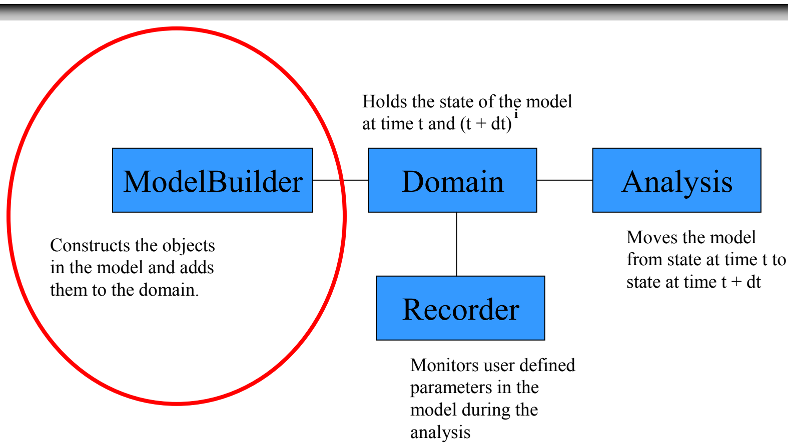
OpenSees: Modelling I

Frank McKenna

2003 Users Workshop
Berkeley, CA
August 21-22, 2003

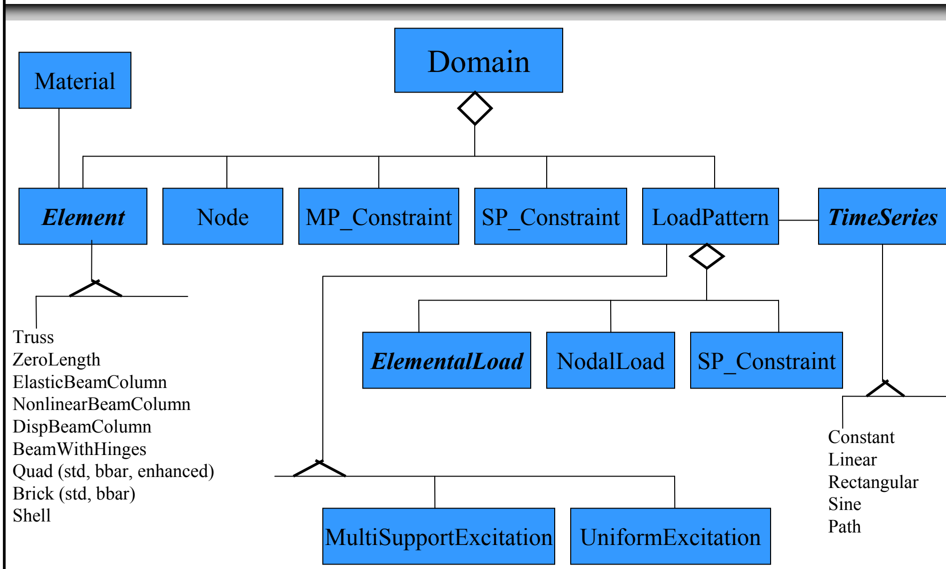
Sponsored by the National Science Foundation
through the Pacific Earthquake Engineering Research Center

Main Abstractions in OpenSees



Today we focus on model generation with the OpenSees interpreter, i.e. We will create elements/nodes/etc. and add them to the Domain/Model

Domain – only a container



* It is the model builder which constructs these objects.

model command:

*Adds the modelling commands to the interpreter.

- BasicBuilder

$\boxed{\textit{model BasicBuilder –ndm ndm? <-ndf ndf?>}}$

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	

node command:

*Adds nodes to the domain.

node $nodeTag (ndm $coords) <-mass (ndf $massValues)>

Did you forget to add mass when defining the node?

1. *mass $nodeTag (ndf $massValues)*

2. *wipe*

and start again!

sp constraints:

- to specify movement at individual nodal dof's **Uc_i = value**

•Homogeneous Constraints **$Uc_i = 0$**

Use flags to specify constraint at each dof at a node:

1 – fixed, 0 – free

fix nodeTag? (ndf constraintFlags?)

fixX xCoordinate? (ndf constraintFlags?) <-tol $tol>

fixY xCoordinate? (ndf constraintFlags?) <-tol $tol>

fixZ xCoordinate? (ndf constraintFlags?) <-tol $tol>

•Non-Homogeneous Constraints **Uc_i = value**

sp nodeTag? dofTag? dofValue?

mp constraints:

- to specify how a node moves relative to another node

$$\mathbf{U_c = C_{rc}\, U_r}$$

equalDOF nodeTagR? nodeTagC? dof1? dof2? …

$$[\mathbf{I}]\ [\mathbf{U_{r_i}}] = [\mathbf{U_{c_i}}]$$

rigidDiaphragm perpDirn? nodeTagR? nodeTagC1? nodeTagC2? …

$$\begin{bmatrix} 1 & 0 & -d2 \\ 0 & 1 & d1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_{\theta 3} \end{bmatrix} = \begin{bmatrix} U_1 \\ U_2 \\ U_{\theta 3} \end{bmatrix}$$

rigidLink type? nodeTagR? nodeTagC?

valid types:

•rod – for translational dof

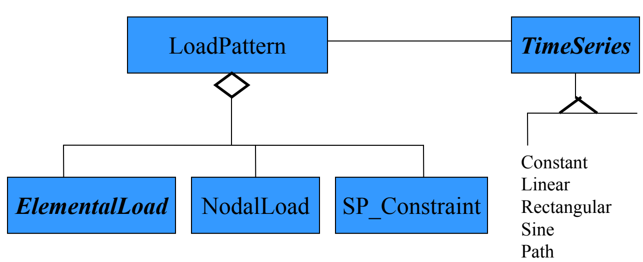
$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} U_d \\ U_\theta \end{bmatrix} = \begin{bmatrix} U_d \\ U_\theta \end{bmatrix}$$

•beam – for all dof

$$\begin{bmatrix} 1 & d \\ 0 & 1 \end{bmatrix} \begin{bmatrix} U_d \\ U_\theta \end{bmatrix} = \begin{bmatrix} U_d \\ U_\theta \end{bmatrix}$$

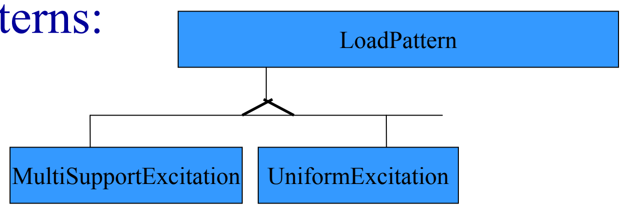
loads & load patterns:

* to add loads to the domain



pattern Plain $tag (TimeSeriesType arguments) {
load $nodeTag (ndf $loadValues)
sp $nodeTag dof? value?
eleLoad type? …
….
}

more patterns:



- Uniform Excitation – specfy ground acceleration

pattern UniformExcitation $tag $dir -accel (TimeSeriesType arguments)

- Multi-Support Excitation – specify nodal response at supports

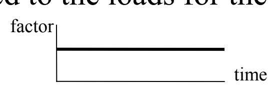
pattern MultipleSupport $tag {
groundMotion (groundMotion-command arguments)
imposedMotion $nodeTag $dirn $gMotionTag
…
}

groundMotion $gMotionTag Plain <-accel (accelSeriesType accelArgs)> <-vel (velSeriesType velArgs)> <-disp (dispSeriesType dispArgs)> <-int (IntegratorType intArgs)>

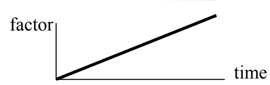
time series:

- specifies the load factor applied to the loads for the domain time

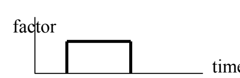
Constant <-factor $cFactor>



Linear <-factor $cFactor>



Rectangular $tStart $tFinish <-factor $cFactor>



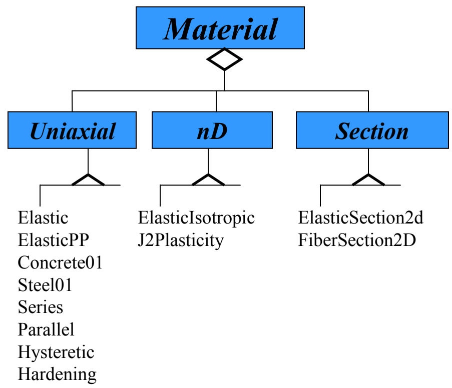
Sine $tStart $tFinish $period <-shift $shift> <-factor $cFactor>

Series -dt $dt -filePath $fileName <-factor $cFactor>

Series -fileTime $fileName1 -filePath $fileName2 <-factor $cFactor>

material commands:

- Adds the materials to the model builder.
- Each element makes copies of it's own materials



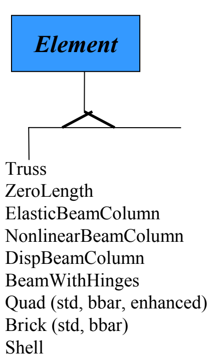
uniaxialMaterial $type ($args)

nDMaterial $type ($args)

section $type ($args)

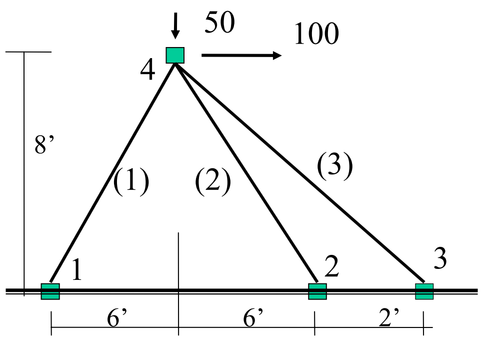
element commands:

- Adds the elements to the model.



element $type ($args)

Example 1: Simple Truss



	E	A
1	3000	10
2	3000	5
3	3000	5

OpenSees