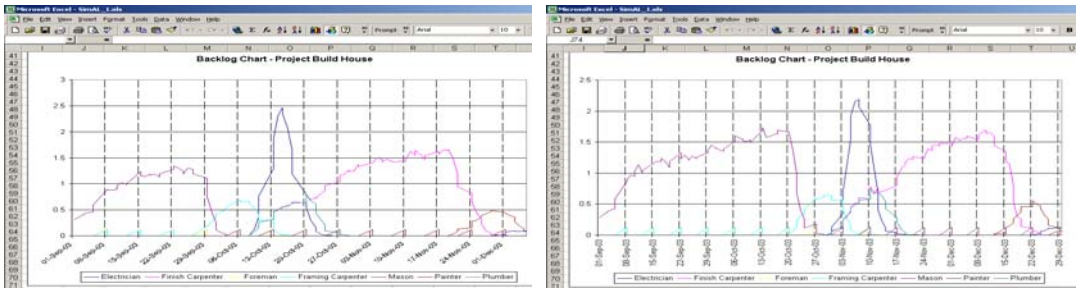


Figure 9. The Impact of Weather on Task Backlog Displayed as Charts in Excel

4.2 A WEB-ENABLED MODEL-BASED CAD SIMULATION FRAMEWORK

Computer-aided design (CAD) applications have been widely used in civil engineering. There have been growing interests in developing online catalogs that can integrate and interact with design tools (see, for example, http://smartbim.reedconstructiondata.com/). The ability to source design components from different vendors on the web and incorporate them in developing a design solution will greatly impact the way design is developed and will create many business opportunities (Cheng 2009). With the emerging drag-and-drop technology, an example scenario is presented to illustrate a web-enabled model-based CAD environment that can enhance engineering design and simulation process.

Traditionally, a design is drawn using a combination of geometric entities such as lines, areas, and volumes. This geometry-based approach has evolved to an object-oriented model-based approach, in which each design object component (e.g. door, wall, and slab) has its own properties and semantics (Eastman et al., 2008). By capturing the component object information, such as product data, supplier information, and schedule information, in a database or inside the CAD drawings, a model-based CAD environment can facilitate decision making, production of construction documents, prediction of building performance, cost estimating, and construction planning (Cheng et al 2010). Therefore, a CAD application is no longer just a drawing tool but becomes a service hub for acquiring, capturing and managing design and project information.

A web-enabled CAD environment is a setting that enables users to incorporate, interact and integrate online information and/or services within the CAD system, therefore extending the scopes and functionalities of the system. The interaction between the users and the web service providers has been greatly facilitated by the drag-and-drop technology recently developed by CAD vendors and as a feature in the proposed HTML5 standards (see http://dev.w3.org/html5/html4-differences/#apis). The technology allows a user to “drag” content from the web and to “drop” the information directly into desktop programs and files or another web service.

Our example scenario builds on SpecifiCAD developed by CADalytic, Inc., a plug-in tool that dynamically matches user-defined building product content with online product data in a CAD environment. As illustrated in Figure 10, when a designer clicks a window in Sketchup and selects the “catalog” option in SpecifiCAD, a list of window alternatives is shown. In fact, since the component name contains the word “window,” the program searches within the extranet all the windows from partnering suppliers and obtains the product and contact information through the suppliers’ web services. The window object can be “dragged-and-dropped” into the CAD drawing. The attributes of the component, such as cost, supplier information, and properties, are updated automatically in the CAD model.

Besides geometry, a model-based CAD environment captures building component information which can be utilized for simulation and analysis of building performance. In this example, Integrated Environmental Solutions (IES) Virtual Environment is used to simulate and evaluate the energy consumption and carbon emissions of a building within Google Sketchup. Before simulation, IES Virtual Environment extracts the information from the Sketchup drawing and identifies individual rooms (Figure 11). Room types such as “dining area” and “bedroom” can be defined using the IES plug-in. The size and property definitions of the rooms and of the components such as windows and doors are then transferred to IES Virtual Environment. With reference to local climate data, the program estimates the annual carbon emissions and energy use.

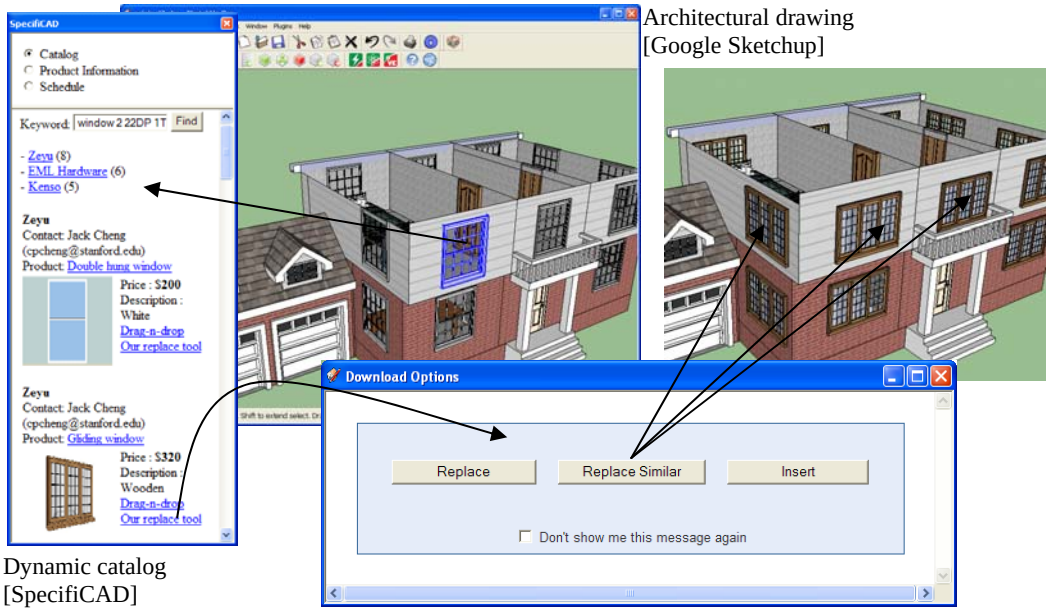
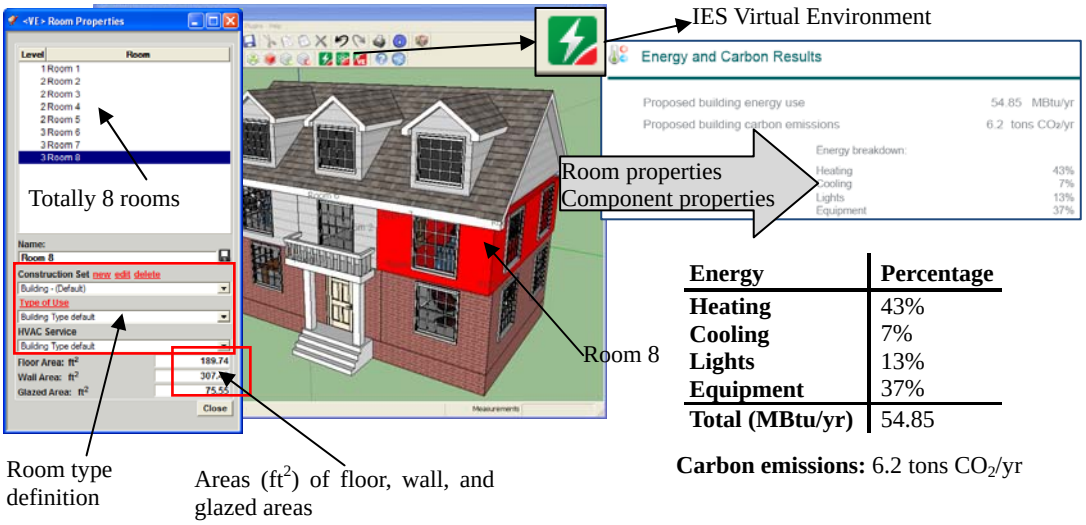


Figure 10. Interaction between CAD object components and dynamic online information.



Energy	Percentage
Heating	43%
Cooling	7%
Lights	13%
Equipment	37%
Total (MBtu/yr)	54.85

Carbon emissions: 6.2 tons CO_2/yr

Figure11. Energy and carbon emissions analysis of a building.

The web-enabled configuration design environment described can be used to facilitate design generation. Designers can conveniently modify a design and evaluate its building performance before procurement. Design configurations can be easily changed and evaluated since building components are readily accessible on the web. Last but not least, since the supplier information is captured in the model-based CAD model during the design process, procurement and supply chain processes can be integrated with the model-based CAD environment (Cheng 2009).

5 CONCLUSIONS

Since its inception two decades ago, the web, which utilizes the Internet as communication infrastructure, has undergone explosive growth in services and types of services it provides. By treating software components and applications as web services accessible via the Internet, the functionalities and capabilities of each individual application service can be extended by making it interoperable and integrated with other application services. We discuss the possibility of integrating into a finite element software framework the element services provided by third party developers. By wrapping legacy applications as web services using standard web protocols and information models, engineering tools can be orchestrated to provide mega-services supporting workflow and supply chain management. Online information can be dynamically incorporated into web services to facilitate engineering design and project management simulations. As the Internet infrastructure continues to grow and web services continue to expand, there will be abundant opportunities to develop innovative uses of the technology and create new businesses. This paper reviews some of the attempts to illustrate the development of web-based application services in the civil and structural engineering domain.

ACKNOWLEDGEMENTS

The author would like to gratefully acknowledge his former students, Drs. Charles Han, Jun Peng, David Liu, Jinxing Cheng and Jack Cheng, for their research contributions on engineering web services and Internet computing. The research described has been partially supported by the National Science Foundation, Pacific Earthquake Engineering Research Center, the National Institute of Standards and Technology and the Center for Integrated Facility Engineering at Stanford University. No approval or endorsement of any commercial products by NIST, NSF or Stanford University is intended or implied. Last but not least, any opinions are those of the author and do not necessarily reflect the views of NSF, NIST, CIFE, Stanford University or his collaborators.

REFERENCES

Andrews T., et.al. (2003). *Specification: Business Process Execution Language for Web Services (BPEL4WS)*, Version 1.1., http://www-106.ibm.com/developerworks/library/ws-bpel/.

Archer G. C. (1996). *Object-Oriented Finite Element Analysis*, Ph.D. Thesis, Department of Civil and Environmental Engineering, University of California, Berkeley, CA, 1996.

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

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*, 4:307-328.

Birrell A.D. and Nelson B.J. (1984). Implementing Remote Procedure Calls, *ACM Transactions on Computer Systems*. 2(1):39-59.

Brickley D. and Guha R.V., (1999) (editors), *Resource Description Framework (RDF) Schema Specification*, World Wide Web Consortium. (available at http://www.w3.org/TR/1998/WD-rdf-schema).

Carney D.J. and Oberndorf P.A. (1997) The Commandments of COTS: Still Searching for the Promised Land, *CrossTalk* ,10(5):25-30.

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*, 4:307-328.

Cheng J. (2004). *A Simulation Access Language and Framework with Applications to Project Management*, Ph.D. Thesis, Department of Civil and Environmental Engineering, Stanford University, Stanford, CA.

Cheng J.C.P. (2009). *SC Collaborator: A Service Oriented Framework for Construction Supply Chain Collaboration and Monitoring*, Ph.D. Thesis, Department of Civil and Environmental Engineering, Stanford University, Stanford, CA.

Cheng J.C.P., Law K.H., Zhang Y. and Han C.S. (2010). Web-Enabled Model-Based CAD for the Architectural, Engineering and Construction Industry, *1st Int. Conference on Sustainable Urbanization*, Hong Kong.

De Santiago E. and Law K. H. (2000). A Distributed Implementation of an Adaptive Finite Element Method for Fluid Problems, *Computers and Structures*, 74(1), 97-119.
Dean M. et.al. (2002),. *Web Ontology Language (OWL) Reference*, Version 1.0, W3C Organization, (available at http://www.w3.org/TR/2002/WD-owl-ref-20021).
Eastman C.,M., (1999). *Building Product Models*, CRC Press.
Eastman C., Teicholz P., Sacks R. and Liston K. (2008). *BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors*, Wiley.
Han C.S., Kunz J.C. and Law K.H. (1998). Internet-Based Computer-Aided Design: Leveraging Product Model, Distributed Object, and World Wide Web Standards, *Structural Congress*, ASCE, San Francisco, CA.
Han C.S., Kunz J.C. and Law K.H. (1999). Building Design Services in a Distributed Service Architecture, *Journal of Computing in Civil Engineering*, 13(1), 12-22.
IFC (1997). *Industry Foundation Classes*, Release 1.0, Int. Alliance for Interoperability, Washington, D.C..
IGES (1983). *Initial Graphics Exchange Specification*, National Bureau of Standards, Version 2.
ISO (1991). *DIS 10303, Product Data Representation and Exchange*, ISO, TC184/SC4.
Liang S. (1999). *Java Native Interface: Programmer's Guide and Specification*, Addison-Wesley, Boston, MA.
Liu D., Law K. H. and Wiederhold G. (2002). Data-flow Distribution in FICAS Service Composition Infrastructure, *Proceedings of the 15th International Conference on Parallel and Distributed Computing Systems*, Louisville, KY., 2002.
Liu D. (2003). *A Distributed Data Flow Model for Composing Software Services*, PhD Thesis, Department of Electrical Engineering, Stanford University, Stanford, CA.
Mackie R. I. (1992). Object-Oriented Programming of the Finite Element Method, *International Journal for Numerical Methods in Engineering*, 35(2):425-436.
Mackay D.R. and Law K.H. (1996). A Parallel Implementation of a Generalized Lanczos Procedure for Structural Dynamic Analysis, *International Journal of High Speed Computing*, 8(2):171-204.
McKenna F. (1997). *Object-Oriented Finite Element Programming: Frameworks for Analysis, Algorithm and Parallel Computing*, Ph.D. Thesis, University of California at Berkeley, Berkeley, CA.
McKenna F. and Fenves G. L. (2000). *Introducing a New Element into OpenSees*. http://opensees.berkeley.edu/OpenSees/NewElement.pdf
Menzel C. and Gruninger M. (2001). A Formal Foundation for Process Modeling, *Proceedings of Formal Ontology in Information Systems*, Ogunquit, Maine, pp.256-269, 2001.
Miller G. R. (1991). An Object-Oriented Approach to Structural Analysis and Design, *Computers and Structures*, 40(1):75-82.
NBIM (2007), *National Building Information Modeling Standard Part-1: Overview, Principles and Methodologies*, National Institute of Building Sciences (available at http://www.wbdg.org/bim/nbims.php).
Peng J. (2002). *An Internet-Enabled Software Framework for the Collaborative Development of a Structural Analysis Program*, Ph.D. Thesis, Department of Civil and Environmental Engineering, Stanford University, Stanford, CA.
Peng J. and Law K. H. (2002). A Prototype Software Framework for Internet-Enabled Collaborative Development of a Structural Analysis Program, *Engineering with Computers*, 18(1):38-49.
Pitt E. and McNiff K. (2001). *Java(tm).RMI: The Remote Method Invocation Guide*, Addison-Wesley.
Plasil F., Visnovsky S. and Besta M. (1999). Bounding Component Behavior via Protocols, *Proceedings of TOOLS USA 1999: the 30th International Conference & Exhibition*, Santa Barbara, CA, 1999; 387-398.
Pope A. (1998). *The CORBA Reference Guide*, Addison-Wesley.
Schlenoff C., Gruninger M. and Ciocoiu M. (1999). The Essence of the Process Specification Language, *Transactions of the Society for Computer Simulation*,16(4):204-216.
W3C (2000). *Simple Object Access Protocol (SOAP)*, version 1.1, World Wide Web Consortium.
W3C (2004). *Web Services Description Language (WSDL)*, version 2.0, World Wide Web Consortium.
Young, M.J. (2001). *Step by Step XML*, Microsoft Press.
Zimmermann T., Dubois Pelerin Y. and Bomme P. (1992). Object-Oriented Finite Element Programming: I. Governing Principles, *Computer Methods in Applied Mechanics and Engineering*, 98(2):291-303.