OpenSees

Introduction to OpenSees and Tcl

Frank McKenna
UC Berkeley

OpenSees Parallel Workshop
Berkeley, CA

What is OpenSees?

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

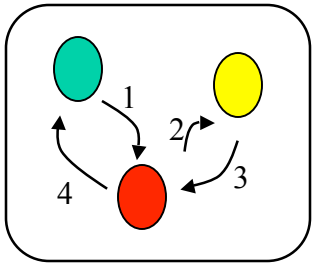
- A software *framework* for developing sequential, parallel and grid-enabled finite element applications in earthquake engineering.
- It is written primarily in the object-oriented programming language C++.
- C++ wrappers are provided so that legacy and new procedures (elements, materials, numerical 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.
- In object-oriented terms, a framework is a collection of abstract and concrete classes. The abstract classes define how the classes interact, the concrete classes are classes users can use to build applications.
- A framework dictates the architecture of the application. A good design will cover the majority of applications within the domain.
- Loose-coupling of components within the framework is essential for extensibility and re-usability of the applications.

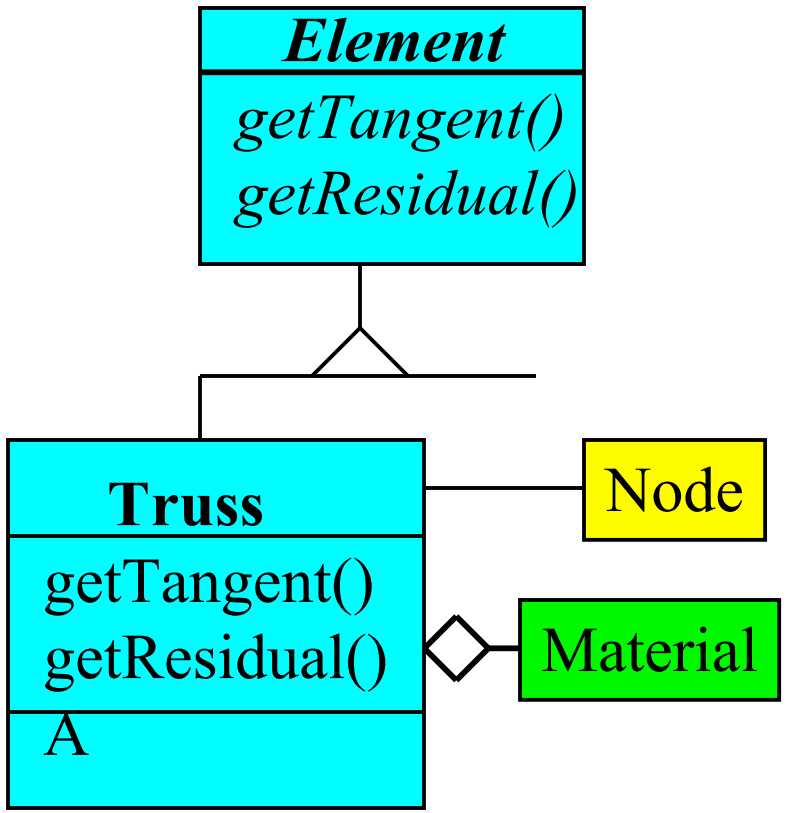
Object-Oriented Programming

- In object-oriented programming, the program is seen as a collection of objects. Each object is capable of receiving messages, processing data, and sending messages to other objects.

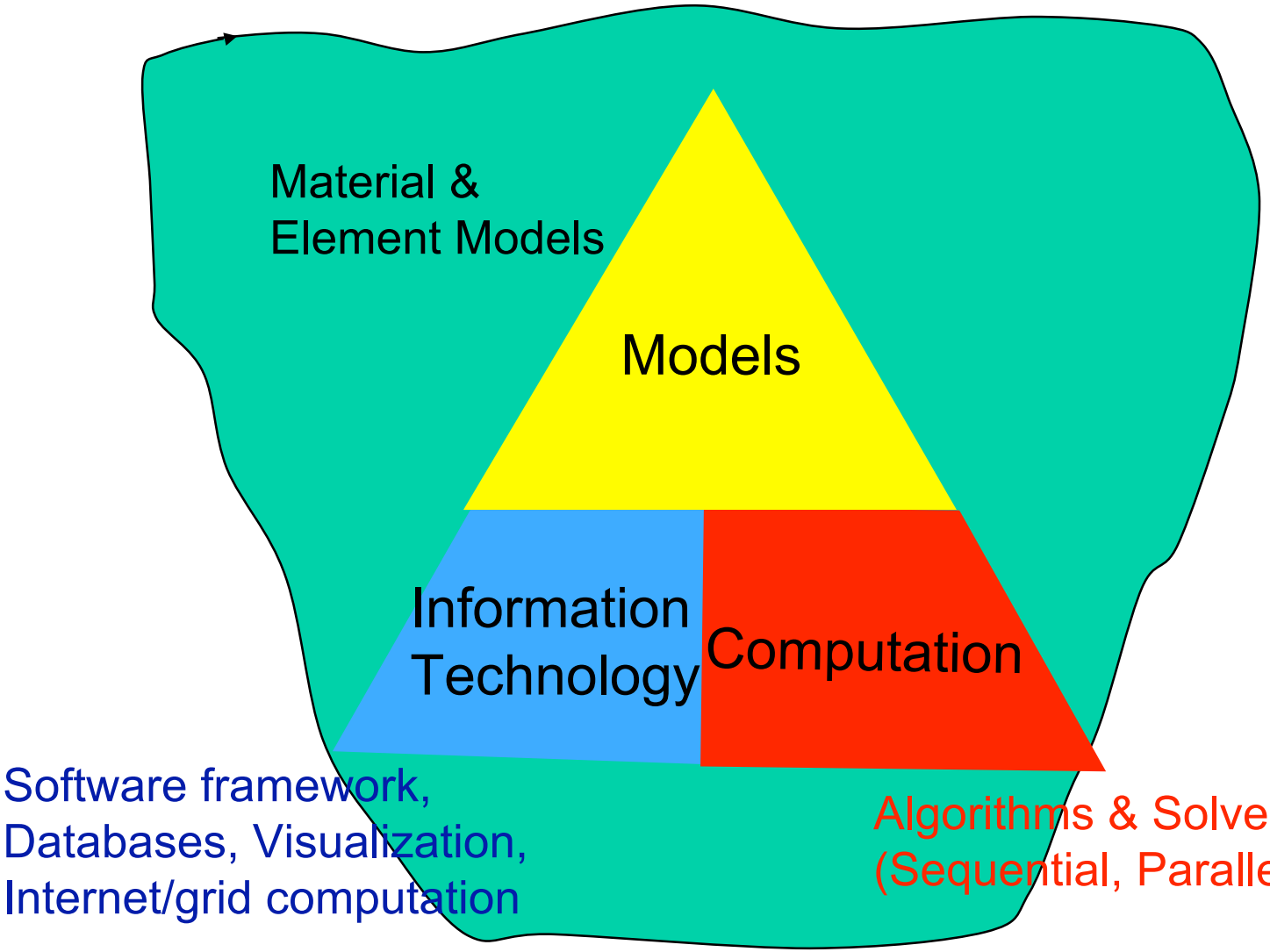


- Each object is of a particular type or Class. The class defines the data and methods of an object.

- 3 basic relationships between classes
  1. IS-A (Truss is an Element)
  2. KNOWS-A (Truss knows a Node)
  3. HAS-A (Truss has a Material)

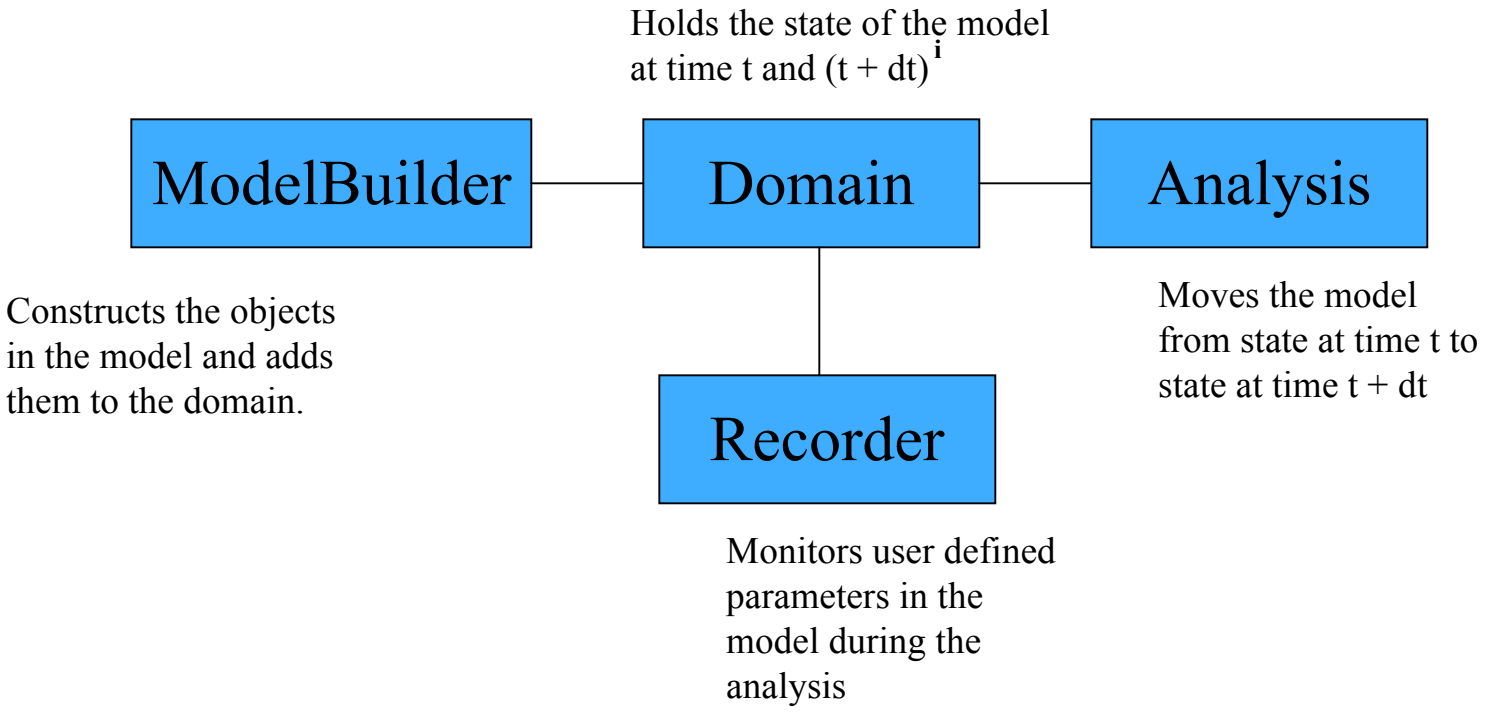


Conceptual Approach for Simulation

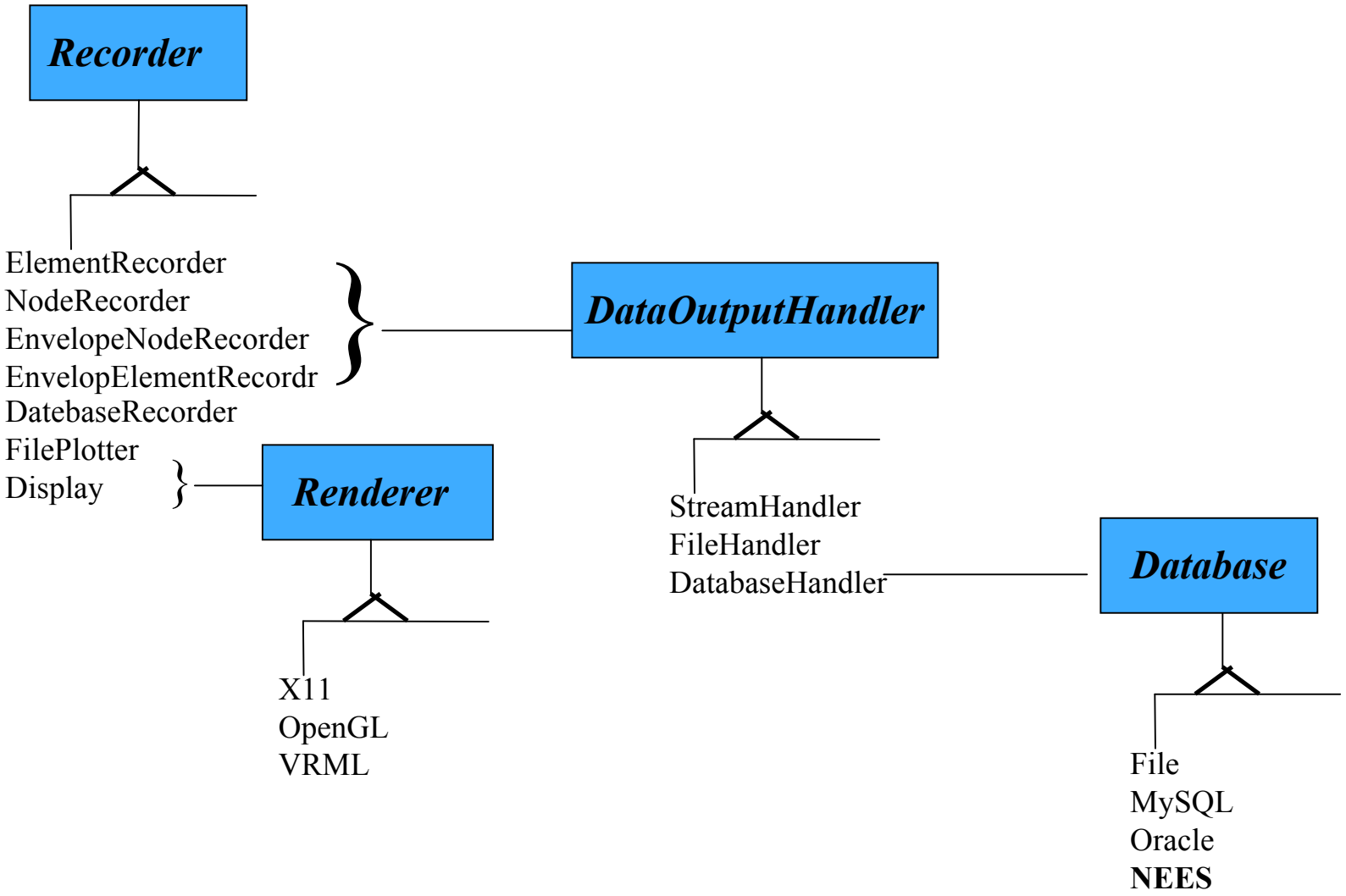


Simulation Framework

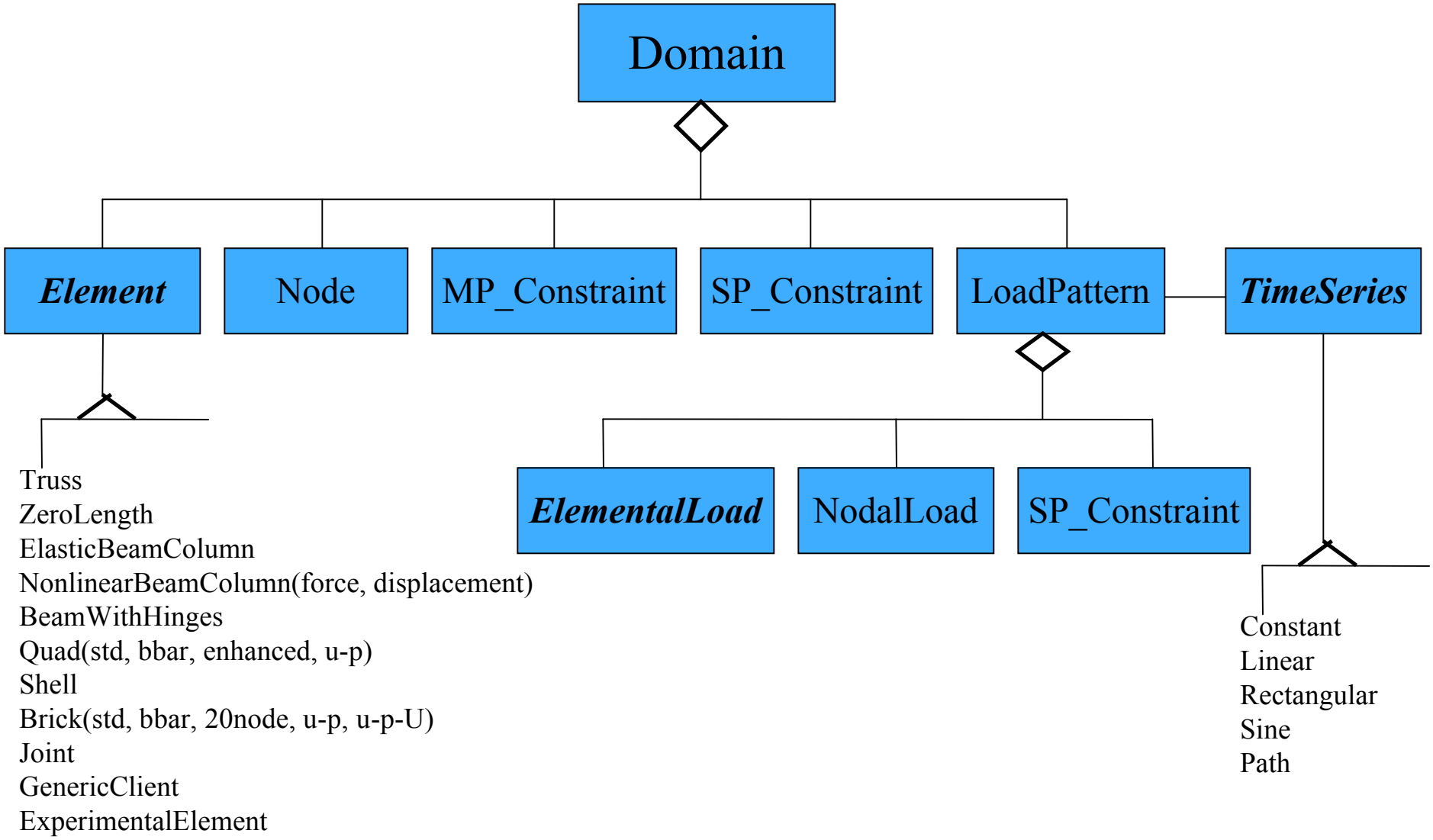
Main Abstractions in OpenSees



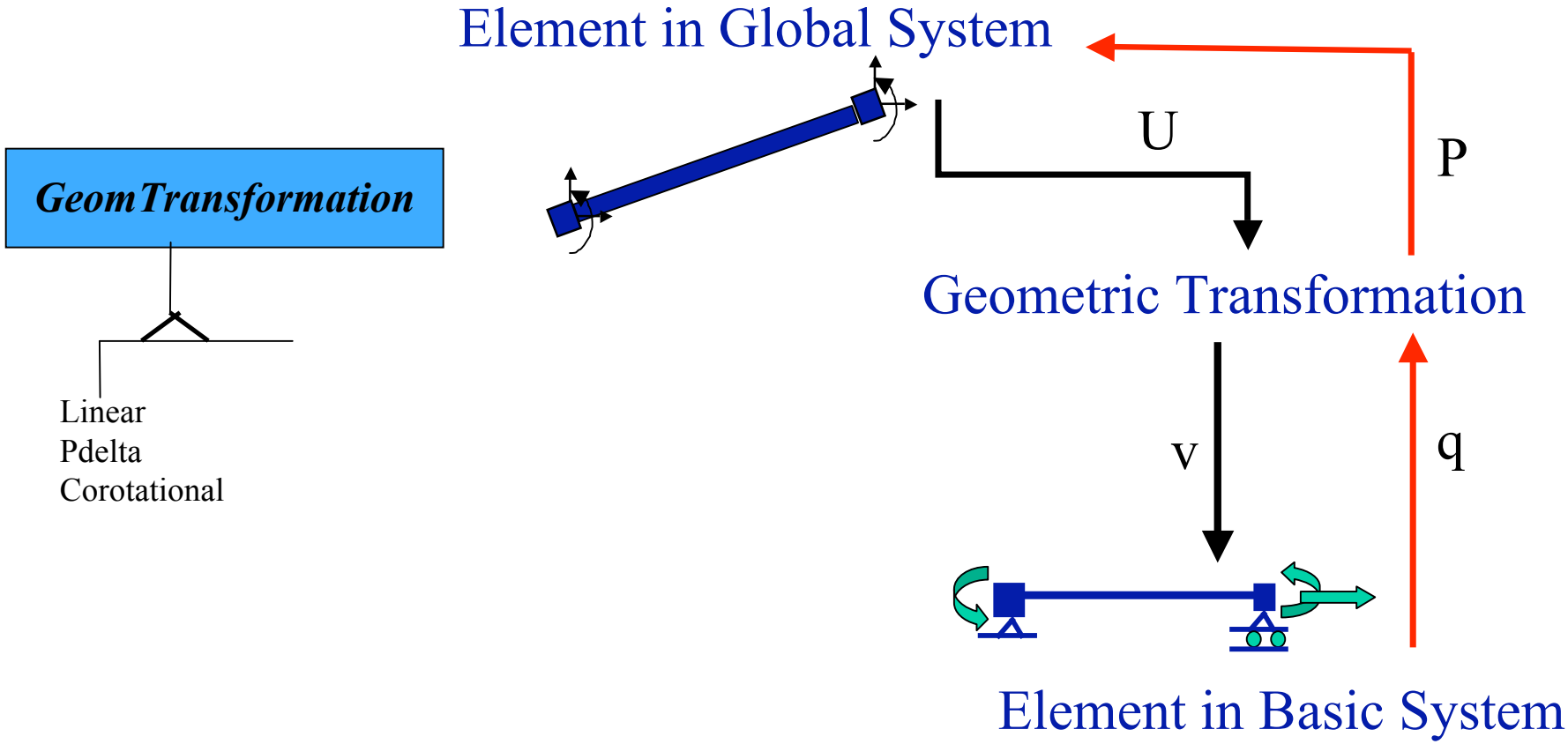
Recorder Types in OpenSees



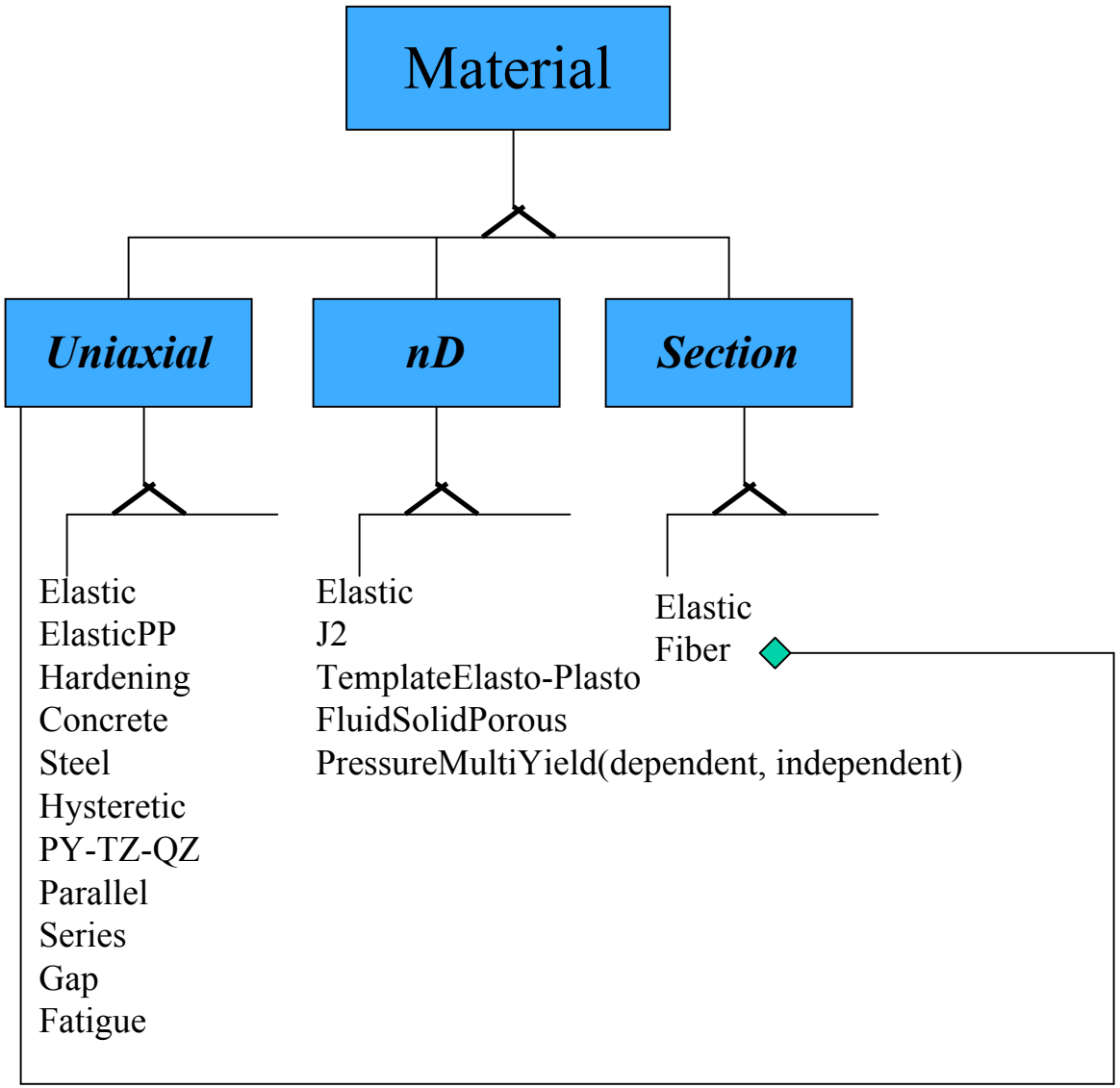
What is in a Domain?



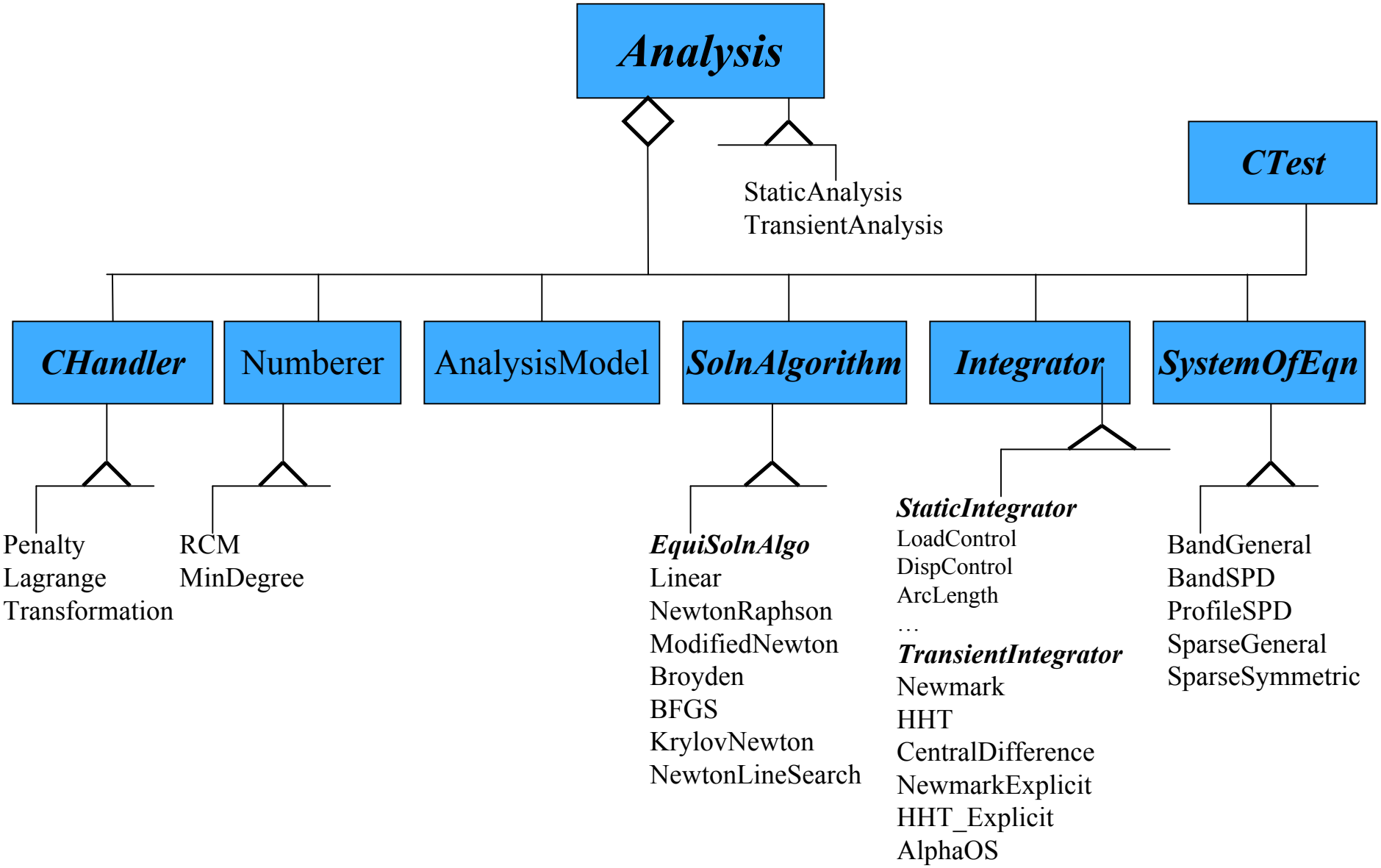
Some Other Classes associated with Elements:



Other Classes associated with Elements:



What is an Analysis?



How Do People Use 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.

OpenSees.exe

- OpenSees.exe is an extension of the Tcl interpreter for finite element analysis which uses this framework. It is an example of an application that can be developed using the framework.

What is Tcl

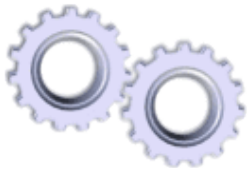
- **Tcl is a programming language**.
  - It is a string based command language.
  - Variables and variable substitution
  - Expression evaluation
  - Basic control structures (if , while, for, foreach)
  - Procedures
  - File manipulation
  - Sourcing other files.
- Comand syntax:

  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
```

Commands to Tcl for OpenSees

- For OpenSees we have added commands to Tcl 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.

Model Generation:

*Adds the modelling commands to the interpreter.

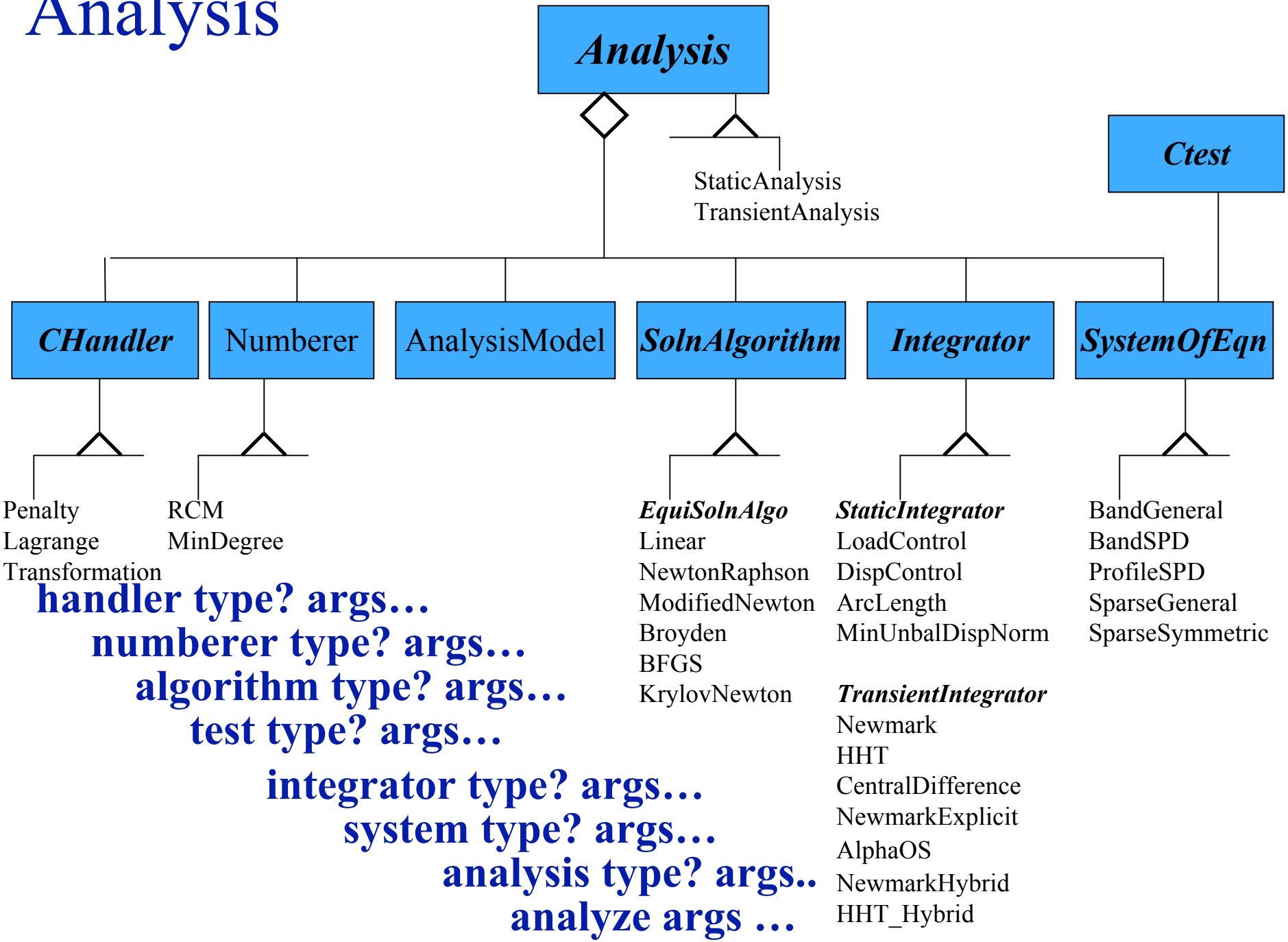
•BasicBuilder

model Basic –ndm ndm? <-ndf ndf?>

This command now adds the following commands to the interpreter:

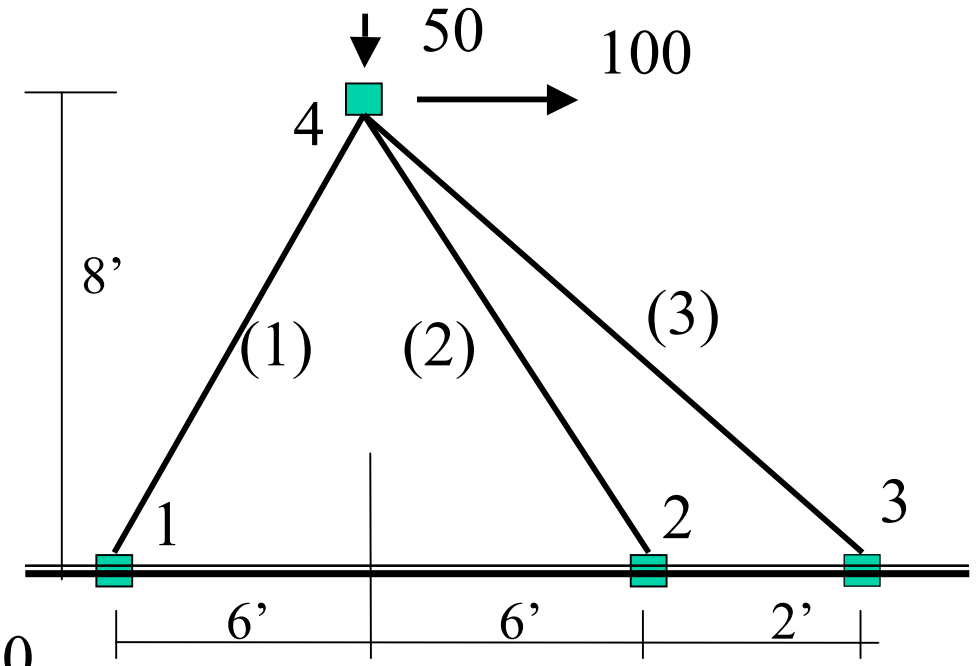
node	mass	block2D
element	fix	block3D
pattern	fixX	patch
fix	fixY	layer
equalDOF	fixZ	fiber
pattern	uniaxialMaterial	
load	nDMaterial	
eleLoad	section	
sp	geomTransf	

Analysis


Analysis
StaticAnalysis
TransientAnalysis
Ctest
CHandler
Numberer
AnalysisModel
SolnAlgorithm
Integrator
SystemOfEqn
Penalty
Lagrange
Transformation
RCM
MinDegree
EquiSolnAlgo
Linear
NewtonRaphson
ModifiedNewton
Broyden
BFGS
KrylovNewton
StaticIntegrator
LoadControl
DispControl
ArcLength
MinUnbalDispNorm
TransientIntegrator
Newmark
HHT
CentralDifference
NewmarkExplicit
AlphaOS
NewmarkHybrid
HHT_Hybrid
BandGeneral
BandSPD
ProfileSPD
SparseGeneral
SparseSymmetric
handler type? args…
numberer type? args…
algorithm type? args…
test type? args…
integrator type? args…
system type? args…
analysis type? args..
analyze args …

Example Model:

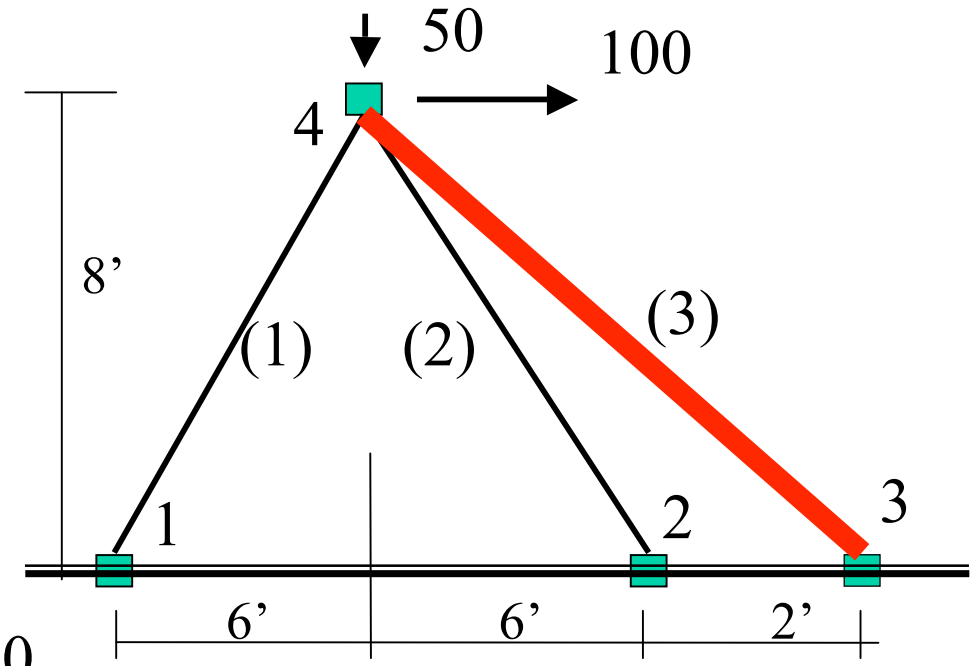
```
model Basic -ndm -ndf 2
node 1 0.0 0.0
node 2 144.0 0.0
node 3 168.0 0.0
node 4 72.0 96.0
fix 1 1 1
fix 2 1 1
fix 3 1 1
uniaxialMaterial Elastic 1 3000.0
element truss 1 1 4 10.0 1
element truss 2 2 4   5.0 1
element truss 3 3 4   5.0 1
pattern Plain 1 “Linear” {
   load 4 100.0 -50.0
}
```



	E	A
1	3000	10
2	3000	5
3	3000	5

Example Model:

```
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 genericClient 3 -node 2 3 -dof 1 2 -server 8090 127.0.0.1
pattern Plain 1 “Linear” {
   load 4 100.0 -50.0
}
```



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
```

Commands that Return Values

- analyze command

  The analyze command returns 0 if successful.
  It returns a negative number if not

  set ok [analyze numIter <ΔT>]

- getTime command

  The getTime command returns pseudo time in Domain.

  set currentTime [getTime]

- nodeDisp command

  The nodeDisp command returns a nodal displacement.

  set disp [nodeDisp node dof]

Example Usage – Displacement Control

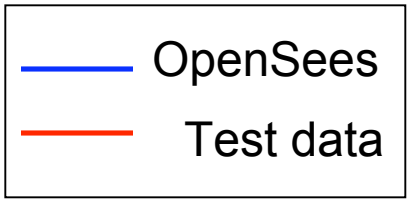
```
set maxU 15.0; set dU 0.1
constraints transformation
numberer RCM
system BandGeneral
test NormDispIncr 1.0e-6 6 2
algorithm Newton
integrator DispControl  3 1 $dU
analysis Static
set ok 0
set currentDisp 0.0
while {$ok == 0 && $currentDisp < $maxU} {
        set ok [analyze 1]
        if {$ok != 0} {
            test NormDispIncr 1.0e-6 1000 1
            algorithm ModifiedNewton –initial
            set ok [analyze 1]
            test NormDispIncr 1.0e-6 6 2
            algorithm Newton
        }
        set currentDisp [nodeDisp 3 1]
}
```

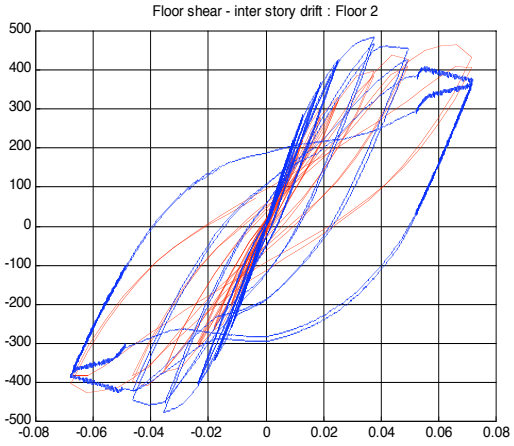
Example Usage – Transient Analysis

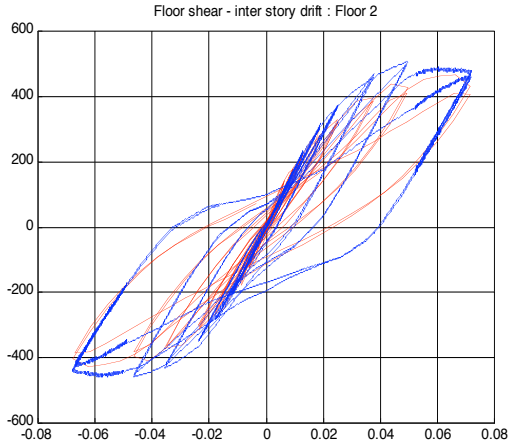
```
set tFinal 15.0;  set dT 0.01;
constraints  Transformation
numberer RCM
system BandGeneral
test NormDispIncr 1.0e-6 6 2
algorithm Newton
integrator Newmark 0.5 0.25
analysis Transient
set ok 0
set currentTime 0.0
while {$ok == 0 && $currenTime < $tFinal} {
        set ok [analyze 1 $dT]
        if {$ok != 0} {
            test NormDispIncr 1.0e-6 1000 1
            algorithm ModifiedNewton –initial
            set ok [analyze 1 $dT]
            test NormDispIncr 1.0e-6 6 2
            algorithm Newton
        }
        set currentTime [getTime]
}
```

And Why do Finite Element Analysis

NCEER frame tested at the Taiwan facility







Centerline model and model with joint comparison

And Finally Why 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)