13th World Conference on Earthquake Engineering
Vancouver, B.C., Canada
August 1-6, 2004
Paper No. 1492

AN OBJECT-ORIENTED SOFTWARE ENVIRONMENT FOR COLLABORATIVE NETWORK SIMULATION

Gregory L. FENVES[1], Frank McKENNA[2], Michael H. SCOTT[3], and Yoshikazu TAKAHASHI[4]

SUMMARY

The Open System for Earthquake Engineering Simulation (OpenSees) is a software environment for the network-based simulation of structural and geotechnical systems. The software design of the structural and geotechnical models in OpenSees is modular and hierarchical to mirror the equations of structural mechanics. The mathematical software components for computational simulation are also modular, allowing new components to be added to the framework. The ability to perform network-based structural simulation with OpenSees follows directly from the modular software design because the components necessary for network-based simulation can be added to the framework as another implementation of an object than can be communicated between processors over a network. Modules for network-based simulation derive their behavior from the existing components in OpenSees, which are also movable objects that can be sent over communication channels during a computational simulation. New networked-based applications in OpenSees include parallel computation, databases, and hybrid experimental methods.

INTRODUCTION

The simulation of a structural and geotechnical system is a crucial step in its design and assessment. The purpose of an earthquake simulation is to estimate the performance of the system under seismic excitation or loads that represent the effects of an earthquake. Structural and geotechnical simulation has a long history of development in earthquake engineering with advances moving in parallel with an increased understanding of the behavior of materials, components, and systems under cyclic loads. With the increased emphasis on performance-based earthquake engineering methodologies that are concerned with estimates of damage, there is an increasing need for more realistic models that represent the cyclic degradation of materials and components. There are significant challenges in the simulation of complex systems, such as soil-foundation-structure

[1]Professor, Department of Civil and Environmental Engineering, University of California, Berkeley, USA. Email: fenves@ce.berkeley.edu
[2]Research Engineer, Department of Civil and Environmental Engineering, University of California, Berkeley, USA. Email: fmckenna@ce.berkeley.edu
[3]Graduate Student Researcher, Department of Civil and Environmental Engineering, University of California, Berkeley, USA. Email: mhscott@ce.berkeley.edu
[4]Research Associate, Kyoto University, Yoshida-Honmachi, Kyoto, Japan. Email: yos@catfish.kuciv.kyoto-u.ac.jp

interaction for structural systems subjected to liquefaction and lateral spreading of soil, or the loss of lateral load carrying capacity of reinforced concrete frames with poorly confined columns.

In the past decade there have been rapid advances in computing and information technology. These advances include an increase in processor speed, an increase in memory, parallel computers with high-speed networks, grid-based computing, and visualization. Other fields in science and engineering have taken advantage of these computing advances, but earthquake engineering simulation development has lagged behind.

To address the needs for improved modeling capabilities for performance-based earthquake engineering while taking advantage of the rapid advances in computing, the Pacific Earthquake Engineering Research Center (PEER) is developing the Open System for Earthquake Engineering Simulation (OpenSees). The goal of the OpenSees project is to develop and utilize a new software framework for simulating the seismic response of structures and geotechnical systems to support PEER's research mission in performance-based earthquake engineering (PBEE). The methodologies for PBEE require improved models of systems, methods of simulation, particularly for degrading systems, and large scale computations, all of which OpenSees is intended to address.

OpenSees is a software framework that consists of a set of cooperating modules that can be used to construct applications, in this case for earthquake engineering simulation. The design of a software framework is based on assumptions about the applications, and OpenSees is based on the finite element method [1], which is the most general approach for the computational simulation of structural and geotechnical systems. OpenSees is designed in a modular fashion to support the finite element method with loose coupling between modules. This allows users and developers in different fields, including engineering, computer science, and numerical analysis, to develop and modify specific modules with relatively little dependence on other modules. A developer does not need to know everything that is in the framework, allowing them to concentrate on making improvements in areas of their own expertise. Furthermore, modules can be modified and optimized to take advantage of computing hardware, communication, and visualization without the changes propagating throughout the software system.

Most users of OpenSees will utilize a "glue language" to develop a model and analysis procedure for the simulation of a structural and geotechnical system. We currently use the language Tcl [2] as a method for selecting the specific modules and software objects for a simulation. There are many advantages of giving users a fully programmable, interpreted language for defining models and solution methods, including the ability to conduct parameter studies, to provide network access to data and storage, and to communicate with graphical user interfaces.

From the outset of PEER's effort, it was intended that OpenSees would be a community-based endeavor, in which the development team would grow to include a wide range of researchers and users in earthquake engineering. To allow this to happen, the OpenSees software is open-source. That is, the code is available at http://opensees.berkeley.edu for those outside the core development team to read, redistribute and modify. Also, web based applications are provided to allow users to interact with each other and other developers. This is in contrast to the traditional closed model, where only a few developers have access to the code, and there is not a support system for ongoing community-based development.

With this introduction to OpenSees, the objective of this paper is to present the software architecture and show how it addresses the goals of flexibility and extensibility for simulation applications. These characteristics are then utilized to develop network applications for simulation that are briefly described in the final section.

OPENSEES SOFTWARE ARCHITECTURE

High-level Architecture

The traditional design for finite element software is based on procedures to perform specific computations, such as material constitutive evaluation, element integration, and equation solution. Procedural-oriented design tends to de-emphasize the organization of data needed to represent the software components, such as the material and element state and the system of equations. As a result, procedural software components are often tightly bound to each other with data structures that are either stored in common memory or other global data stores. The procedures must interact with the global data structures, thereby limiting the flexibility and reusability of the software components. The high-level view of traditional software architecture is represented in Figure 1(a), in which the base code includes procedures for materials, elements, solution algorithms, and equation solvers. Many finite element programs allow users to develop elements (or materials) as long as the specific forms of the data structures are communicated with the base code. In most software the base code is tightly linked to the user interface and processors for the input, and the software is implemented for a specific compute technology (types of processors and communication methods). With the traditional design of finite element software, it is often difficult to extend or add new solution methods, solvers, interfaces with databases, and other modifications because of the dependence of the procedures on global data structures. The linking also makes it difficult to take advantage of improved and emerging compute technologies, such as solvers, parallel processing, database interfaces, and grid-based computing.

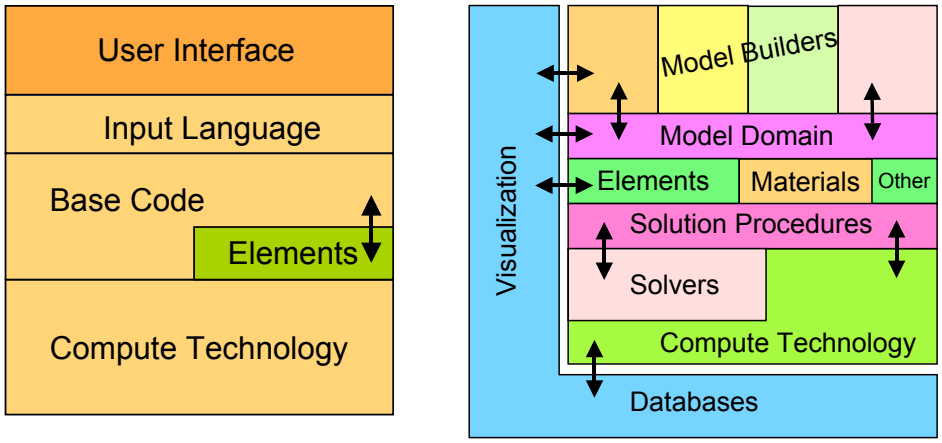
A more flexible approach for the high-level architecture of finite element software is illustrated in Figure 1(b). The concept is to provide loose coupling between the important components with an application program interface (API). The API is a set of calls that tell the component to perform specific operations. Generally, each component is responsible for maintaining the state (and hence, data representing the state) and only essential information need be communicated with other components. This approach provides flexibility for constructing simulation applications and extensibility because the components are fairly independent. A central component is the *Domain* that represents the state of a finite element model at any point in a simulation. Different *ModelBuilders* may be used to construct the *Domain* for a model, such as a text based model-building language or a graphical user interface. A *Domain* interacts with the *Elements*, *Materials,* and other aspects of the model that will be described in the following. A simulation may use one of a variety of *Solution Procedures* that invoke *Solvers* to solve systems of equations. The *Solvers* and *Solution Procedures* can take advantage of different *Compute Technologies* (multiple processors, vector processors, etc.). Several of the components interact with *Databases* and *Visualization* software for storing and viewing the results of a simulation.

A collection of components, interacting through an API, to solve a class of problems is termed a software framework. A framework allows the construction of specific applications within the assumptions about what class of problems is addressed. In this regard, OpenSees is a software framework for developing applications to simulate structural and geotechnical systems using finite element methods with the high-level components illustrated in Figure 1(b).

Object-Oriented Framework Design

The design of the OpenSees framework is based on the concept of objects. Objects represent the state and behavior of components in a system being modeled computationally. The software design process involves specifying computational abstractions and defining objects and relationships between objects to represent the abstractions [3]. An abstraction is the essential function that must be represented for other components to utilize it. An object implements the abstraction for other components to utilize, but it hides the specific data and computational procedures. Objects contain the internal data necessary to represent the state of the object and to perform the required operations on the data. This is in contrast to the procedural approach in which the operations

implemented by a procedure are defined separately from the data necessary for accomplishing the operation.



a) Traditional finite element analysis code

b) Framework of components

Figure 1. Software design approach for finite element analysis software. Arrows represent application program interfaces (API) for communication between software components.

The objects in OpenSees have been designed to represent the governing equations of mechanics using the finite element method and to use algorithms to solve the equations. The most basic example of an abstraction in finite element analysis is of an element. Abstractly, an element must provide the resisting force and tangent stiffness matrix for a specified trial displacement at the nodes defining the element. In OpenSees, the Element is a key object that is responsible for maintaining its state (as data stored in the object) independently of all other objects and for operations on the element that are functions of the state, such as computing the resisting force, computing the tangent matrix, and committing its state when accepted. In addition to objects, the software architecture must represent associations between objects. For example, to compute its resisting forces, the element object must interact with other objects that represent the constitutive behavior at integration points. As another example, a root finding object interacts with a linear equation solver as it iterates toward the solution to the equations of structural equilibrium.

An important relationship between objects is one of inheritance. Returning to the example of elements, all objects must support state determination operations, but there are many types of finite elements, such as beam-column elements for frames, planar elements (membrane and plate bending), and continuum elements for solids. Elements can be organized by dimensionality (2d or 3d) or by formulation (displacement, force, or mixed). These are examples that can be organized in an inheritance hierarchy, with increasing specificity of the operations and data needed to represent the different types of elements.

With this summary of object-oriented software design, the fundamental objects and their relationships in OpenSees are defined in Figure 2. The Domain object represents the entire state of a finite element model, which changes as a result of the Analysis object advancing the state. Domains are created by the ModelBuilder object, which can be changed at any time. Recorder objects report information from the Domain for post-processing and visualization of the simulation results. These high-level objects are each composed of more detailed information. For example, Figure 3 shows that the Domain object is an aggregation of the abstractions for a finite element model, such as

nodes, boundary conditions, loads, and constraints (single-point and multi-point). An example inheritance hierarchy is exemplified for Element, and the LoadPattern object is itself an aggregation of loads applied to elements and nodes, and single-point constraints. Figures 2 and 3 give the fundamental OpenSees objects and relationships needed to accomplish the general representation in Figure 1(b).

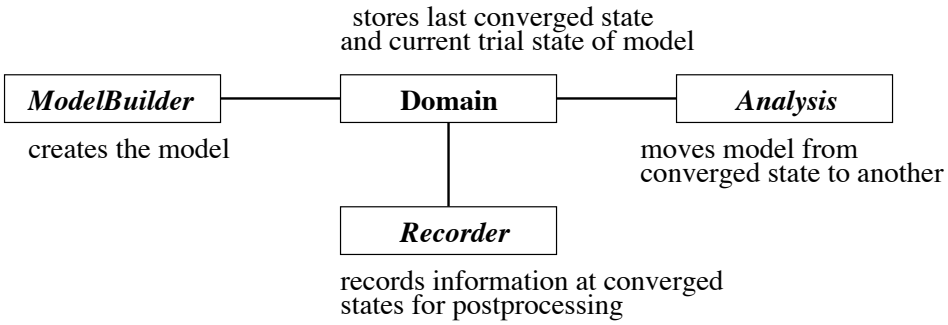


Figure 2. High-Level OpenSees objects in the software framework

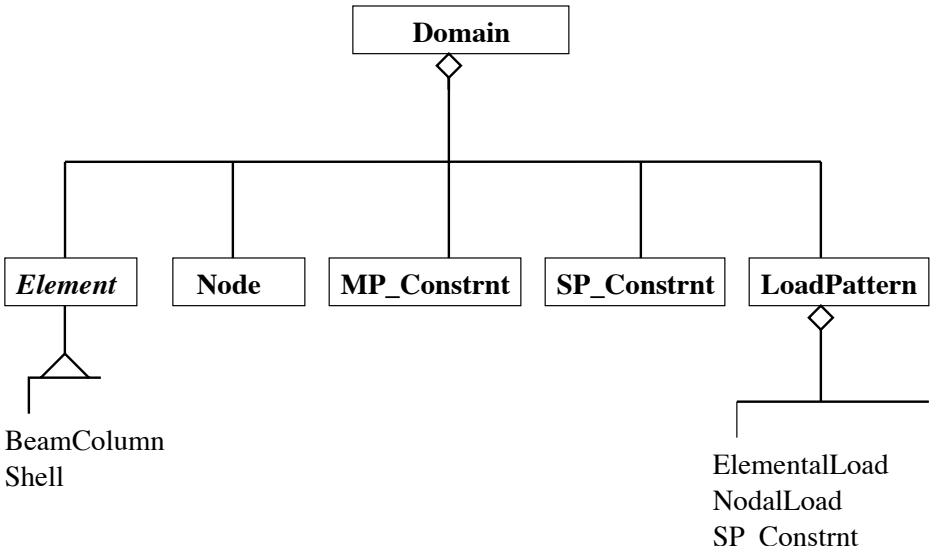


Figure 3. Domain as an aggregation of objects in a finite element model

Software Design Patterns

The previous section illustrated important types of relationships between objects such as aggregation and inheritance. The design of objects to represent an element is relatively straightforward because there is direct correspondence with the mathematical requirements for an element. As discussed in reference to Figure 1(a), traditional finite element software recognizes that an element must be clearly defined so that new elements can be added according to the API. For a flexible and extensible framework, however, there are many other objects and relationships that can be used to provide the components in Figure 1(b). The modern approach for designing the objects and relationships uses *software design patterns*. Software design patterns are standard relationships between objects that are needed to represent the aspects of the model, processes, transformations, and control. Gamma [4] has formalized the specification and application of design patterns.

To provide flexibility and extensibility, several components of OpenSees have been designed using software design patterns. One example is the aggregation of uniaxial material relationships in series and parallel to form a general relationship using a *Composite* pattern. Another example of a *Composite* pattern is the Domain object can be built-up hierarchically from sub-domains, which not only is a useful modeling approach but also supports domain decomposition solution methods for parallel processing. An application of a design pattern for a process is the *Strategy* pattern for mapping a time integration method to a solution method for solving the resulting equations discretized in time.

EXAMPLES OF OPENSEES SOFTWARE DESIGN

This section presents selected examples of the modeling approaches and applications supported by the OpenSees framework. It is not intended to be exhaustive, but rather to illustrate the flexibility provided by the software architecture for implementing a wide-range of approaches for simulation.

Modeling

Returning to the familiar example of elements, beam-column elements are the most common models of behavior in the computational simulation of frame structures. The formulation of the beam-column elements in OpenSees takes place in a basic system, free of rigid-body displacement modes. In the two-dimensional simply supported basic system, there are three basic element deformations and three basic forces, as illustrated in Figure 4. At cross-sections along the element are the section deformations and forces. The compatibility relationship between the element and section deformations, and the equilibrium relationship between the section and basic forces, depends on the beam-column element formulation, for which there are two approaches: displacement-based and force-based. Regardless of the element formulation, the element response depends on the response at each of its sections. The use of a software abstraction to represent the force-deformation response of a section facilitates the implementation of the beam-column element models in OpenSees. Each section object encapsulates the force-deformation response by either a resultant plasticity model or by the numerical integration of the material stress-strain response over the section area to describe the interaction of section forces. Therefore, the software design proceeds hierarchically from element to section, and in turn from section to material, which follows directly from the equations of structural mechanics.

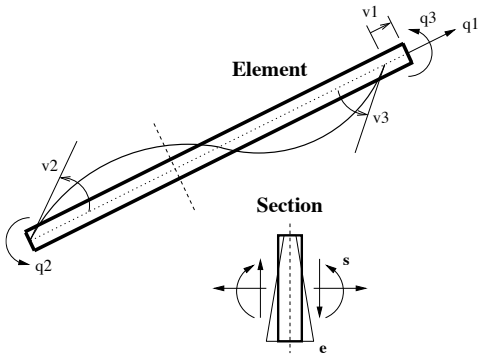


Figure 4. Basic system for 2d beam-column elements

The displacement-based formulation follows the standard finite element procedure of specifying an approximate displacement field over the element domain, from which compatible deformations are computed at each section along the element. The equilibrium relationship between the section forces and the basic forces is satisfied in an average sense or weak form under the displacement-based formulation. In contrast, the force-based formulation exactly satisfies equilibrium between the basic forces and section forces in a strong form, whereas the compatibility relationship between section deformations and basic element deformations is stated in integral form (using the principle of virtual forces). The force-based element state determination procedure, where the section deformations are computed under the condition such that section equilibrium is always satisfied, as well as the advantages of the force-based formulation over the standard displacement-based formulation, are described by Neuenhofer and Filippou [5].

One advantage of the force-based beam-column element formulation is the ease with which the shear force-deformation behavior is taken into account. The shear forces at each section along the element are computed from the basic forces in the same manner as the axial force and the bending

moments, and the relationship between the shear deformations and the element deformations come from the average compatibility inherent in the force-based formulation. The simulation of the cyclic response of a shear critical reinforced concrete column demonstrates the modeling capabilities in OpenSees, as shown by comparison of the simulation results with experimental data from Lynn [6] in Figure 5. The integration of the uniaxial stress-strain response of the steel and concrete materials over the column cross-section captures the axial-moment interaction, while a uniaxial hysteretic relationship describes the shear force-deformation behavior of the column. The hysteretic models in OpenSees are built from simpler components that represent the backbone, or envelope of the cyclic response, and the cyclic degradation of stiffness and strength.

Analysis of a Model

As shown in Figure 3, Analysis is a high-level class that is invoked on a Domain to advance the state. The object represents the mathematical abstractions for performing an analysis to solve the governing equations, as represented by the current state of the Domain. To accomplish this, an Analysis object is an aggregation of five objects that are responsible for important aspects of an analysis (illustrated in Figure 6):

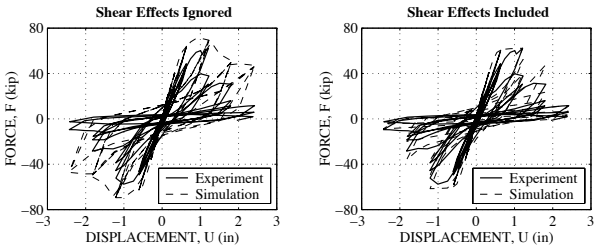


Figure 5. Comparison of experimental behavior of columns with OpenSees model including shear deformation.

Algorithm: The solution Algorithm object is responsible for orchestrating the steps performed in the analysis.

Integrator: The Integrator object is responsible for defining the contributions of the Elements and Nodes to the system of equation and for updating the response quantities at the Nodes with the appropriate responses, given the solution to the system of equations.

CHandler: The CHandler object is responsible for ensuring that the single and multi-point constraints in the Domain are enforced.

Numberer: The Numberer object is responsible for mapping equation numbers in the system of equations to the degrees-of-freedom.

SystemOfEqn: The SystemOfEqn object encapsulates the system of equations and provides operations to solve the system.

To perform a simulation, an analyst creates a procedure by specifying the objects for an Analysis. This approach offers great flexibility because the analysis can be changed by creating new objects at different times in the solution. The software design for Analysis is extensible because new procedures can be implemented by introducing specific objects to implement an Integrator, an Algorithm, or other components of an analysis.

An example of this approach for an analysis is as shown Figure 7(a) for 9-story moment-resisting steel frame using nonlinear beam-column elements with the force-formulation. Figure 7(b) shows the script for creating an Analysis object for transient analysis as an aggregation of the Newton-Raphson algorithm and Newmark time integration method. The equations are numbered using the Reverse Cuthill-McKee (RCM) method, constraints are enforced by transformation, and a banded system of equations is used. The convergence test is based on the relative change in the displacement norm. The example script is set up to invoke the analysis object for each time step. If the analysis fails because of a lack of convergence, the algorithm is switched to a modified Newton method with the initial stiffness and a large number of iterations. The combination of an analysis object and a programmable scripting language provides a great deal of flexibility in creating robust simulations. For the example, the transient analysis gives the response of the frame; Figure 7(c) shows the residual deformation of the frame and the maximum plastic rotation in the plastic hinges of the beam-column elements.

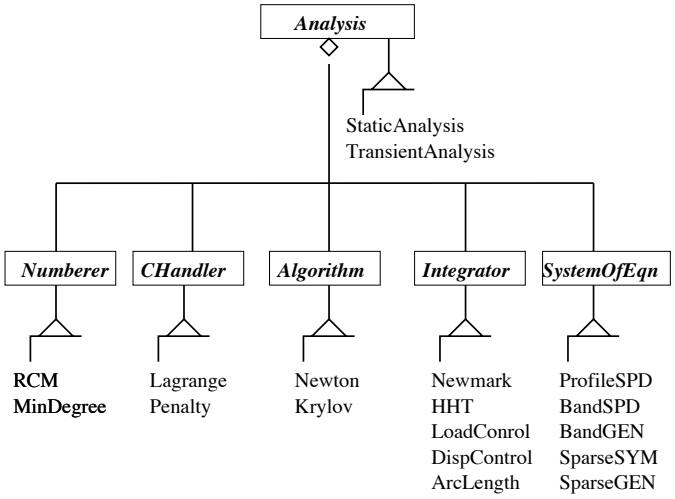
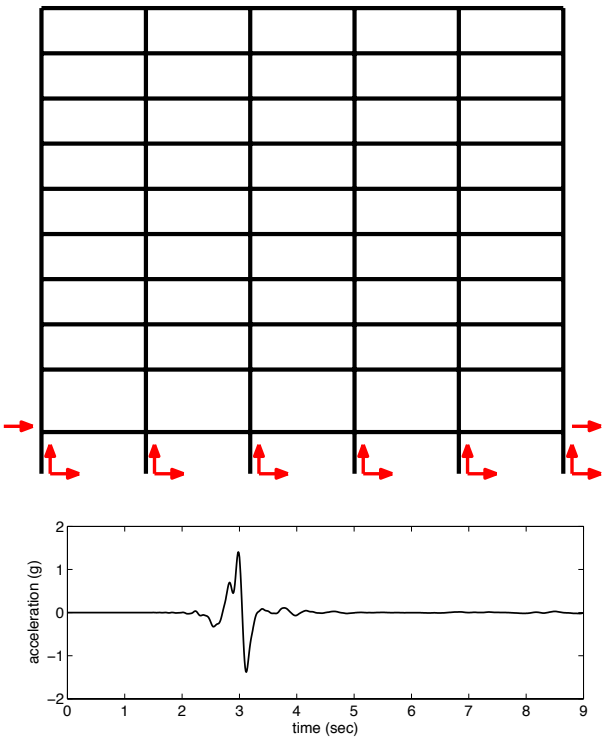


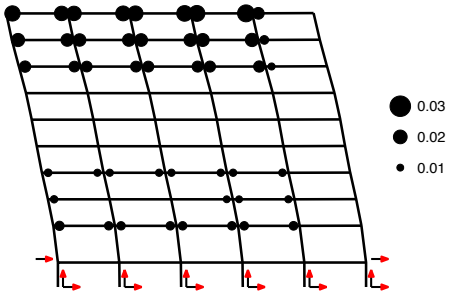
Figure 6. Analysis object as an aggregation



a) Frame model and uniform horizontal support acceleration

```
numberer RCM
constraints Transformation
system BandGeneral
test RelativeNormDispIncr 10 1.0e–6
algorithm Newton
integrator Newmark 0.5 0.25
analysis Transient
while {$t < $tFinal} {
    set ok [analyze 1 $dt}
    if {$ok != 0) {
      algorithm Newton –initial
      test RelativeNormDispIncr 1000 1.0e–6
      analyze 1 $dt
    }
  •
  •
}
```

b) Script for performing analysis



c) Plastic-rotations in members and residual deformation of frame

Figure 7. Example OpenSees analysis of 9-story steel moment-resisting frame

Sensitivity of the Solution

The simulation of the response of a structural system to an applied loading depends on the parameters that define the system, which include the properties of the materials, the geometric properties of the structural model, as well as the applied loading itself. The computation of the

gradient of the structural response with respect to these parameters lends insight into how sensitive the response is to changes in the parameters that define the structural system. In addition to the determination of the structural response sensitivity, gradient computations are essential for applications in structural reliability, optimization, and system identification where the convergence to an optimal design point depends on accurate derivatives of the structural response. To support applications for response sensitivity, structural reliability, optimization, and system identification, gradient computations have been incorporated into the OpenSees framework [7]. The computation of gradients in OpenSees is another demonstration of the extensibility of the software architecture with minimal effects on the existing components.

Two approaches to the computation of response gradients are implemented in OpenSees. First is the finite difference method (FDM) where the structural simulation is repeated with a perturbed value of a chosen parameter and the gradient is the difference in response divided by the perturbation. Although it is conceptually simple, the FDM is computationally inefficient because it requires the entire simulation to be repeated for each parameter that defines the structural model and it is subject to numerical roundoff error for small parameter perturbations. A second, more computationally efficient and numerically accurate approach to the computation of response gradients is the direct differentiation method (DDM). The DDM is based on the exact differentiation of the governing equations of structural equilibrium and compatibility [8] and it gives the response gradients for all parameters as the simulation proceeds rather than by simulations with perturbed parameter values.

The computation of the gradients of the simulated structural response by the DDM necessitates additional behaviors from the objects that define the system configuration and the element and material objects that represent the structural behavior, as well as the solution strategies that enable the structural simulation. For example, in addition to computing its resisting forces for assembly into the equations of structural equilibrium, an element object must also be able to compute the gradient of its resisting forces for assembly into the equations that yield the gradient of the structural response. The application of the DDM to compute the gradient of the simulated response for structural systems comprised of both displacement-based and force-based beam-column elements is described by Scott [9].

The gradient computations by the DDM in OpenSees are demonstrated for a cantilever beam of length L. The cantilever is loaded monotonically at its free end by a transverse load, P, and the gradient of the resulting transverse displacement, U, is computed. The moment-curvature behavior along the element length is bilinear with yield moment M_y. A single force-based beam-column element with N_p Gauss-Lobatto integration points represents the cantilever. The gradient of the load-displacement response is computed with respect to the yield moment and the element length, the results of which are shown in Figure 8. As the number of integration points increases, the computed gradients converge to the exact derivative of the closed form load-displacement response for the cantilever. In general, closed form solutions are not easily attained, and the verification of the DDM takes places by comparison with FDM computations for successively smaller parameter perturbations.

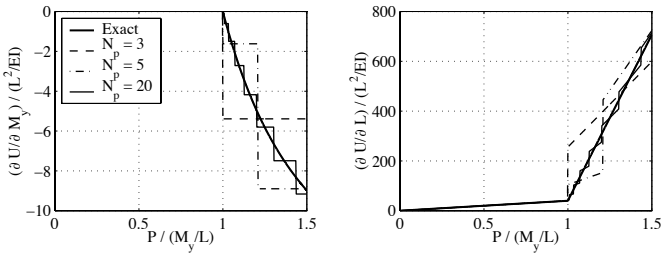


Figure 8. Example of sensitivity computation for a cantilever beam

NETWORK-BASED SIMULATION

The OpenSees framework was designed to support distributed and parallel computation [10]. Computation using a distributed memory model requires the ability to move an object from one processor (or memory space) to another, and moving requires a communication method. To accomplish these functions, most OpenSees objects are specific forms of MovableObject as illustrated in Figure 9. All MovableObjects have the ability to send and receive themselves over a communication channel. Every component required for a structural simulation, such as a beam-column element, its sections, and the material models in each section, is a MovableObject. Shown later in Figure 11 is the Channel object, which is responsible for communication of movable objects. OpenSees Channels can be implemented using standard protocols such as TCP/IP sockets or MPI. The ability to move objects between processors is a key requirement for network-based simulation applications. This section briefly describes three of these applications in OpenSees

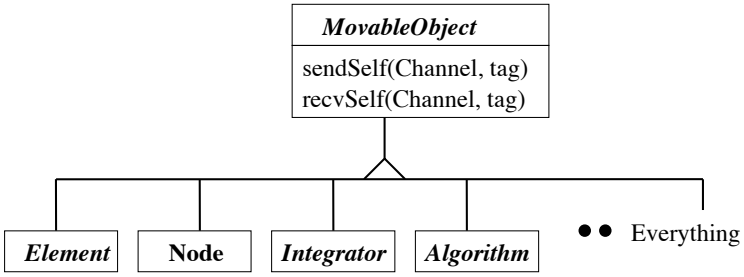


Figure 9. Inheritance hierarchy for MovableObject

Parallel Computation

Parallel processing, including distributed computing on a network, offers the ability to both solve large problems faster than they can be solved on a single processor and to solve much larger problems than can be solved on a single processor machine. Parallel processing is typically performed on a multiprocessor (shared memory) or multi-computer (distributed memory) system, where communication distances between processors are short. Distributed computing is typically performed on multiple computers in which the communication distance between machines is large and the validation of user and the reliability of communication is an issue. Distributed computing has been explored by McKenna [11] using domain-decomposition.

In a parallel/distributed program, the computation to be performed must first be broken down into a number of tasks which are then assigned to processes. The object-oriented paradigm is ideally suited to the development of parallel programs because the tasks can be identified as the invocation of the operations on objects, and the assignment of tasks that share common data to a process can be identified as assigning an object to a process. This is accomplished by the MovableObject in Figure 9, along with channels for communicating the objects between processors.

Databases

The ability to store the results of a simulation and to checkpoint the model at various stages in the analysis is essential for nonlinear analysis. Databases are used to store simulation results (response, performance measures, damage indices) for post-processing and consistent storage of state for checkpoints. In a network environment, remote access of the database facilitates collaboration among users, as illustrated in Figure 10.

To provide this functionality, OpenSees includes a Database object. As shown in Figure 11, the Database is a subclass of Channel. As with Channel, a database represents a point in the local process through which an object, typically Recorders, can send or receive information to and from a database. The advantage of subclassing a Channel object is that the sendSelf() and recvSelf() operations provided by every object in a simulation and hence a variety of database services can be provided.

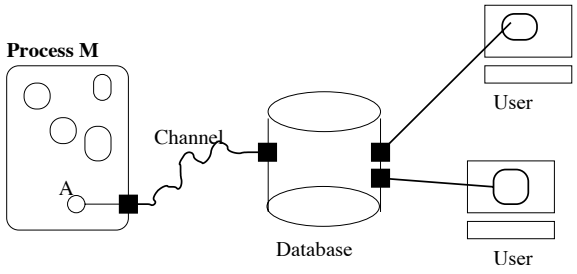


Figure 10. Schematic of communication of MovableObjects with a Database

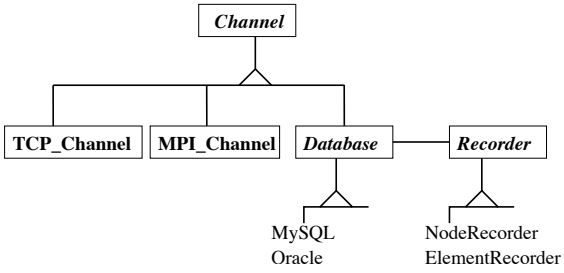


Figure 11. Inheritance hierarchy for Channel objects

Hybrid Experimentation

A final example of the network capabilities of OpenSees is the ability to implement hybrid experimental methods, which combine physical testing with computational simulation. For example, it is possible to combine a simulation model with an experimental component and apply boundary conditions to the experimental component as if it were integral with the larger system represented by the computational model. This can be accomplished within the OpenSees framework because of the object-oriented design. An Element may be an experimental specimen with actuators and data acquisition. From the viewpoint of OpenSees, as long as the experimental object can determine the resisting forces when the element is subjected to a set of displacements, there is no distinction between the physical component and mathematical elements for the rest of the system. The information is communicated over a Channel, which allows for network-based hybrid experimentation.

As a specific example of this approach, the pseudodynamic test method is a computer controlled (on-line) experimental technique for simulating the response of structures. This test method requires the software to solve the equations of motion by a step-by-step time integration scheme to determine the loading path in the next step, which can be accomplished with the OpenSees Integrator objects. Moreover since the Analysis object can manage the simulation, including the time step control, a pseudodynamic test can be conducted in the same way as a numerical simulation in OpenSees.

For the pseudodynamic test method, an ExperimentalElement is introduced so that it inherits the functionality of the Element object. Since this object relates the middleware for controlling experimental facilities, it can calculate the resisting force vector from the experimental data. If the Channel object is used in the middleware, a distributed pseudodynamic test can be carried out as a parallel simulation in OpenSees, as illustrated in Figure 12.

CONCLUSIONS

The Open System for Earthquake Engineering Simulation (OpenSees) is a software framework for developing simulation applications for structural and geotechnical systems. Its modular design allows extensibility and flexibility for modeling and analysis. This has been demonstrated by extensions of OpenSees for applications involving parallel computation, databases for simulation, and hybrid experimentation.

As open-source software, OpenSees has become a powerful collaboration mechanism for exchanging and implementing software implementations of research for nonlinear models of structural and geotechnical materials, components, and systems, and also for solution methods and equation solvers.

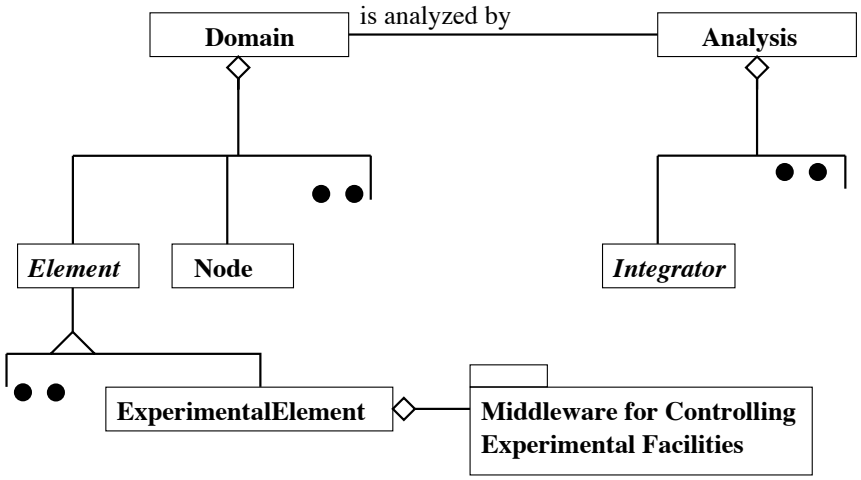


Figure 12. High-level OpenSees software architecture for hybrid experimentation

ACKNOWLEDGEMENTS

The development of OpenSees has been supported since 1997 by the Pacific Earthquake Engineering Research Center (PEER), headquartered at the University of California. PEER is an Earthquake Engineering Research Center supported by NSF under grant number 9701578. The authors thank NSF Program Directors Joy Pauschke and Lynn Preston and PEER Director Jack Moehle for their continued support of OpenSees. Significant contributions to OpenSees have been made by Filip C. Filippou (UC Berkeley), Boris Jeremic (UC Davis), Gregory G. Deierlein (Stanford), and Terje Haukaas (Univ. of British Columbia). Jaesung Park performed the simulations for the 9-story building example.

REFERENCES

1. Bathe, K.J. *Finite element procedures*. Prentice Hall, 1996.
2. Welch, B., Jones, K., and Hobbs, J. *Practical Programming in Tcl and Tk*. 4th edition, Prentice Hall, 2003.
3. Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Lorensen, W. *Object-oriented modeling and design*. Prentice Hall, 1991.
4. Gamma, E., Helm, R., Johnson, R., and Vlissides, J.. *Design patterns: elements of reusable object-oriented software*. Addison-Wesley, 1995.
5. Neuenhofer, A., and Filippou, F.C. "Geometrically nonlinear flexibility-based frame finite element," *Journal of Structural Engineering*, 1998, 124:6, 704-71.
6. Lynn, A.C., Moehle, J.P., Mahin, S.A., and Holmes, W.T. "Seismic evaluation of existing reinforced concrete buildings," *Earthquake Spectra*, 1996, 12:4, 715-739.
7. Haukaas, T. "Finite element reliability and sensitivity methods for performance-based engineering." Ph.D. dissertation, University of California, Berkeley, 2003.
8. Kleiber, M, Antunez, H., Hien, T.D., and Kowalczyk, P. *Parameter sensitivity in nonlinear mechanics*. John Wiley & Sons, 1997.
9. Scott, M.H., Franchin, P., Fenves, G.L., and Filippou, F.C. "Response sensitivity for nonlinear beam-column elements," *Journal of Structural Engineering*, 2004, accepted for publication.
10. McKenna, F. "Object-oriented finite element analysis: frameworks for analysis, algorithms and parallel computing." Ph.D. dissertation, University of California, Berkeley, 1997.
11. McKenna, F., Fenves, G.L. "An object-oriented software design for parallel structural analysis," *Proceedings*, ASCE Structures Congress 2000, Philadelphia, Pennsylvania, May 8-10, 2000.