DISSERTATION PROPOSAL

A Software Framework for Collaborative Development of Nonlinear Dynamic Analysis Code

Jun Peng
Stanford University
Nov. 17, 2000

Introduction

It is well recognized that a significant gap exists in the state-of-the-art computing methodologies and the state-of-practice in structural engineering analysis programs. In current engineering practice, finite element software packages need to be able to accommodate new advances in element and material formation, solution strategies, and computing environments. However, most prevailing structural analysis programs bundle all the procedures and program kernels into software packages that are developed by individual organizations. Extending these programs to incorporate new developments is a difficult process and more importantly, there is no easy way to link customized components by users and researchers separately outside the organization.

The development of OpenSees (McKenna 1997) now is facing exactly the same challenge. At the early stage of OpenSees, it was developed and tested within a small group at UC, Berkeley and Stanford University. The communication and cooperation among the researchers are very easy to achieve. After gaining its acceptance and popularity, OpenSees is now being used and developed by PEER researchers. Many people are involved in the development with different perspective and different focus. There are users whose main purpose is to use OpenSees as a structural analysis tool. There are core developers who are intended to incorporate new developments and to update analysis core. There are solution and analysis strategies developers as well as researchers whose main focus is developing new element and material technologies. Since the development is concurrent and incremental, traditional waterfall software development approach cannot meet the requirements. The management of developers and source code becomes important for the success of continuing development.

With the maturation of information and communication technologies, the concept of building collaborative systems to distribute the services over the Internet is becoming a reality (Han et al. 1999). Following this idea, we have proposed an open collaborative framework for the continuing development of OpenSees (Peng and Law 2000a, Peng et al. 2000b). A collaborative system is one where multiple users or agents engage in a shared activity, usually from remote locations. Unlike the development of traditional packaged

structural analysis programs, the collaborative framework can potentially reduce the overhead of continuous upgrade and extension. Users and engineers can select appropriate services and can easily replace a specific module if a superior module becomes available. Developers and researchers can concentrate on developing components and can easily integrate their component to the core through a *plug-and-play* environment.

Architecture of Collaborative Model

The system architecture of our proposed collaborative framework is schematically depicted in Figure 1. The core server is designed in a modular manner. The *Analysis Core* module is the part that most PEER researchers are working on. New element and material technologies are brought into this module to enhance the functionality of the core. In this part, parallel and distributed computing can also be deployed so that large-sized engineering problems can be solved within a reasonable amount of time (Santiago 1996).

Our work involves prototyping and developing the other modules: The *User-Interaction* module is deployed to provide web-based interface to the users and developers. The *Registration and Naming Service* is provided for on-line services to register to the core so that these services can be found during analysis. The *Distributed Element Service* is provided for remote access to elements resided in different sites. The *Dynamic Linked Element Service* is implemented to provide a flexible way of dynamically binding elements to the core in real time. Last but not least, the *Database Interface* module can take advantage of a commercial database to provide efficient data access and to enable flexible post-processing.

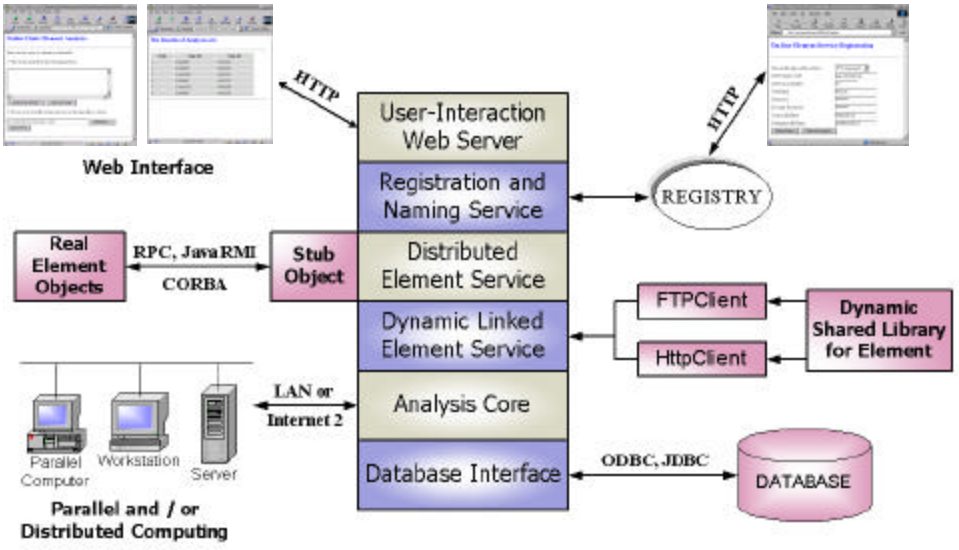


Figure 1 – Collaborative System Modules

The proposed Internet-enabled collaborative model can provide greater flexibility and extensibility than traditional structural analysis programs, which are typically individually packaged. The mechanics of the collaborative model is illustrated in Figure 2. In the proposed framework, the users build their structural model by using a web-based model-building service on the client site. The model then can be sent to the analysis core by using the Internet as the communication channel. Upon receiving the analysis model, the core server will perform the analysis in a collaborative manner. During the analysis, some elements can be obtained locally from the core element library. In order to find other required elements that are not existed in local element library, the *registry* will be queried to find the on-line element services, which have already been pre-registered with the core platform. After the analysis is completed, the results will be returned to the user by generating a dynamic web page in the user's web browser.

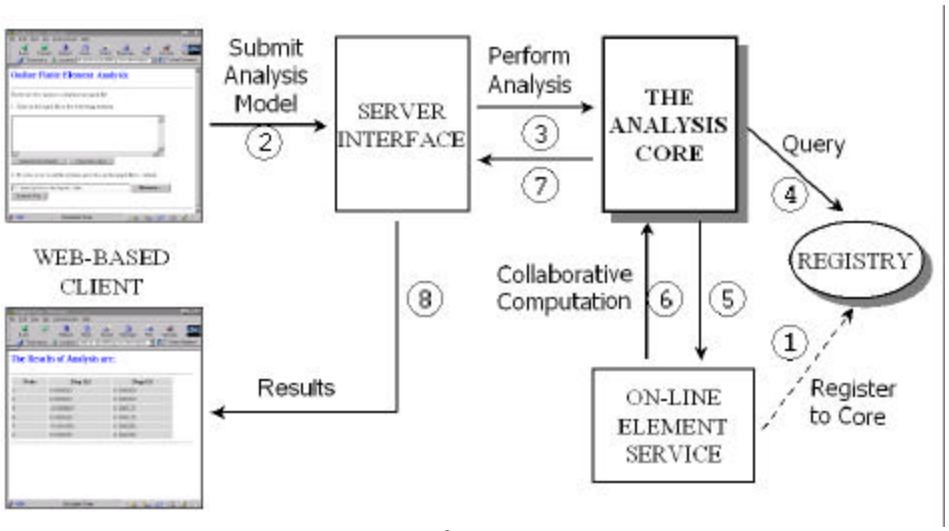


Figure 2 – Mechanics of the Collaborative Model

Since the proposed collaborative distributed system relies on the aggregate behavior of loosely coupled subsystems, component-based design and modeling can be used to facilitate the concurrent development process. The Internet has introduced a transport mechanism that allows various disparate components to interact with each other to provide more complex, complete system behavior (Hopkins 2000). We can now look at software development from a higher level of abstraction, one that treats the individual components as the target platforms. The interactions between them form the dynamic behavior of the system. By utilizing a component-based approach, the application is easy to build and is easy to modify and extend.

On-line Element Services

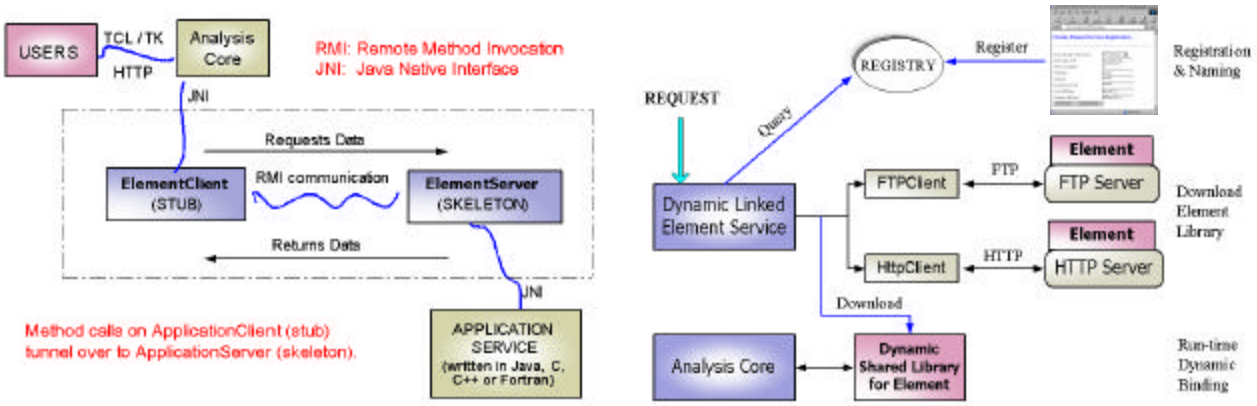
There is a standard interface/wrapper for communicating the element with the object-oriented analysis core. To introduce new elements into the analysis core generally

composes of creating subclasses of *Element* class whose common interface is defined in the analysis kernel. After the development process is finished, the new element code can be migrated into the core platform and become part of the static element library.

In addition to the traditional way of building element library for new element development, the new elements can also be developed in the form of on-line element service. As shown in Figure 3, there are two types of on-line element services. They are distributed element service and dynamic shared library service respectively. We can differentiate the two types based on where the actual computation code resides.

For the distributed element model (shown in Figure 3a), the actual element code resides on the element service provider's site and it runs as a compute engine. Whenever the core needs certain element data, for example the stiffness matrix, the core has to request the service provider through a sequence of remote procedure calls. The meaningful computation is performed at the service provider's site and the requested data can be sent back to the server after the computation.

The distributed element model is flexible and convenient, however, the drawback here is performance. The initiation of remote procedure calls is quite expensive considering that it has to be done for every element and a typical structural model has more than tens of thousands of elements. To improve performance without losing flexibility, the dynamic shared library service is proposed (shown in Figure 3b). In this model, the computation code of element is built in the form of dynamic shared library conforming to a standard interface. The shared library then will be put into an ftp server or an http server. When the core needs this element, the dynamic shared library can be downloaded from the service provider's site. The downloaded shared library will dynamically link with the analysis core during the run-time to perform structural analysis.



(a). Distributed Element Service (b). Dynamic Shared Library Service

Figure 3 – On-line Element Services

In summary, the new element can be developed in the following forms: static element library, distributed element service, and dynamic shared element library. Which form will be used for new element development is decided by the developers for their convenience. As long as the new element conforms to the standard interface, it should be able to communicate with analysis core. After the development is completed, the new element service can be registered to the *Registration and Naming Service* by telling the server its name, location, form and other information. Users don't need to worry about what kind of element service to choose and where to find the service. Even though there are three representations of elements, the choosing and binding of element services are automated and totally transparent to users.

Data Access System

Traditionally, there are two ways of obtaining the requested analysis results from structural analysis programs. One way is pre-defining all the wanted data before analysis and saving these data during analysis. A problem associated with this method is that if the users want certain data other than pre-defined ones, a complete re-analysis has to be performed to generate requested results. For nonlinear dynamic analysis of a large structure model, the re-analysis can be very expensive. The other way of obtaining analysis results is simply putting all the interim and final analysis data into files. These files can be searched for the post-processing phase of structural analysis. Obvious drawbacks of this method are substantial amount of storage space and bad performance because of the expensive searching.

To overcome the drawbacks of the traditional ways of post-processing, we propose an on-line data access system for structural analysis program. A commercial database system – Oracle 8i is used in our implementation – is employed for the efficient access of large structured data, like matrixes and vectors (Oracle 2000). In the proposed system, only selected results, not all results, will be stored into the database during the analysis. When the user wants to access the results, the request will be forwarded to the analysis core. If the data that the user wants is already stored in the database, the action is no more than a database query; otherwise, the program will automatically instantiate the required new objects to re-compute the required data. Compared to the tradition way of redoing the whole analysis to obtain certain data, the re-computation should be more efficient and only involves a small portion of the program.

Figure 4 shows the architecture of our proposed data access system. In this system, users interact with the server through web-based interface. Using dynamic HTML pages together with JavaScript code, the server will be able to generate rich and flexible content in users' browser. The server side is divided into three layers. *Database*, which is used for the storage and retrieval of selected analysis results, is the back-end of the system. Java Servlet enabled *Web Server* is deployed to receive users' requests and hand them to the *Application Server*. The *Application Server* is the middle layer for handling communication between *Web Server* and *Database* as well as doing analysis and

generating results. As shown in Figure 4, some extra functionality will be added to OpenSees in order to modify it into the *Application Server*.

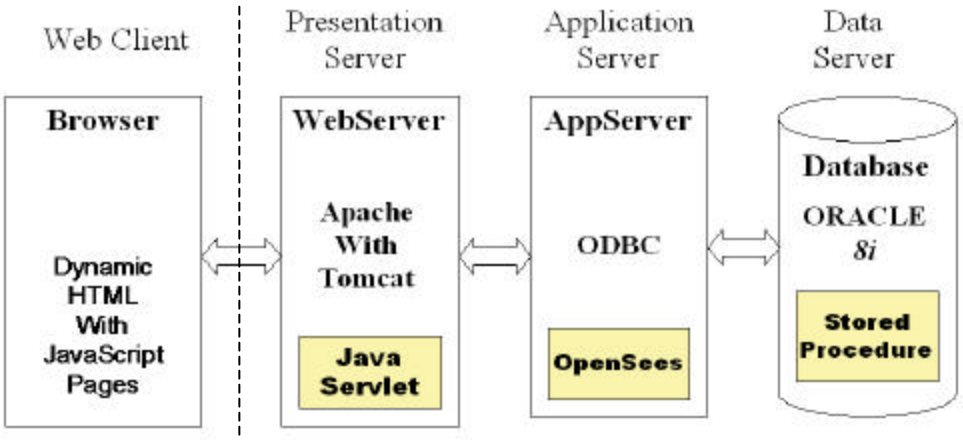


Figure 4 – Data Access System Architecure

Summary

The collaborative system implementation of a structural analysis program has at least three benefits. First, the framework provides a means of distributing services in a modular and systematic way. Users can select appropriate services and can easily replace a service if a superior service becomes available, without having to recompile the existing services being used. Second, it provides a means to integrate legacy code as one of the modular services in the infrastructure. Third, the framework alleviates the burden of managing a group of developers and their source code. Once a common communication protocol is defined, participants can write their code based on the protocol. The approach minimizes the need to constantly merge the source code written by different participants.

Most of my work for the last two years has been concentrated on testing new ideas and prototyping. I have already developed working prototypes for both on-line element services and data access system. My plan for the next step will be to continue implementation and to further improve the existing systems. For the on-line element services part, the goal is still to standardize element service interface. With the incorporation of new elements from Geotechnical Engineering field, the element interface has to be further improved and tested. I will continue the work in collaboration with researchers of UC, Berkeley and other element developers. For the data access system implementation, I plan to finalize the system design and try to improve flexibility and performance. Another possible improvement in this part is to make the data access system automatically generate results in certain formats that can be directly interpreted by Excel, Matlab, and other popular software packages.

References

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*.

Hopkins, J. (2000). "Component Primer." *Communication of the ACM, 43(10), 27-30*.

McKenna, F. (1997). "Object Oriented Finite Element Analysis: Frameworks for Analysis Algorithms and Parallel Computing." *Ph.D. Thesis, Department of Civil Engineering, University of California, Berkeley*, CA.

Oracle Corporation (2000). "Building JSP Internet Application with Oracle JDeveloper." *An Oracle Technical White Paper*. April 28, 2000.

Peng, J. and Law, K. H. (2000a), "Framework for Collaborative Structural Analysis Software Development." *Structural Congress & Expositions ASCE*. May 2000, Philadelphia, PA.

Peng, J., McKenna, F., Fenves, G. L. and Law, K. H. (2000b), "An Open Source Model for Collaborative Development of Finite Element Program." *International Conference on Computing in Civil and Building Engineering*, August 2000, Palo Alto, CA.

Santiago, E. D. (1996). "A Distributed Implementation of The Finite Element Method for Coupled Fluid Structure Problems." *Ph.D. Thesis, Department of Civil Engineering, Stanford University, Stanford, CA*.