OpenSees

Getting Started With OpenSees

Frank McKenna
UC Berkeley

OpenSees User Workshop
September 8, 2008

Outline

- Introduction To OpenSees
  - Framework
  - Tcl
  - OpenSees.exe
    - Basic Modelling
    - Basic Analysis

What is OpenSees?

The **Open S**ystem for **E**arthquake **E**ngineering **S**imulation is:

- A software *framework* for developing sequential, parallel and grid-enabled finite element applications in earthquake engineering.
- It is written primarily in the object-oriented programming language C++.
- C++ wrappers are provided so that legacy and new procedures (elements, materials, numerical routines) written in other languages C, Fortran can be used.
- Funding for OpenSees has been provided by:
  - PEER (Pacific Earthquake Enginering Research Center)
  - NEES (George E. Brown, Jr. Network for Earthquake Engineering Simulation) through NEESit.
  - NSF (National Science Foundation)

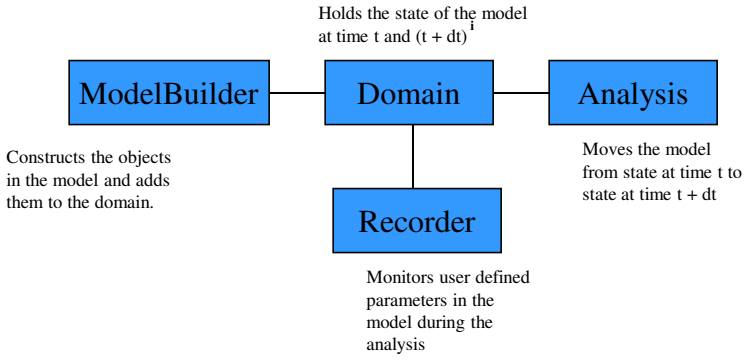
What is a Software Framework?

- A framework is **NOT an executable**.
- A ***framework*** IS a set of cooperating software components for building applications in a specific domain.

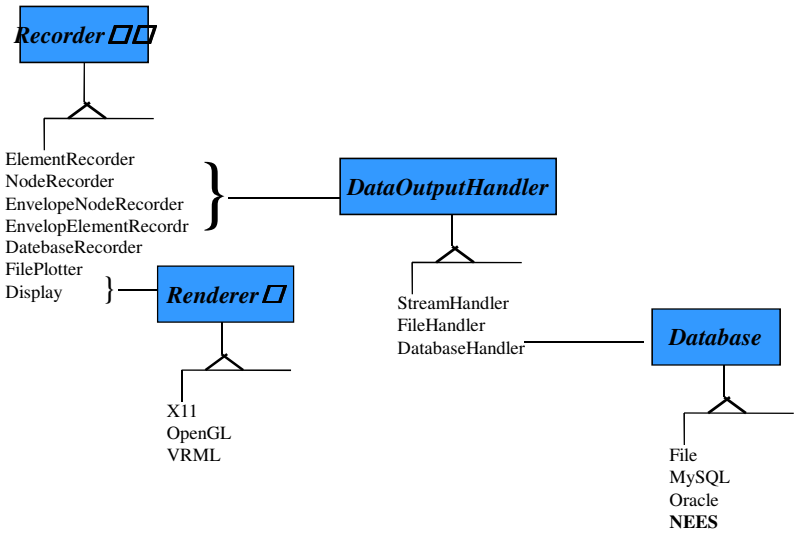
Why is OpenSees OpenSource?

- Linus's Law: "Given enough eyeballs, all bugs are shallow".
- Zero cost software attracts users!
- Prevents community from losing software. (especially true in a research community)
- Allows Community to examine new ideas. (vital in a research community were new ideas can be tested and validated by all instead of being hidden behind some theory in a paper and limited to researchers own test data)

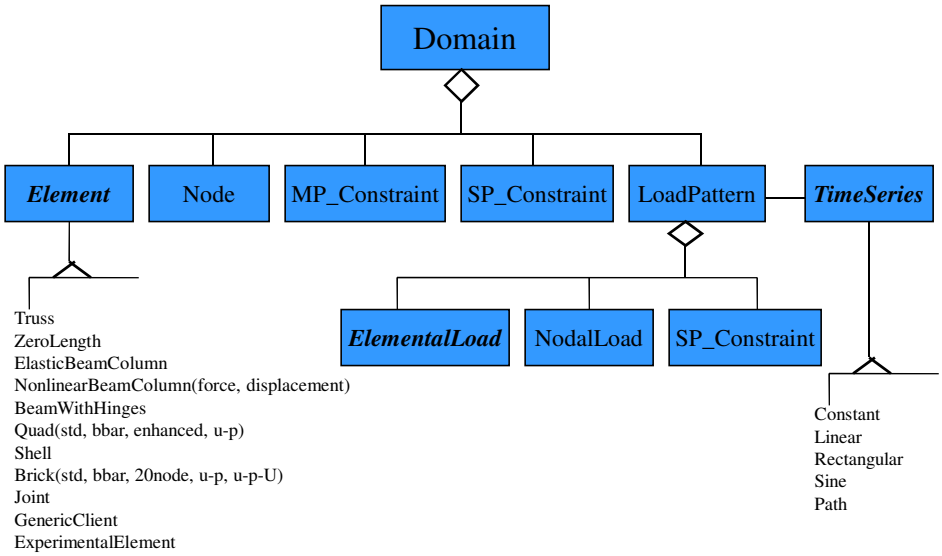
Main Abstractions in OpenSees Framework



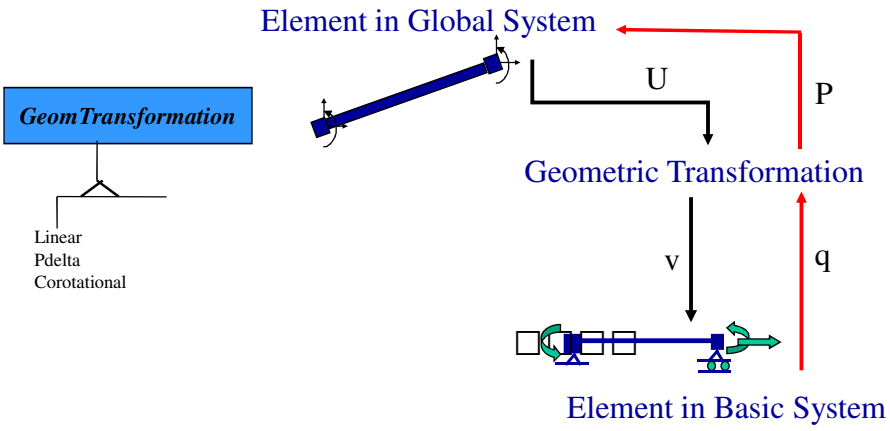
Recorder Types in OpenSees



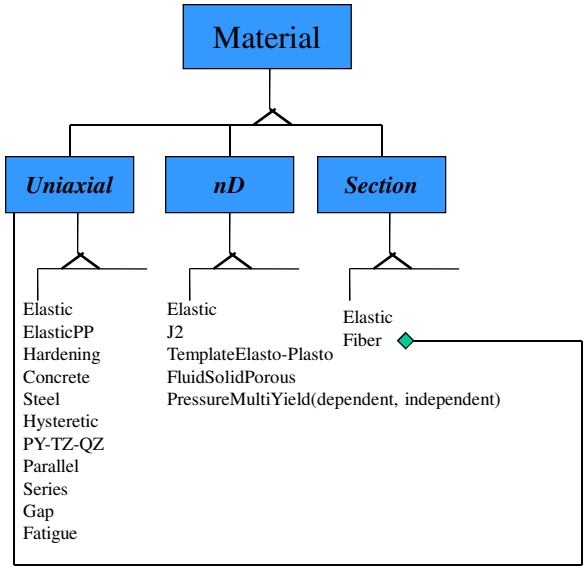
What is in a Domain?



Some Other Classes associated with Elements:



Other Classes associated with Elements:



How Do People Use OpenSees Framework?

- Provide their own main() function in C++ and link to framework.
- Use OpenSees interpreter**S**. These are extensions of the Tcl interpreter for finite element analysis which use the framework.
  1. OpenSees.exe
  2. OpsessSP.exe
  3. OpenSeesMP.exe

What is Tcl

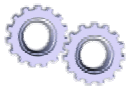
- **Tcl is a programming language**.
  - It is a string based command language.
  - Variables and variable substitution
  - Expression evaluation
  - Basic control structures (if , while, for, foreach)
  - Procedures
  - File manipulation
  - Sourcing other files.
- Comand syntax:

  command arg1 arg2 …
- Help
  1. http://dev.scriptics.com/scripting/primer.html
  2. Practical Programming in Tcl and Tk, Brent B. Welch.

Tcl Interpreters

- **wish** and **tclsh** **are tcl interpreters**.
  - Interpreters (Perl, Matlab, Ruby) are programs that execute programs written in a programming language immediately.
  - There is no separate compilation & linking.
  - An interpreted program runs slower than a compiled one.

puts “sum of 2 and 3 is [expr $2 + $3]”



sum of 2 and 3 is 5

Example Tcl

•variables & variable substitution

```
>set a 1
>1
>set b a
>a
>set b $a
>1
```

•expression evaluation

```
>expr 2 + 3
> 5
>set b [expr 2 + $b]
> 3
```

•file manipulation

```
>set fileId [open tmp w]
>??
>puts $fileId “hello”
>close $fileID
>type tmp
hello
```

•sourcing other files

```
>source Example1.tcl
```

•procedures & control structures

```
for {set i 1} {$i < 10} {incr i 1} {
        puts “i equals $i”
}
set sum 0
foreach value {1 2 3 4} {
   set sum [expr $sum + $value]
}
set $sum
>10
>proc guess {value} {
    global sum
    if {$value < $sum} {
       puts “too low”
    } else {
       if {$value > $sum} {
          puts “too high”
       } else { puts “you got it!”}
   }
}
 > guess 9
too low
```

OpenSees Interpreters

- The OpenSees interpreters are tcl interpreters which have been extended to include commands for finite element analysis:
  1. Modeling – create nodes, elements, loads and constraints
  2. Analysis – specify the analysis procedure.
  3. Output specification – specify what it is you want to monitor during the analysis.

- Being interpreters, this means that the files you create and submit to the OpenSees interpreters are not input files. You are creating and submitting PROGRAMS.

Model Generation:

*Adds the modeling commands to the interpreter.

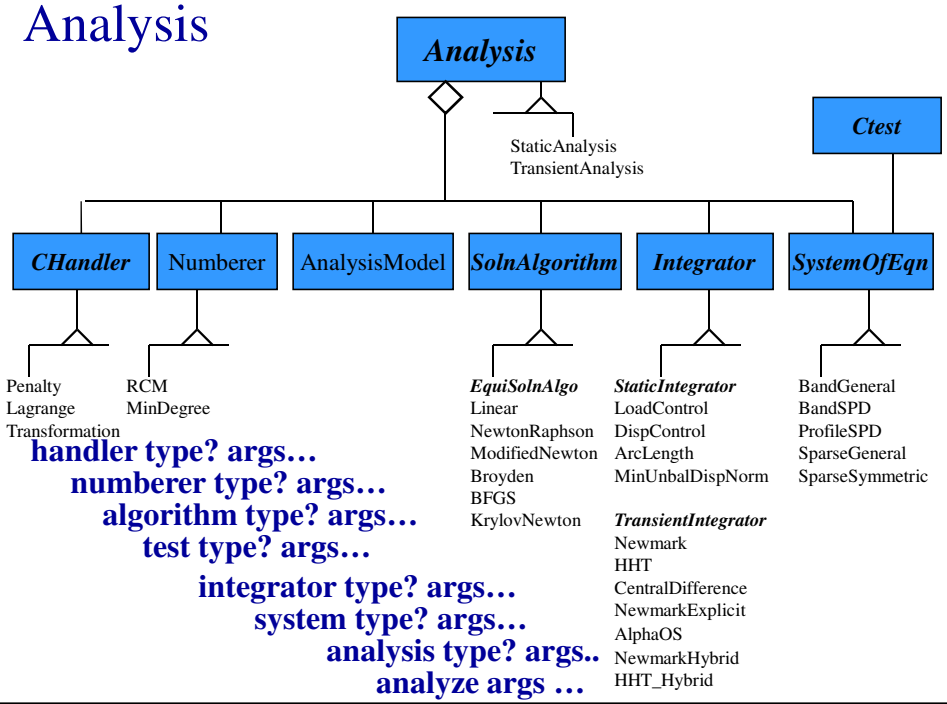
•BasicBuilder

model Basic –ndm ndm? <-ndf ndf?>

This command now adds the following commands to the interpreter:

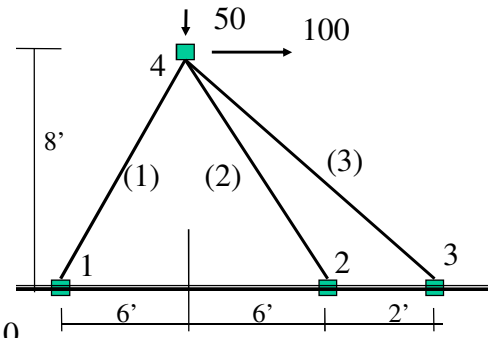
node	mass	block2D
element	fix	block3D
pattern	fixX	patch
fix	fixY	layer
equalDOF	fixZ	fiber
pattern	uniaxialMaterial	
load	nDMaterial	
eleLoad	section	
sp	geomTransf	

Analysis



Truss Example (/TrussExample/Truss.tcl):

```
model Basic -ndm -ndf 2
node 1 0.0 0.0
node 2 144.0 0.0
node 3 168.0 0.0
node 4 72.0 96.0
fix 1 1 1
fix 2 1 1
fix 3 1 1
uniaxialMaterial Elastic 1 3000.0
element truss 1 1 4 10.0 1
element truss 2 2 4   5.0 1
element truss 3 3 4   5.0 1
pattern Plain 1 “Linear” {
  load 4 100.0 -50.0
}
```



	E	A
1	3000	10
2	3000	5
3	3000	5

Example Analysis:

- Static Nonlinear Analysis with LoadControl

```
constraints Transformation
numberer RCM
system BandGeneral
test NormDispIncr 1.0e-6 6 2
algorithm Newton
integrator LoadControl 0.1
analysis Static
analyze 10
```

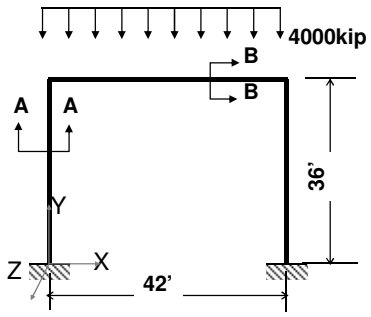
- Transient Nonlinear Analysis with Newmark

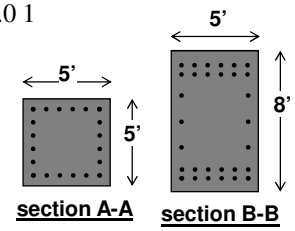
```
constraints Transformation
numberer RCM
system BandGeneral
test NormDispIncr 1.0e-6 6 2
algorithm Newton
integrator Newmark 0.5 0.25
analysis Transient
analyze 2000 0.01
```

Plane Frame (/LinearFrame/example.tcl):

```
model Basic -ndm 2 -ndf 3
node 1    0.0    0.0
node 2 504.0    0.0
node 3    0.0 432.0
node 4 504.0 432.0
fix 1 1 1 1
fix 2 1 1 1
mass 3 5.18 0.0 0.0
mass 4 5.18 0.0 0.0
geomTransf Linear 1
element elasticBeamColumn 1 1 3 3600.0 4227.0 1080000.0  1
element elasticBeamColumn 1 4 3 3600.0 4227.0 1080000.0  1
element elasticBeamColumn 3 3 4 5760.0 4227.0  4423680.0 1
pattern Plain 1 Linear {
  load 3 0.0 -2000.0 -168074.0
  load 4 0.0 -2000.0  168074.0
}
```

source example.tcl





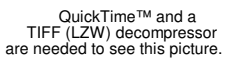
Gravity Load Analysis

```
# first source in the model
source example.tcl

# create analysis & perform analysis
constraints transformation
numberer RCM
system BandGeneral
test NormDispIncr 1.0e-6 6 2
algorithm Newton
integrator LoadControl 0.1
analysis Static
analyze 10

# look at what happened to node 3
print node 3
```

* We will call this file example1.tcl for future examples



Cyclic Lateral Load Analysis

```
# first source in the model and do gravity load analysis
source example1.tcl

# set gravity loads constant & reset time in domain
loadConst -time 0.0

# create load pattern for lateral loads
pattern Plain 2 Linear {
        load 3 200.0 0.0 0.0
        load 4 200.0 0.0 0.0
}

# do some cyclic analysis
foreach {numSteps stepSize} {10 0.1 10 -0.1 10 -0.1 10 0.1 10 0.1} {
        integrator LoadControl $stepSize
        analyze $numSteps

        set time [getTime]
        set disp [nodeDisp 3 1]
        puts "Time: $time Displacement $disp"
}
```

* We will call this file example2.tcl for future examples

QuickTime™ and a
TIFF (LZW) decompressor
are needed to see this picture.

Any Questions?