



A MODULAR FRAMEWORK FOR EARTHQUAKE ENGINEERING ANALYSIS AND SIMULATIONS

Jun PENG¹, Frank MCKENNA² and Kincho H. LAW³

SUMMARY

This paper describes a modular framework that facilitates the development of finite element analysis (FEA) programs. The framework allows users easy access to the FEA program and the analysis results by using a web-browser or other application programs, such as MATLAB. In addition, the framework enables new as well as legacy codes to be incorporated from disparate sites in a dynamic and distributed manner. To provide flexible project management and data access, a database system is employed to store project-related information and selected analysis results. The prototype framework demonstrates that the Internet can potentially enhance the flexibility and extendibility of traditional FEA programs.

INTRODUCTION

Practicing engineers today usually perform earthquake engineering simulations on a dedicated computer using the developments offered by a finite element analysis (FEA) program. Typically, a FEA program bundles all the procedures and program kernels that are developed by an individual organization. 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. Object-oriented design principles and programming have been utilized in FEA software development to support better data encapsulation and to facilitate code reuse [1]. Object-oriented FEA programs, particularly those written in C++, have been shown to have comparable performance to their procedural-based counterparts [2, 3]. 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. Moreover, there is no easy way to access computing resources and structural analysis services distributed in remote sites.

With the maturation of information and communication technologies, the concept of building collaborative systems to distribute web services over the Internet is becoming a reality [4]. Using a set of standardized communication protocol, web services can be universally accessed and deployed,

¹ Research Associate, Stanford University, Stanford, CA, USA. E-mail: junpeng@stanford.edu

² Research Associate, University of California, Berkley, CA, USA. E-mail: fmckenna@ce.berkeley.edu

³ Professor, Stanford University, Stanford, CA, USA. E-mail: law@stanford.edu

independent of the underlying operation environment. The goal is to provide a platform for building distributed applications that utilize software components running on disparate sites. Following the web services model, we have designed and prototyped an Internet-enabled modular framework for the usage and development of a FEA program [5, 6]. The modular framework adopts web service model to enhance and improve the capability of a FEA program by seamlessly integrating legacy code and new developments. Developments 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 data and project management functionalities. A database system is employed to store selected analysis results and to provide flexible data management and data access. Project management functions allow users to access information about previous simulations of related models stored distributively in different sites.

In the prototype implementation of the Internet-enabled framework, OpenSees [7] is utilized as the finite element analysis core. OpenSees (Open System for Earthquake Engineering Simulation) is an object-oriented program for the seismic simulation of structural and geotechnical systems. It is sponsored by PEER (Pacific Earthquake Engineering Research) center, and is intended to serve as the computational platform for research in performance-based earthquake engineering. While the prototype implementation is using OpenSees as the FEA core, the framework is sufficiently general to be incorporated by other FEA programs.

STRUCTURAL ANALYSIS FRAMEWORK

The design of the framework is focused on the interaction of services, with which the system can provide users a plug-and-play character. The overall system architecture of the Internet-enabled collaborative framework is schematically depicted in Figure 1. The architecture defines the dependency and the interaction among the participants.

- In this framework, the structural analysis core program is running as a computational server. Element and material models, solvers, as well as analysis strategies and solution strategies, are brought into this module to improve the functionality of the core. In the prototype implementation, OpenSees [7] is employed as the analysis core.
- Users of the framework 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) contributed by other developers that have been tested and incorporated into the core platform.
- For element developers, a standard interface/wrapper is defined for communicating the element(s) 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. Moreover, the element code can be constructed as a distributed service and then be accessed remotely over the Internet.
- A database system is linked with the core server to support the data storage and project management service. This service can provide selective 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.

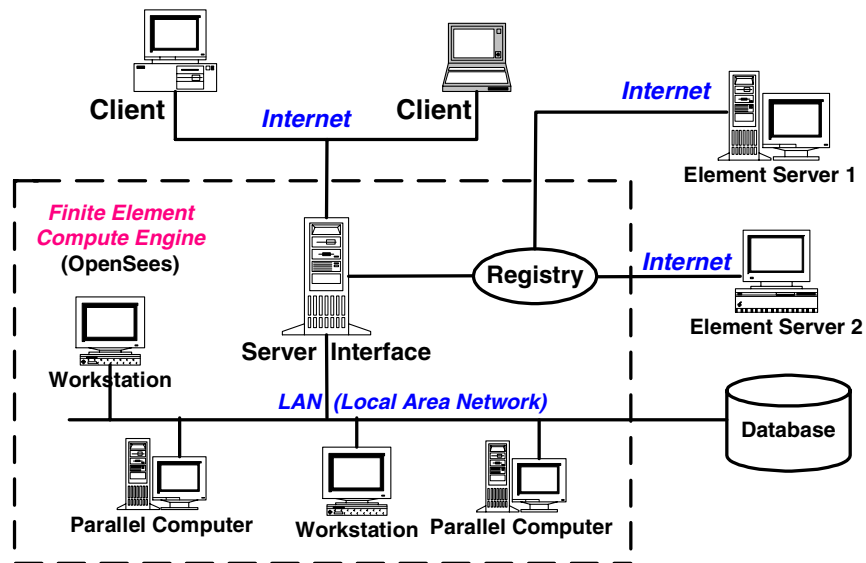


Figure 1. The Collaborative System Architecture

USER INTERACTION

As shown in Figure 1, the 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 distributed and collaborative manner. Another benefit is that this framework can widen the reach of the FEA core program to the users and external developers. The core platform offering the FEA service stays at the provider's site, where the software core is developed, kept securely, operated, maintained and updated.

Web-Based User 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 World Wide Web browser is utilized to interact with the FEA core. The user 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 properly 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.

MATLAB-Based User Interface

For web-based services, all too often analysis result is downloaded from the computational 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 available to take advantage of the flexibility and graphical processing power of MATLAB. In the implementation, some extra functions are added to the standard MATLAB to handle the network communication and data processing. These add-on functions can be directly invoked from a MATLAB-based graphical user interface.

DISTRIBUTED ENGINEERING SOFTWARE SERVICES

One of the salient features of the software framework is to facilitate analysts to integrate new developments remotely with the analysis core so that the functionalities can be enhanced. The modular framework would allow new application services to be incorporated with the analysis core in a dynamic and distributed manner. A diverse group of users and developers can easily access the FEA program and contribute their own developments to the central core. There are two types of online element services, namely distributed element service and dynamic shared library element service. As opposed to the traditional statically linked element library, the online element services will not expose the source code to the core. Therefore, the framework allows the building of proprietary element services and facilitates the linking of legacy applications.

Distributed Element Service

A key feature of the 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 [7]. This local object-computing paradigm can be extended to support distributed services. Instead of only using the objects that reside exclusively on one local computer, the framework utilizes distributed objects to allow the building of a distributed application to facilitate new element development.

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) [8] is chosen to handle communication for the distributed element services over the Internet. 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.

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. The shared library element service allows the replacement of an element service without reinitiating the core server.

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.

DATA AND PROJECT MANAGEMENT

The handling of data shared between disparate systems requires persistent storage of the data and corresponding interfaces to access the data. As shown in Figure 1, a database is linked with the framework to provide users with easy and efficient access to project information and structural analysis results [9]. The data management service would allow the users to query the core server for useful analysis results, and the information retrieved from the database through the core server is returned to the users in a standard format. The project management service also facilitates the users to manage and to collaborate on engineering analysis and simulation project.

Multi-tiered Architecture

Figure 2 depicts the architecture of the data management service. A multi-tiered architecture is employed instead of the traditional two-tier client-server architecture. The multi-tiered architecture provides a flexible mechanism to organize distributed services. Since components in the system are modular and self-contained, they could be designed and developed separately. The multi-tiered data management system has the following components:

- A standard interface is provided for the *Remote Client* programs to access the server system. Application programs, such as web browsers or MATLAB, can access the server core and the analysis results from the client site via the pre-defined communication protocols.
- Java Servlet-enabled *Web Server* is employed to receive the requests from the clients and forward them to the Application Server. The Web Server also plays the role of re-formatting the analysis results in certain HTML format for the web-based clients.
- The *Application Server* is the middle layer for handling communication between the Web Server and the Data Server. The Application Server also provides the core functionalities for performing analyses and generating analyses results. In the Internet-enabled framework, the FEA core is situated in the Application Server.
- A COTS (Commercial Off-The Shelf) database system is utilized as the Data Server for the storage and retrieval of selected analysis results. Examples of COTS database systems include Oracle [10], Access [11], and MySQL [12]. The communication between the Application Server and the Database is handled via the standard data access interfaces based on Open Database Connectivity (ODBC) that most COTS database systems provide. ODBC makes it possible to access different database systems with a common language.

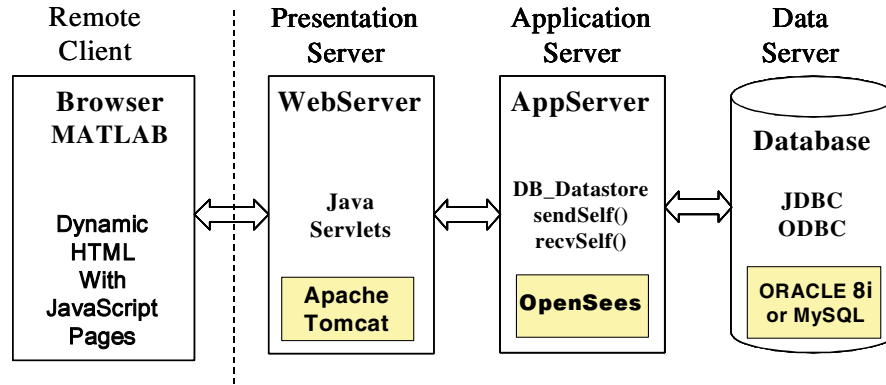


Figure 2. Project and Data Management System Architecture

Data Storage Scheme

A typical finite element analysis generates large volume of data, which can be saved and retrieved in two ways. One approach is to pre-define all the required data and save only those pre-defined data during the analysis. However, if analysis results other than pre-defined are needed, a complete re-analysis is needed to generate the results. An earthquake simulation of a large structural model can be expensive in terms of both processing time and storage requirement. The other approach is simply dumping all the interim and final analysis data into files, which are then utilized to retrieve the required results as a postprocessing task. The drawbacks of this approach are the substantial amount of storage space and the potential poor performance due to the expensive search on the large data files.

In the data management proposed in this work, the data storage scheme is carried out in a different fashion. The data management service allows the collection of certain information to be saved as the analysis progresses, e.g. the maximum nodal displacement at a node or the time history response of a nodal displacement. The purpose is to keep track of the progress of an analysis and output the users' pre-specified results. Besides the recording functionalities, the data access system also has the restart capability. Certain selected data are stored during the analysis that allows the analysis results to be restored as needed. In the data access system, we use object serialization [13] to facilitate the restart function. Ordinarily, an object lasts no longer than the program that creates it. In this context, persistence is the ability of an object to record its state so that the object can be reproduced in the future, even in another runtime environment. Through object serialization, the object can be shared outside the address space of an application by other application programs. Consider a Truss element as an example, its nodes, dimension, number of DOFs, length, area, and material properties can be saved in a file or a database system. Based on these stored data, a copy of the Truss object can be restored, and the stiffness matrix of the Truss element can then be re-generated.

The restart function introduced in the framework is different from those supported by current commercial FEA programs (e.g. ANSYS, ABAQUS, etc.). For most commercial FEA programs, the data saved in the restart file must conform to certain pre-defined data format. The restart function in the framework, on the other hand, relies on object serialization. As long as a replica of an object can be recreated with the saved data, the developer of the class can freely decide what kind of data to be saved and manipulate the saved data. Furthermore, the restart file of most commercial FEA programs is organized as a sequential file. On the other hand, the restart data saved in the framework can be retrieved randomly. Therefore, a particular object or a sub-model of the finite element model can be easily restored without retrieving unnecessary data.

Project Management

In a typical structural engineering project, many simulations of the same structure with different model characteristics are performed. To conduct a probability analysis, earthquake records with different magnitudes are applied to the same analysis model. A project management mechanism is therefore desirable to facilitate storing and managing model information and simulation results. The modular framework provides users with project management capability. Due to the potentially large amount of model data and simulation results of a structure, it is infeasible to store all analysis related data in a single computer. A centralized control-distributed data storage scheme is adopted. 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. The actual analysis data (such as finite element models, documents, simulation results, etc.) of each project resides on disparate sites and managed by the owner of the simulation.

There are several other capabilities of the project management service implemented in the framework, such as access control and version control. Access control is incorporated to differentiate users and to allow them access projects based on their privileges. 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.

APPLICATION EXAMPLE

This section presents the usage of the Internet-enabled framework to investigate seismic performance of the Humboldt Bay Middle Channel Bridge, which is located at Eureka in northern 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 Middle Channel Bridge, including the superstructure, piers, and supporting piles, was developed as shown in Figure 3 [14]. 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.

The prototype framework is employed to conduct a nonlinear dynamic analysis on the analysis model. As an example, the quadrilateral element in the model is built as a distributed element service. Figure 4 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 web server and Java Servlet server are also running on this computer. The developed quadrilateral element services are running on a computer named *galerkin.stanford.edu*. As we indicated before, users only need to know the location of the central server without the awareness of the underlying distributed framework. Figure 4 also shows a postprocessing service running on *epci21.Stanford.edu*. This postprocessing service is employed to generate graphical representations of time history responses. Besides the illustrated web-based interface, 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 5(a). We can also directly generate graphical representation of a particular response time history. For instance, Figure 5(b) shows the result of using command `res2Dplot('press1315_2.out')` to plot the pore pressure time history of a particular node.

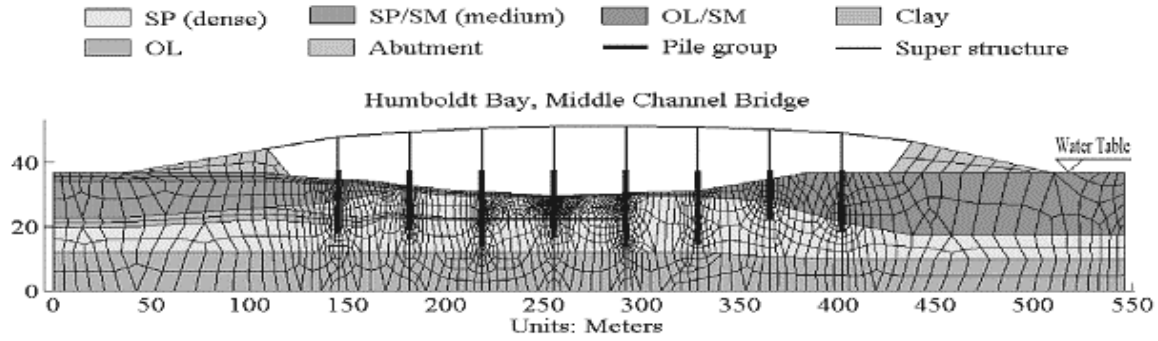


Figure 3. Finite Element Model for Humboldt Bay Bridge (from [14])

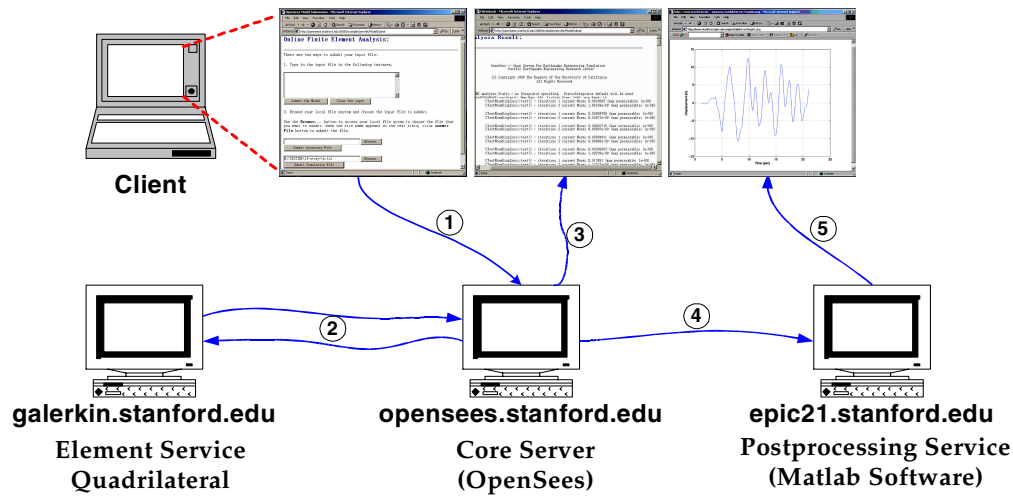


Figure 4. Interaction of Distributed Services

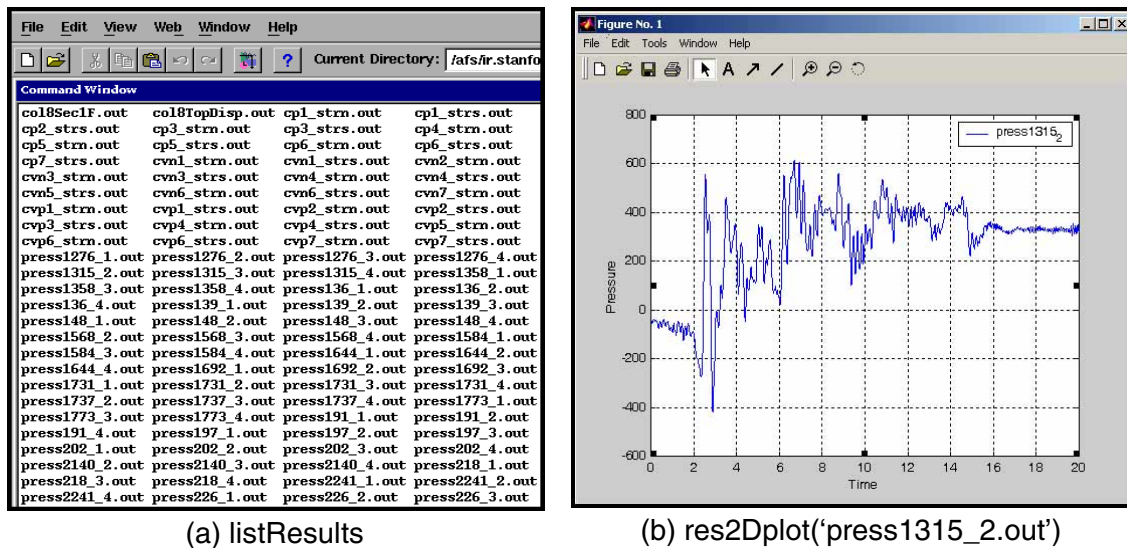
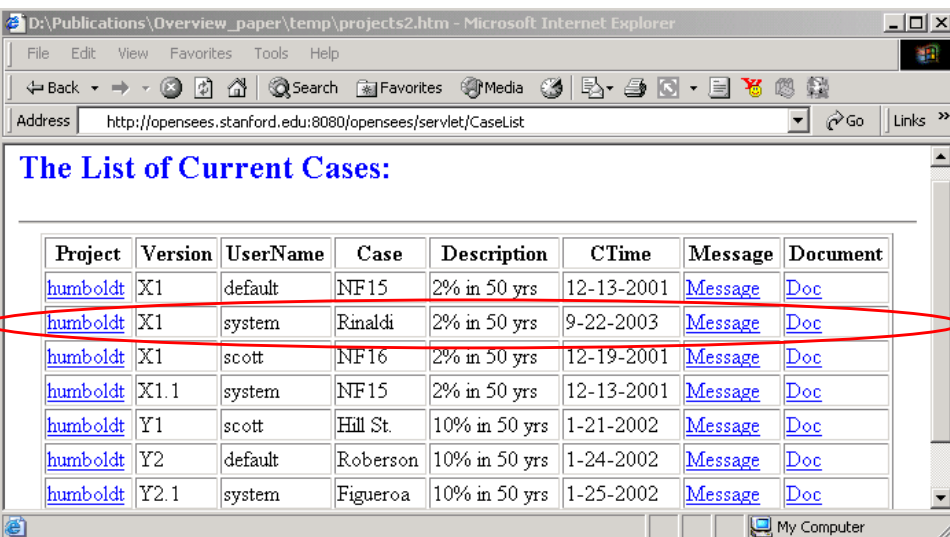


Figure 5. Sample MATLAB-Based User Interface

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. Many simulations are conducted on the bridge model by applying different ground motion records. Figure 6 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 put all the project related data and simulation results in a local website, and then 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. We will use case Rinaldi of project X1 (highlighted in Figure 6) as an illustration. The ground motion applied to this case is the 1994 Northridge earthquake recorded at the Rinaldi station (PGA = 0.89g, PGV = 1.85 m/sec, PGD = 0.60 m), with a probability of 2% occurrence in fifty years.



The screenshot shows a web browser window with the address bar displaying 'http://opensees.stanford.edu:8080/opensees/servlet/CaseList'. The page title is 'The List of Current Cases:'. Below the title is a table with the following data:

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	Figuroa	10% in 50 yrs	1-25-2002	Message	Doc

Figure 6. The List of Updated Humboldt Bay Bridge Cases

CONCLUSIONS

This paper described a modular framework that can facilitate the development of earthquake engineering simulation programs 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 framework and attach their own developments to the core server. Users can select appropriate services and can easily replace one service by another without having to recompile or reinitialize the existing services. The framework provides both web-based interface and MATLAB-based interface to facilitate the user interaction. The software framework described in this paper provides an execution environment that users only deal with a single server. 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. A selective data storage scheme is introduced to provide flexible support for the tradeoffs between the time used for reconstructing analysis model and the space used for storing the analysis results. 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 project model.

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 are grateful to Dr. Zhaohui Yang, Mr. Yuyi Zhang, Prof. Ahmed Elgamal, and Prof. Joel Conte 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.

REFERENCES

1. McKenna F. "Object-oriented finite element programming: Frameworks for analysis, algorithm and parallel computing." Ph.D. Thesis, University of California at Berkeley, Berkeley, CA. 1997.
2. Dubois-Pelerin Y, Zimmermann T. "Object-oriented finite element programming: III. An efficient implementation in C++." *Computer Methods in Applied Mechanics and Engineering* 1993; 108(1-2): 165-83.
3. Rucki MD, Miller GR. "Algorithmic framework for flexible finite element-based structural modeling." *Computer Methods in Applied Mechanics and Engineering* 1996; 136(3-4): 363-84.
4. Han CS, Kunz JC, Law KH. "Building design services in a distributed architecture." *Journal of Computing in Civil Engineering* 1999; 13(1): 12-22.
5. Peng J, Law KH. "A prototype software framework for internet-enabled collaborative development of a structural analysis program." *Engineering with Computers* 2002; 18(1): 38-49.
6. Peng J, McKenna F, Fenves GL, Law KH. "An open collaborative model for development of finite element program." *Proceedings of the Eighth International Conference on Computing in Civil and Building Engineering (ICCCBE-VIII)*, Palo Alto, CA. 2000.
7. McKenna F, Mazzoni S, Scott MH, Fenves GL. "OpenSees: Open system for earthquake engineering simulation." <http://opensees.berkeley.edu>, 2003.
8. Pitt E, McNiff K. "Java^(tm).RMI: The remote method invocation guide." Addison-Wesley, 2001.
9. Peng J, Liu D, Law KH. "An engineering data access system for a finite element program." *Advances in Engineering Software* 2003; 34(3): 163-81.
10. Kyte T. "Expert one on one: Oracle." Wrox Press, 2001.
11. Andersen V. "Access 2000: The complete reference." McGraw-Hill Osborne Media, 1999.
12. DuBois P, Widenius M. "MySQL." New Riders Publishing, 1999.
13. Breg F, Polychronopoulos CD. "Java virtual machine support for object serialization." *Proceedings of the ISCOPE Conference on ACM 2001 Java Grande*, Palo Alto, CA. 2001.
14. Conte JP, Elgamal A, Yang Z, Zhang Y, Acero G, Seible F. "Nonlinear seismic analysis of a bridge ground system." *Proceedings of the 15th ASCE Engineering Mechanics Conference*, New York, NY. 2002.