Internet-Enabled Software Model for Nonlinear Structural Analysis and Simulations

Jun Peng, Stanford University, Stanford, CA 94305, USA (junpeng@stanford.edu)
Gloria Lau, Stanford University, Stanford, CA 94305, USA (glau@stanford.edu)
Kincho H. Law, Stanford University, Stanford, CA 94305, USA (law@stanford.edu)

Summary

This paper describes an Internet-enabled software model that could facilitate the development and utilization of nonlinear structural analysis programs. The software model allows users easy access to the analysis core program and the analysis results by using a web-browser or other application programs. In addition, new and legacy codes can be incorporated as distributed services and be integrated with the software framework from disparate sites. A distributed project management system, taking advantages of Internet and database technologies, is implemented to store and manage model information and simulation results. Nonlinear dynamic analysis and simulations of a bridge structure is performed to illustrate the facilities of the Internet-enabled software model.

1 Introduction

Current structural finite element analysis (FEA) programs are "monolithic" in that they are typically implemented and run on a dedicated computer. As technologies and structural theories continue to advance, FEA programs need to be able to accommodate new developments such as element formulation, material relations, analysis algorithms, and solution strategies. The need to develop and maintain large complex software systems in a dynamic environment has driven interest in new approaches to FEA software design and development. Object-oriented design principles and programming have been employed in FEA software development to support better data encapsulation and to facilitate code reuse. 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 difficult task. Furthermore, there is no easy way to access computing resources and structural analysis services distributively located at different remote sites.

With the maturation of information and communication technologies, the concept of building collaborative systems where software services are distributed over the Internet is becoming a reality (Han et al. 1999). Simply defined, web services are a combination of application and data that can be accessed from any Internet-enabled device (Roy and Ramanujan 2001). Using a set of standardized communication protocol, web services can be universally accessed and deployed, independent of the underlying operation environment. The basic idea behind web services is to adopt the loosely coupled web-programming model to facilitate software component integration. The goal is to provide a platform for building distributed applications that utilize software components running on different operating systems and devices. Web services, acting as the adapter for complicated process components and legacy applications, allow disparate systems to work together.

Following the web services model, we have designed and prototyped an Internet-enabled framework to facilitate the usage and development of a nonlinear dynamic structural analysis program (Peng 2002; Peng and Law 2002). In the prototype implementation of the Internet-enabled framework, OpenSees (McKenna 1997) is utilized as the finite element analysis core. OpenSees (Open System for Earthquake Engineering Simulation) is an object-oriented program for seismic simulation of structural and geotechnical systems. The Internet-enabled framework adopts distributed service model to enhance and improve the capability and performance of a FEA program by seamlessly integrating legacy code and new developments. A software component can be incorporated by directly integrating with the core as a local module and/or by implementing as a web service. Using the Internet as a communication channel, the framework provides users the ability to pick and choose the most appropriate methods and software services for solving a problem. The framework also includes project management capabilities. Project management functions allow users to access information about previous analyses of related models stored distributively on different sites. While the prototype implementation is using OpenSees as the FEA core, the framework is sufficiently general to be incorporated by other FEA programs. The framework described herein intends to extend FEA programs, such as OpenSees, to better utilize the Internet as a communication vehicle.

2 Internet-Enabled Services Framework

The distributed engineering service framework is designed to provide developers and users with easy access to an analysis platform and to incorporate new element technologies, new algorithms, and solution strategies with the analysis core. By adopting the services model, FEA program development can shift the focus from coding to the integration of engineering services.

2.1 Architecture of the Services Framework

When integrating multiple computer applications or systems, a suitable architecture is needed to define how the components (applications or systems) are connected within the constraints (Smith and Scherer 1999). In the Internet-enabled software framework, the structural analysis core program is running on a central server as a compute engine. At the heart of the compute engine is a protocol that enables jobs to be submitted to the compute engine, the compute engine to run those jobs, and the results of the job to be returned to the client. This protocol is expressed in interfaces that are supported by both the core server and the distributed services.

To utilize the software platform, users play the role of clients to the central finite element compute engine. The users can have direct or remote access to the core program through a web-based user interface or other application programs, such as MATLAB. The users can specify desirable features and methods (element types, efficient solution methods, and analysis strategies) that have been implemented and registered with the core platform.

For element developers, a standard interface/wrapper is defined for communicating the element(s) with the analysis core. The element code can be written in languages such as Fortran, C, C++ and/or Java as long as it conforms to the standard interface, which is a set of pre-defined protocols to bridge the element code with the analysis core. If the developer and the system administrator agree, the new element can be merged into the analysis core and become part of the static element library. On the other hand, the element developer can simply register the element code and its location to the analysis core and the element service can then be accessed remotely over the Internet.

2.2 Mechanics of the Services Framework

The mechanics of the Internet-enabled model is illustrated in Figure 1, which shows the basic procedures to access the finite element compute engine and to perform nonlinear dynamic analysis over the Internet. First, a user of the system builds a structural model on the client site and then submits the model to the analysis core via a web-browser or an application program, using the Internet as a communication channel. Upon receiving the model and other related information, the core server authenticates the user's identity and starts performing structural analysis based on the received model. Depending on the underlying hardware and system of the server core, the analysis may be performed in a distributed and collaborative manner. During the analysis, elements that are available in the core can be accessed locally from the static element library (which is the case for most prevailing finite element packages), whereas other elements are obtained from online element services. In order to find the required elements not existing in the local element library, the registry is queried to find the location of the online element services. Once the online element services have been identified and bound, the analysis core can access these element services as if they were static local elements. After the analysis is completed, part of the results are returned to the user by generating a dynamic web page displayed in the user's web browser.

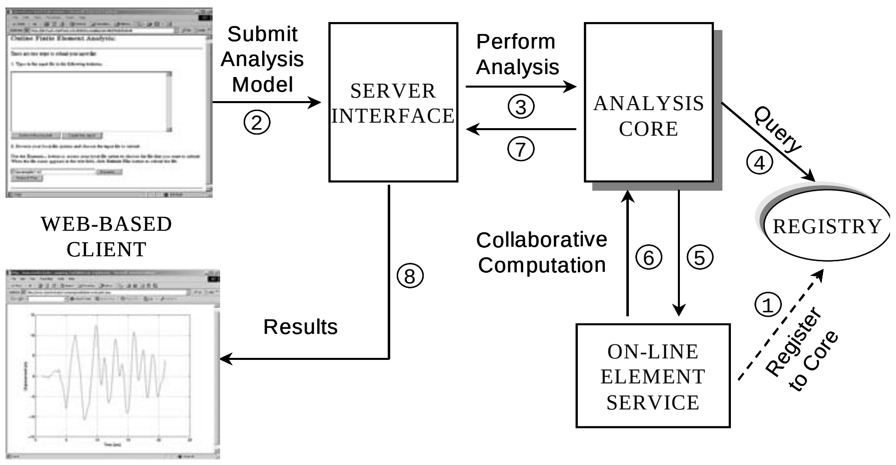


Figure 1. Mechanics of the Internet-Enabled Software Model

3 User Interfaces

As indicated in Figure 1, the services framework can offer users access to the analysis core, as well as the associated supporting services via the Internet. One benefit of this model is the transparency of software services. From a user's perspective, the user deals with a single service from a single point-of-contact – even though the actual structural analysis may be performed in a distributive manner. To facilitate user interaction with the core server, two types of interfaces are provided, namely web-based interface and MATLAB-based interface.

Client browser programs such as the Internet Explorer and Netscape Navigator allow users to navigate and access data across machine boundaries. In the Internet-enabled framework, a standard web browser is utilized to interact with the FEA core, as shown in Figure 2. Users of the framework can build a structural analysis model on the client site, and then save it as an input file. The structural analysis model can then be submitted to the server through the provided web interface. Whenever the server receives a request, it starts a new process to run the FEA core. The server monitors the progress of the simulation and informs the user periodically. After the analysis is complete, some pre-requested analysis results are returned from the FEA core to the user's browser as a generated web page. One feature of this model is that the server supports multithreading, so that the server is able to support simultaneously requests from multiple users without severe performance degradation.

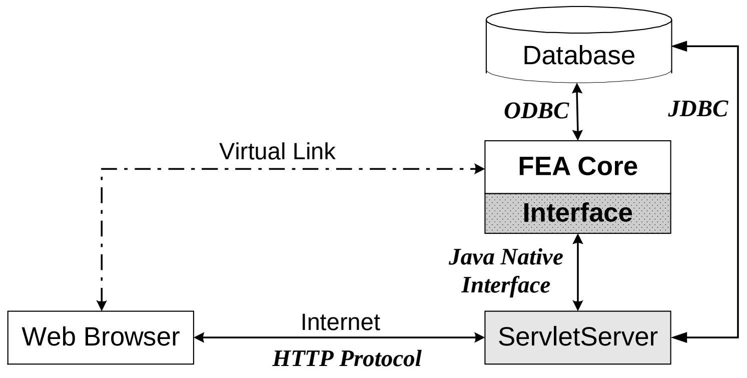


Figure 2. Interaction Diagram for the Web-based User Interface

For web-based services, all too often analysis result is downloaded from the server as a file, and then put manually (cut and paste plus maybe some cumbersome conversions) into another program, e.g. a spreadsheet, to perform postprocessing. For example, if we want to plot a time history response after a dynamic analysis, we might download the response in a data file and then use MATLAB, Excel, or other software packages to generate the graphical representation. It would be more convenient to directly utilize some popular application software packages to enhance the user interaction with the FEA core. In our prototype system, a MATLAB-based user interface is implemented to take advantage of the flexibility and graphical processing power of MATLAB. Various functions have been added to the standard MATLAB to handle the network communication and data processing tasks. These add-on functions can be directly invoked from a MATLAB-based graphical user interface.

4 Internet-Enabled Services Integration

One of the salient features of the software framework is to facilitate analysts to integrate new developments remotely so that the functionalities of the analysis core can be enhanced. A diverse group of users and developers can easily access the FEA program and contribute their own developments (such as elements, materials, analysis algorithms, and solution strategies, etc.) to the central core. For illustration purpose, this paper describes the service integration of elements to the analysis core. There are two types of online element services: namely distributed element service and dynamic shared library element service.

4.1 Software Integration with Services

Software integration takes place in many forms. The early attempts are based on code reuse. The simplest method is to copy the source code to wherever the desired functionality is needed. There are significant drawbacks to this approach, ranging from compiler incompatibility to difficulty in maintaining duplicate copies of code.

The development of network computing, especially the emergence of the Internet, allows software components to be distributed to multiple machines. Each software component runs as a separate process, communicating with each other by exchanging messages. This software integration model is called the distributed component model (Emmerich and Kaveh 2002). Assorted tools and standards for assembling distributed computing applications have been developed over the years. They started as low-level data transmission APIs and protocols, such as TCP/IP and RPC (Birrell and Nelson 1984) and have evolved into object-oriented distribution schemes, such as Object Management Group's (OMG) Common Object Request Broker Architecture (CORBA) (Pope 1998), Microsoft's Distributed Component Object Model (DCOM) (Eddon and Eddon 1998) and Sun Microsystems' Java Remote Method Invocation (RMI) (Pitt and McNiff 2001). These programming tools essentially provide a protocol for transmitting structured data (and, in some case, actual running code) over a network connection.

There are also web services technologies for facilitating distributed service composition, for example, Microsoft .NET (Kirtland 2000) and SOAP (Box et al. 2000). The functionalities provided by the web services are composed together to form an integrated service. Although the web services are distributed and heterogeneous, they are utilized as if they were locally available to the users. Communication messages are exchanged among the web services to coordinate the execution of the web services and to exchange data among the web services.

4.2 Distributed Element Services

A key feature of the distributed services framework is the interchangeability of components and the ability to integrate existing libraries and new components into the analysis core without the need to dramatically change the existing code. In an object-oriented FEA program, introducing a new type of element generally consists of creating a new subclass for a specific element (McKenna and Fenves 2000). This local object-computing paradigm can be extended to support distributed services. The essential requirements in a distributed object system are the ability to create and invoke objects on a remote host or process, and interact with them as if they were objects within the same local process.

In the prototype implementation, Java's Remote Method Invocation (RMI) (Pitt and McNiff 2001) is chosen to handle communication for the distributed element services over the Internet. Java RMI enables a program in one Java Virtual Machine (VM) to make method calls on an object located on a remote server machine (Farley 1998). The design goal for the RMI architecture is to create a Java distributed object model that integrates naturally into the local object model. The easy-to-use interface and the language features in Java can facilitate the development of client/server components. Java RMI also provides a high level of security and reliability in developing a distributed environment. However, compared with distributed computing environment with a design goal of supporting heterogeneous systems (such as CORBA and Microsoft .NET), there is one challenge for building distributed element service by using Java RMI: incorporating legacy systems in the Java infrastructure. A communication support between Java and other languages needs to be provided by using Java Native Interface (JNI) (Liang 1999). By programming through the JNI, we can use Java objects to access native methods and libraries.

Figure 3 shows the mechanics of the distributed element service. Within the infrastructure, an element service can be written in any popular programming languages: Java, C, C++, or Fortran. As long as the element service conforms to the defined protocol, the service can participate in the framework. The actual element code of a distributed element service resides in the service provider's site. The developed element service communicates with the analysis core through a communication layer. A remote method call initiated by the analysis core invokes certain method on the element service via the communication layer. For instance, the analysis core may issue a remote method invocation to send the input data of an element (e.g. geometry, nodal coordinates, Young's modulus, and Poisson ratio, etc.) to the element service. Later on, when the core needs certain element data, for example a stiffness matrix, it requests the data from the element service through another remote method invocation. The computation (e.g. the forming of the stiffness matrix of an element) is performed at the element service provider's site.

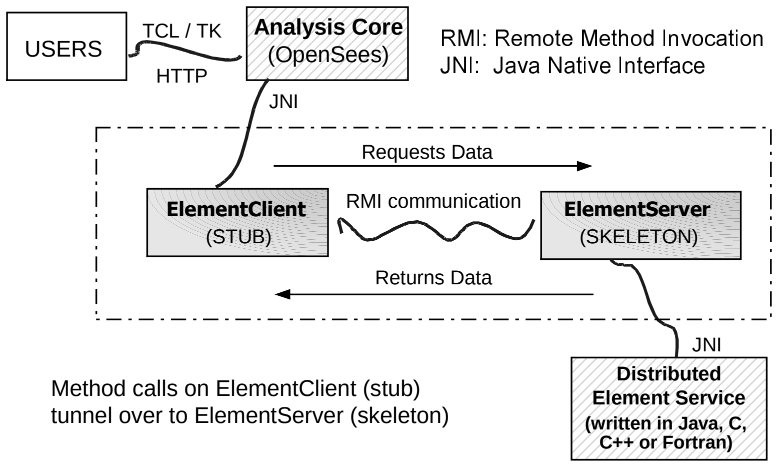


Figure 3. Mechanics of the Distributed Element Service

4.3 Dynamic Shared Library Element Services

The distributed element service model has performance overhead on remote method invocation, which is generally more expensive than a local procedure call. The system performance decreases because a remote method has to be invoked for accessing every distributed element. A dynamic shared library (or simply a shared library) element service is designed to improve the system performance without losing the flexibility and other benefits of the distributed services. Instead of being compiled to a static library and merged to the core server, an element service is built as a dynamic shared library and located 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.

There are many advantages for the shared library element services. One advantage is that the shared library can be loaded at runtime, so that different services can be replaced at runtime without re-compilation and re-linking with the application. Another benefit of using dynamic shared library is that the used binary format guarantees that the source code of the element will not be exposed to the core server, making the building of proprietary software components easier. This also implies that the element developer controls the maintenance, quality, and upgrade of the source code, facilitating bug-fixing and version control of the element service. However, the dynamic shared library element service is platform dependent. 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. Other disadvantages include potential security problem and minor performance overhead due to network downloading and dynamic binding.

5 Distributed Project Management

In nonlinear dynamic analysis, extensive study of a structure involves numerous simulations to be performed. Typically, model calibration requires several analysis models to be constructed with different model characteristics. To conduct a probability analysis, earthquake records with different magnitudes are applied to the same analysis model. Since there are numerous simulations for the same structure, a project management mechanism is needed to store and manage model information and simulation results (Peng and Law 2004).

The Internet-enabled framework provides users with a project management service that enables collaborative work on a specific structure model. In this work, we refer to *project* as a particular analysis model, *case* as a specific simulation, and *owner* as the user who performs simulation on a case and manages the simulation results. Following this convention, a structure could potentially result in several projects, with each representing the analysis of one model. Each project could incorporate many cases by applying different earthquake records or using different analysis strategies on the same model.

Due to the large amount of model data and simulation results as well as multiple users performing the analyses, it is not desirable to store all analysis results and related data in a single computer. Therefore, a centralized control-distributed data storage scheme is adopted in the project management service. Specifically, a central server stores all of the project background information (such as size of the model, the types of finite elements, specific boundary conditions, etc.) and the locations of projects and cases. The actual analysis data (such as finite element models, documents, simulation results, etc.) of each case resides on disparate sites and managed by different owner. The distributed nature of data storage allows the owner to perform simulation and manage the results locally.

There are several other capabilities of the project management service in the framework. One important feature is network transparency, which results in a unified front-end representation to users, regardless of where the data is stored. The complexity of the distributed data storage is hidden from the users. Besides network transparency, other features of the project management service are access control and version control capabilities. Access control is incorporated to differentiate users and to allow them access projects based on their privileges. For instance, owner of a project has full control on all the project data, while some users are only allowed to access the data without the ability to modify. Version control is implemented with the project management service to keep track of the modifications to the analysis models and to present the difference between projects.

6 Application Example

This section illustrates the usage of the Internet-enabled framework by carrying out seismic performance analyses of the Humboldt Bay Middle Channel Bridge located at Eureka, California. The bridge is a 330 meters long, 9-span composite structure with precast and prestressed concrete I-girders and cast-in-place concrete slabs to provide continuity. It is supported on eight pile groups, each of which consists of 5 to 16 prestressed concrete piles. The foundation soil is mainly composed of dense fine-to-medium sand (SP/SM), organic silt (OL), and stiff clay layers. In addition, thin layers of loose and soft clay (OL/SM) are located near the ground surface. A two-dimensional FEA model of the bridge, including the superstructure, piers, and supporting piles, was developed as shown in Figure 4 (Conte et al. 2002). The bridge piers are modeled using 2-D fiber beam-column elements and the abutment joints are modeled using zero-length elasto-plastic gap-hook elements. A four-node quadrilateral element is used to discretize the soil domain.

Nonlinear dynamic analysis is conducted on the analysis model by using the Internet-enabled framework. As an example, the quadrilateral soil element in the model is built as a distributed element service. Figure 5 illustrates the interaction among the distributed services during a simulation of the model. The analysis core is running on a central server computer called *opensees.stanford.edu*. The developed quadrilateral element services are running on a computer named *galerkin.stanford.edu*. Figure 5 also shows a postprocessing service running on *epic21.Stanford.edu*. This postprocessing service is used to generate graphical representations of time history responses. The web-based user interaction interface is shown in Figure 5.

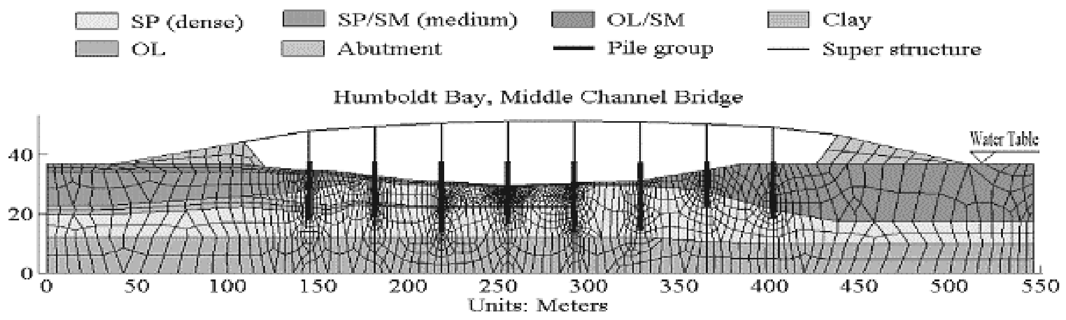


Figure 4. Finite Element Model for Humboldt Bay Bridge (from Conte et al. 2002)

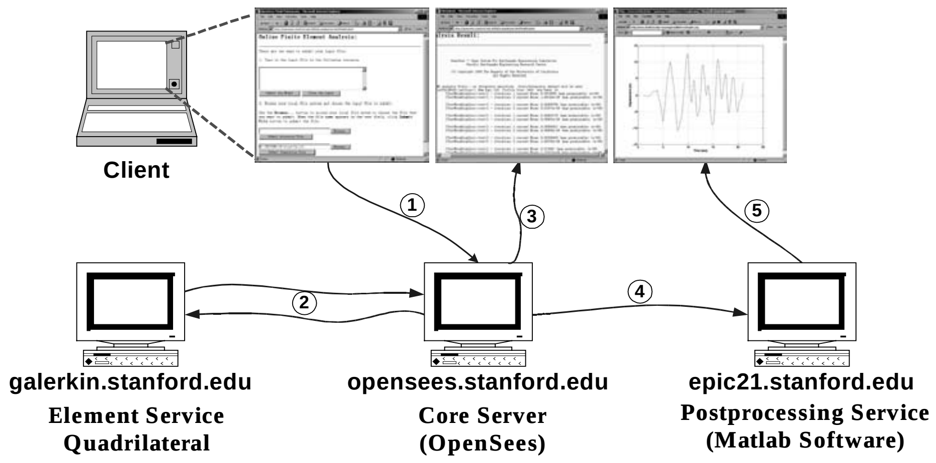
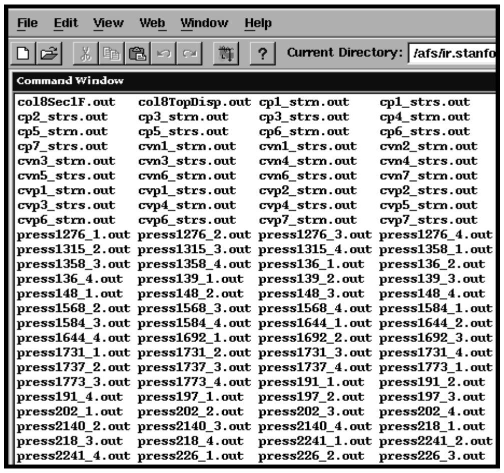


Figure 5. Interaction of Distributed Services

The users can also communicate with the server directly via a MATLAB-based interface. The MATLAB-based interface can be used to invoke structural analysis. After the simulation, the command `listResults` can be issued to retrieve the list of response time history files, as shown in Figure 6(a). We can also directly generate graphical representation of a particular response time history. For instance, Figure 6(b) shows the result of using command `res2Dplot('press1315_2.out')` to plot the pore pressure time history.

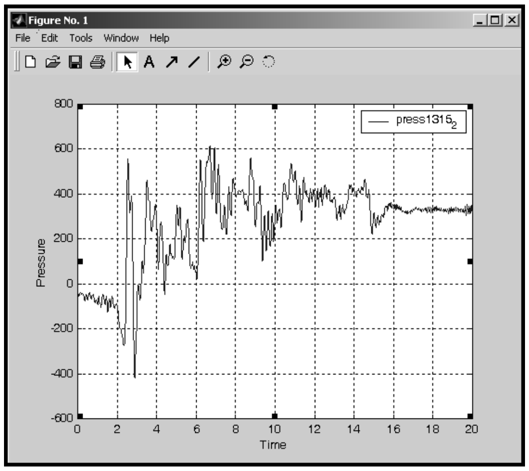
To conduct probability analysis, approximately sixty ground motion records are applied to the bridge model for damage simulation. The ground motion records are divided into three hazardous levels based on their probability of occurrence, which are 2%, 10%, and 50% in fifty years respectively. Figure 7 shows a partial list of the Humboldt Bay Bridge projects. When using the project management service developed in this work, a web interface is a single point-of-entry for all the project related background information, documents, messages, and simulation results. Adding a new simulation to the system is simple. After performing a simulation on the analysis model with the new earthquake record, the owner can register the simulation to the central server. Once the registration process is completed, the central web page will be updated to reflect the modification. Detailed information of a particular simulation can be retrieved by clicking on the project name, which is a hyperlink to the project website located in the owner's computer. As highlighted in Figure 7, the case Rinaldi of project X1 is added and can be accessed.



File Edit View Web Window Help
Current Directory: /afs/ir.stanfo
Command Window

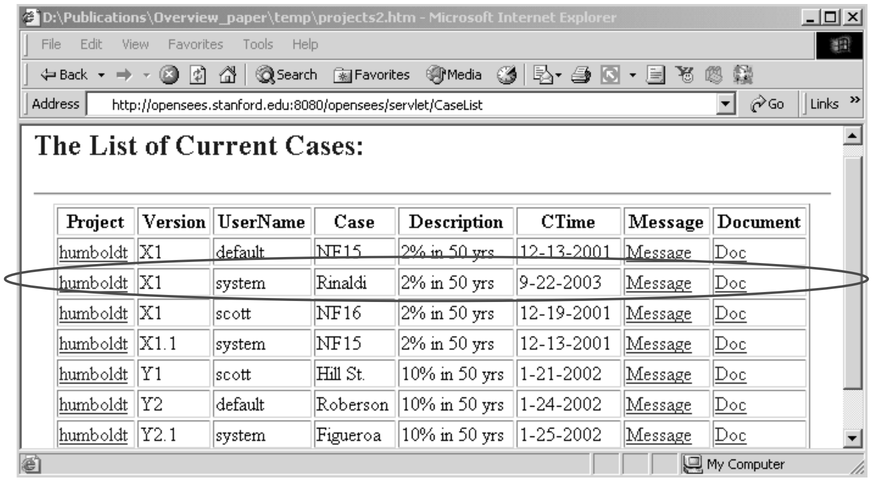
```
col8Sec1F.out   col8TopDisp.out cp1_strn.out    cp1_strs.out
cp2_strs.out    cp3_strn.out    cp3_strs.out    cp4_strn.out
cp5_strn.out    cp5_strs.out    cp6_strn.out    cp6_strs.out
cp7_strs.out    cvn1_strn.out   cvn1_strs.out   cvn2_strn.out
cvn3_strn.out   cvn3_strs.out   cvn4_strn.out   cvn4_strs.out
cvn5_strs.out   cvn6_strn.out   cvn6_strs.out   cvn7_strn.out
cvp1_strn.out   cvp1_strs.out   cvp2_strn.out   cvp2_strs.out
cvp3_strs.out   cvp4_strn.out   cvp4_strs.out   cvp5_strn.out
cvp6_strn.out   cvp6_strs.out   cvp7_strn.out   cvp7_strs.out
press1276_1.out press1276_2.out press1276_3.out press1276_4.out
press1315_2.out press1315_3.out press1315_4.out press1358_1.out
press1358_3.out press1358_4.out press136_1.out  press136_2.out
press136_4.out  press139_1.out  press139_2.out  press139_3.out
press148_1.out  press148_2.out  press148_3.out  press148_4.out
press1568_2.out press1568_3.out press1568_4.out press1584_1.out
press1584_3.out press1584_4.out press1644_1.out press1644_2.out
press1644_4.out press1692_1.out press1692_2.out press1692_3.out
press1731_1.out press1731_2.out press1731_3.out press1731_4.out
press1737_2.out press1737_3.out press1737_4.out press1773_1.out
press1773_3.out press1773_4.out press191_1.out  press191_2.out
press191_4.out  press197_1.out  press197_2.out  press197_3.out
press202_1.out  press202_2.out  press202_3.out  press202_4.out
press2140_2.out press2140_3.out press2140_4.out press218_1.out
press218_3.out  press218_4.out  press2241_1.out press2241_2.out
press2241_4.out press226_1.out  press226_2.out  press226_3.out
```

(a) listResults



(b) res2Dplot('press1315_2.out')

Figure 6. Sample MATLAB-Based User Interface



D:\Publications\Overview_paper\temp\projects2.htm - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://opensees.stanford.edu:8080/opensees/servlet/CaseList
Go Links
My Computer

The List of Current Cases:

Project	Version	UserName	Case	Description	CTime	Message	Document
humboldt	X1	default	NF15	2% in 50 yrs	12-13-2001	Message	Doc
humboldt	X1	system	Rinaldi	2% in 50 yrs	9-22-2003	Message	Doc
humboldt	X1	scott	NF16	2% in 50 yrs	12-19-2001	Message	Doc
humboldt	X1.1	system	NF15	2% in 50 yrs	12-13-2001	Message	Doc
humboldt	Y1	scott	Hill St.	10% in 50 yrs	1-21-2002	Message	Doc
humboldt	Y2	default	Roberson	10% in 50 yrs	1-24-2002	Message	Doc
humboldt	Y2.1	system	Figueroa	10% in 50 yrs	1-25-2002	Message	Doc

Figure 7. The List of Updated Humboldt Bay Bridge Cases

7 Conclusions

This paper presents an Internet-enabled framework that can facilitate the development of structural analysis services and the access to simulation results. The main design principle of this framework is to keep the software kernel flexible and extendible, so that a diverse group of users and developers can easily access the platform and attach their own developments to the core server. The web service architecture allows new services to be remotely incorporated into the modular FEA platform. Users can select appropriate services and can easily replace one service by another without having to recompile or reinitialize the existing services. The Internet is utilized in the software framework as a communication channel to link distributed software services and to access simulation results. One important feature of the Internet-enabled framework is network transparency, which makes it indifferent to users whether code is running locally or remotely. The end users do not need to be aware of the complexity of the core server in terms of both its hardware and software configurations.

A data and project management service is provided to manage simulation results and other pertinent information. This research has presented the potentials of using a project management system to archive project-related data and to perform access and revision control. The project management system allows the usage of a database system to manage the information related to projects. The actual project data is stored in distributed machines. The project management service can also facilitate users to collaboratively working together on a specific structural model.

Performance of the collaborative framework remains to be a key challenge for the wide adoption of the collaborative system for engineering practice. The performance penalties imposed by distributed services can potentially be high. The scalability of the system should also be further improved. The current implementation relies on Java's multithreading feature to handle simultaneous requests from uses. The performance might be substantially degraded when there are a large number of services that are geographically dispersed or where there are numerous clients access the server simultaneously. This scalability problem could be tackled by improving both hardware and software. Multiple core servers and more powerful computers can be deployed, for example, the locally distributed web-server systems are shown to improve the performance and scalability (Cardellini et al. 2002). Another possibility to enhance the framework is to utilize parallel and distributed computing environment (De Santiago and Law 2000) and better fault-tolerance and fault-recovery techniques for distributed applications.

8 Acknowledgements

This work has been supported in part by the Pacific Earthquake Engineering Research Center through the Earthquake Engineering Research Centers Program of the National Science Foundation under Award number EEC-9701568, and in part by NSF Grant Number CMS-0084530. The authors would like to thank Dr. Frank Mckenna and Prof. Gregory Fenves of UC, Berkeley for their collaboration and support of this research. The authors are grateful to Dr. Zhaohui Yang, Mr. Yuyi Zhang, Prof. Ahmed Elgamal, and Prof. Joel Conte of UCSD for providing the Humboldt Bay Bridge model. The authors would also like to acknowledge an equipment grant from Intel Corporation for the support of this research.

9 References

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

Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, H., Thatte, S., and Winer, D. (2000). "Simple Object Access Protocol (SOAP)." W3C Note, http://www.w3.org/TR/SOAP.

Cardellini, V., Casalicchio, E., Colajanni, M., and Yu, P. S. (2002). "The State of the Art in Locally Distributed Web-Server Systems." ACM Computing Surveys, 34(2), 263-311.

Conte, J. P., Elgamal, A., Yang, Z., Zhang, Y., Acero, G., and Seible, F. (2002). "Nonlinear Seismic Analysis of a Bridge Ground System." Proceedings of the 15th ASCE Engineering Mechanics Conference, New York, NY.

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.

Eddon, G., and Eddon, H. (1998). Inside Distributed COM, Microsoft Press, Redmond, WA.

Emmerich, W., and Kaveh, N. (2002). “Component technologies: Java beans, COM, CORBA, RMI, EJB and the CORBA component model.” Proceedings of 24th International Conference on Software Engineering, Orlando, FL, 691-692.

Farley, J. (1998). Java Distributed Computing, O'Reilly & Associates, Sebastopol, CA.

Han, C. S., Kunz, J. C., and Law, K. H. (1999). “Building Design Services in A Distributed Architecture.” Journal of Computing in Civil Engineering, 13(1), 12-22.

Kirtland, M. (2000). “The Programmable Web: Web Services Provides Building Blocks for the Microsoft .NET Framework.” MSDN Magazine, http://msdn.microsoft.com/msdnmag/issues/0900/WebPlatform/default.aspx.

Liang, S. (1999). Java Native Interface: Programmer's Guide and Specification, Addison-Wesley, Boston, MA.

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.

Peng, J. (2002). “An Internet-Enabled Software Framework for the Collaborative Development of a Structural Analysis Program,” Ph.D. Thesis, 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.

Peng, J., and Law, K. H. (2004). “Building Finite Element Analysis Programs in Distributed Services Environment.” Computers & Structures, accepted for publication.

Pitt, E., and McNiff, K. (2001). java™.RMI: The Remote Method Invocation Guide, Addison-Wesley, Boston, MA.

Pope, A. (1998). The CORBA Reference Guide: Understanding the Common Object-Request Broker Architecture, Addison-Wesley, Boston, MA.

Roy, J., and Ramanujan, A. (2001). “Understanding Web Services.” IT Professional, 3(6), 69-73.

Smith, B. L., and Scherer, W. T. (1999). “Developing Complex Integrated Computer Applications and Systems.” Journal of Computing in Civil Engineering, 13(4), 238-245.