

OpenSees: Analysis

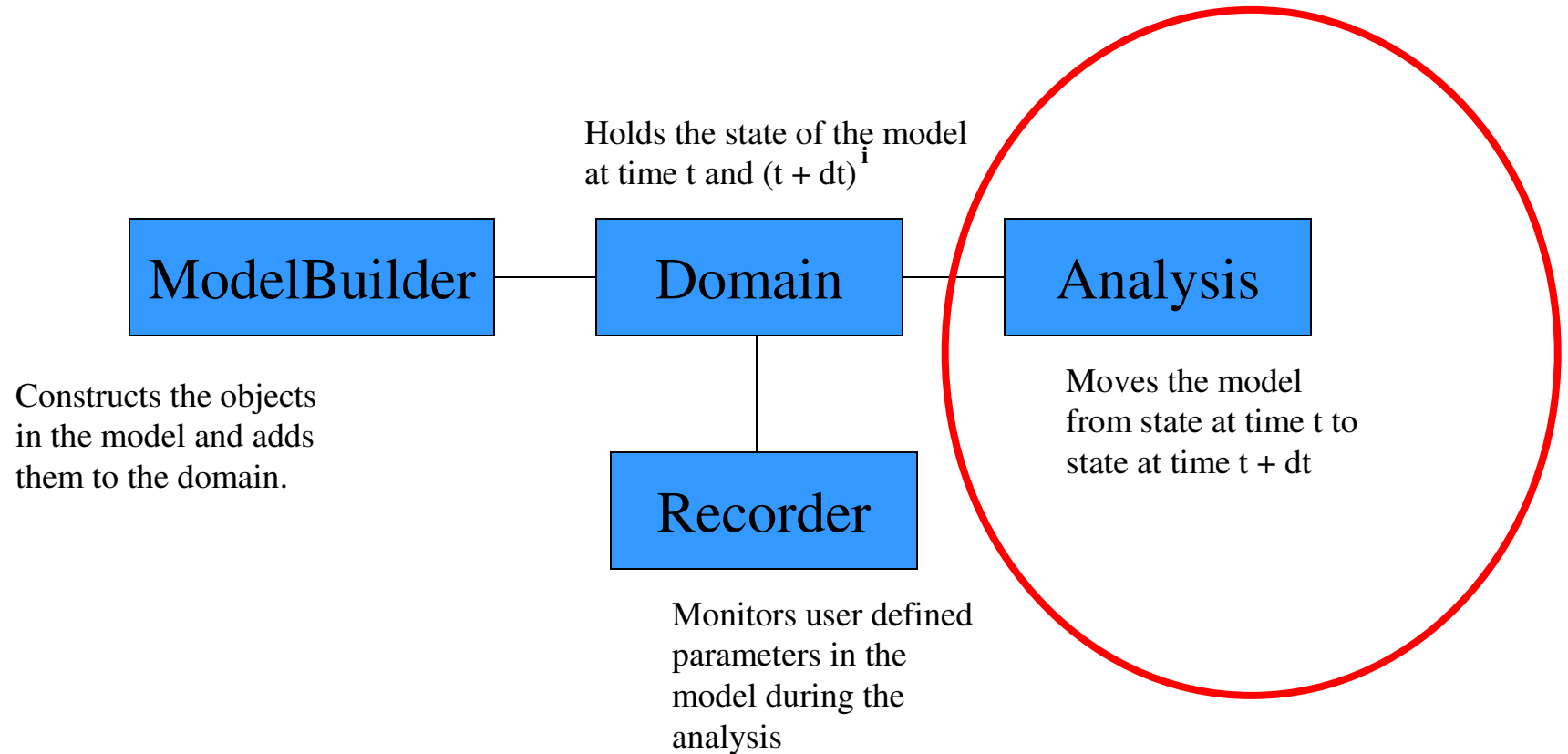
Frank McKenna

OpenSees Developers Workshop
Berkeley, CA
August 15, 2006

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

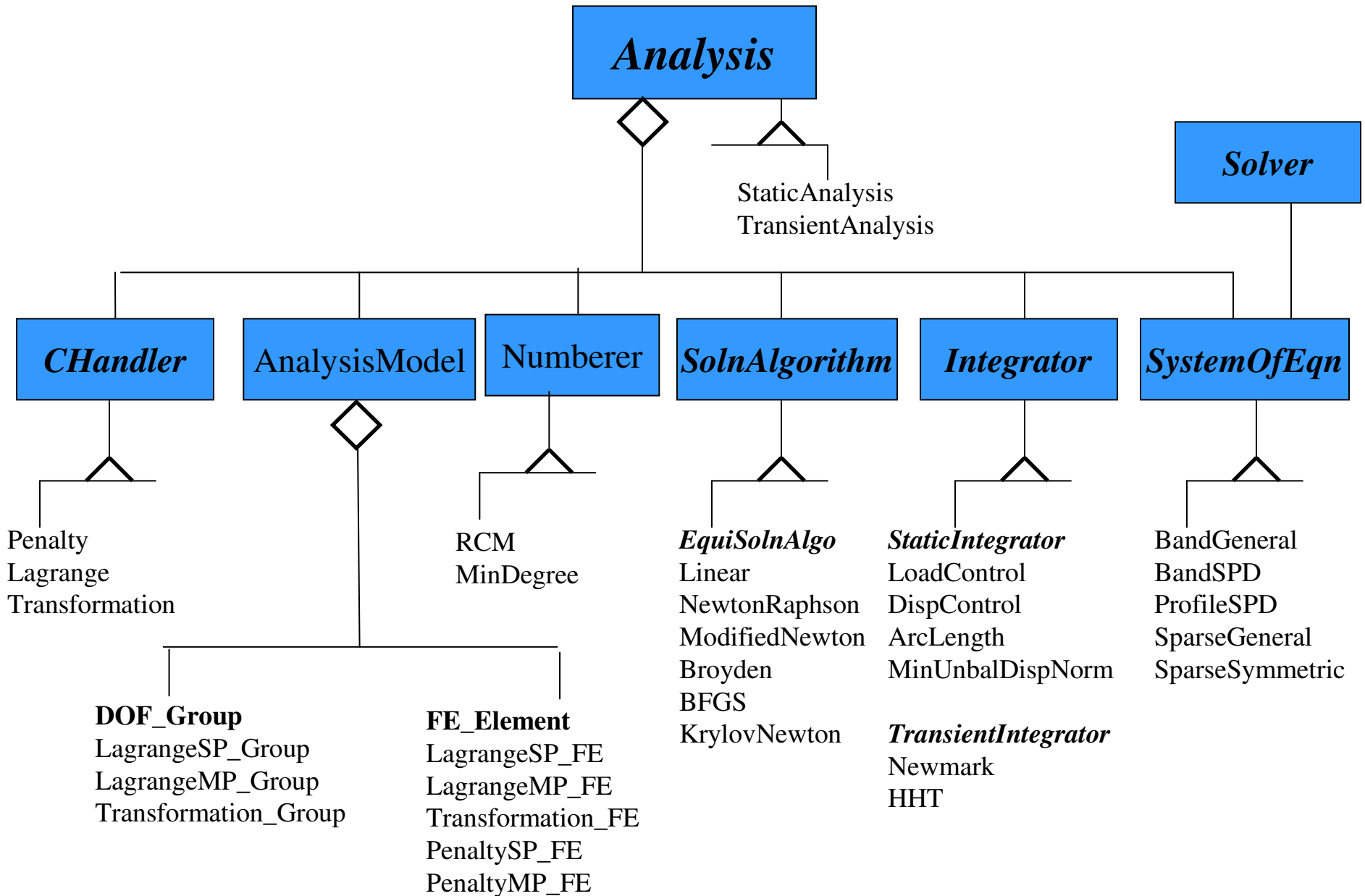


Main Abstractions in OpenSees



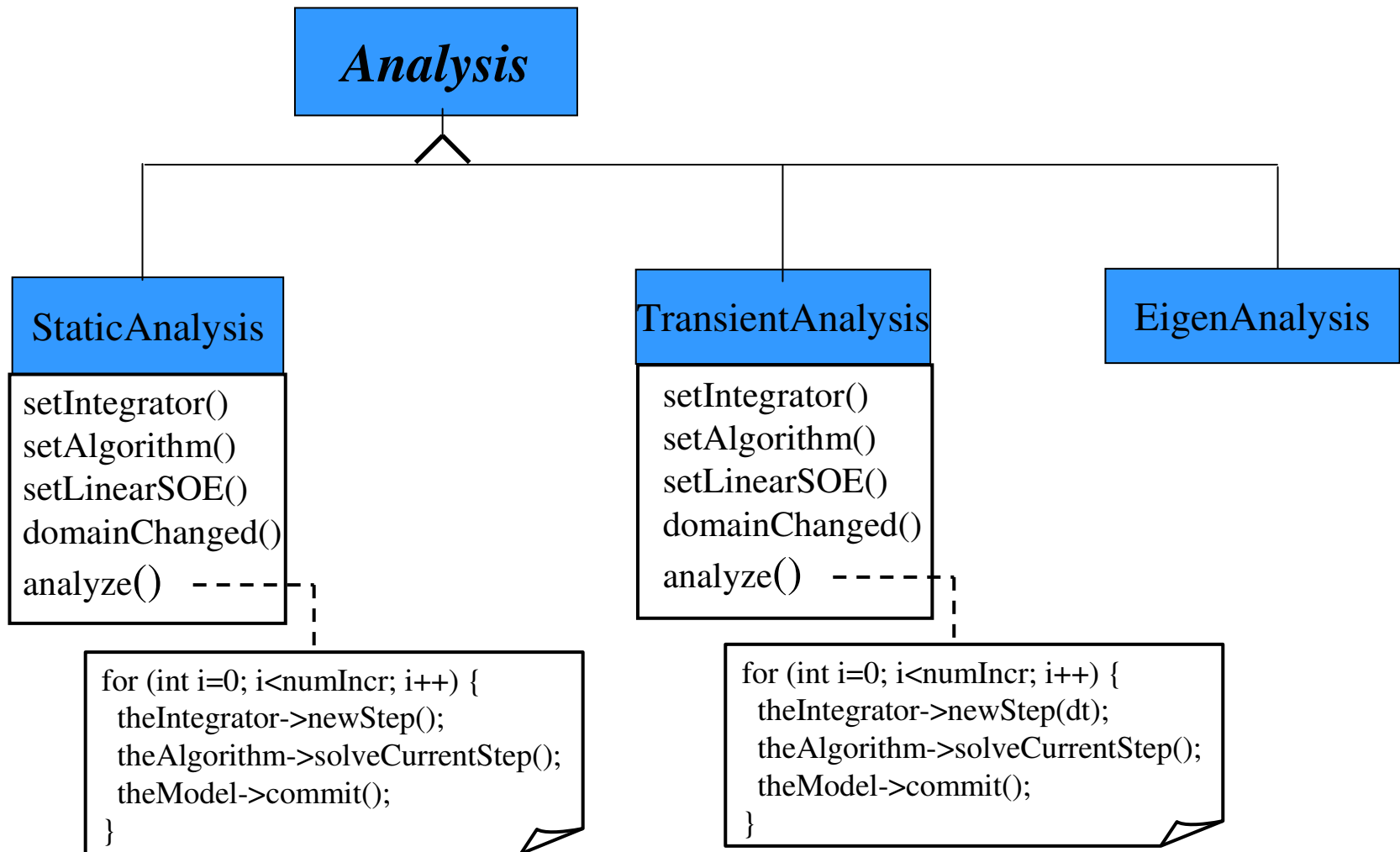
In this presentation we focus on the ANALYSIS

OpenSees Analysis



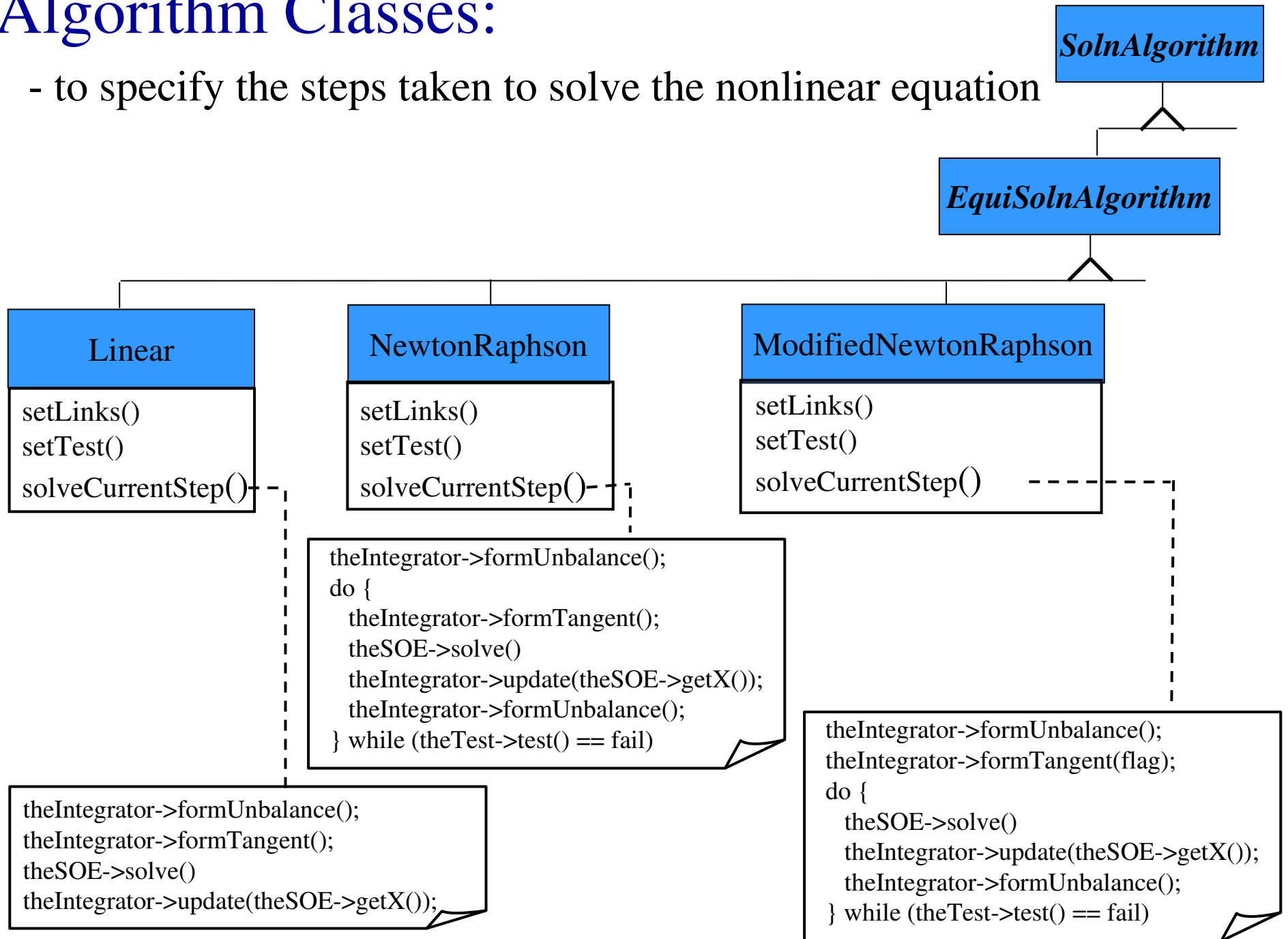
Analysis Classes:

- to update the state of the Domain



Algorithm Classes:

- to specify the steps taken to solve the nonlinear equation



Integrator Classes:

- determines the predictive step for time $t+\Delta t$
- specifies the tangent matrix and residual vector at any iteration
- determines the corrective step based on ΔU

•Transient Integrator for Use in Transient Analysis

Nonlinear equation of the form:

$$\mathbf{R}(\mathbf{U}, \dot{\mathbf{U}}, \ddot{\mathbf{U}}) = \mathbf{P}(t) - \mathbf{F}_I(\ddot{\mathbf{U}}) - \mathbf{F}_R(\mathbf{U}, \dot{\mathbf{U}})$$

•Static Integrators for Use in Static Analysis

Nonlinear equation of the form:

$$\mathbf{R}(\mathbf{U}, \lambda) = \lambda \mathbf{P}^* - \mathbf{F}_R(\mathbf{U})$$

- Load Control $\lambda_n = \lambda_{n-1} + \Delta\lambda$
- Displacement Control $\mathbf{U}_{jn} = \mathbf{U}_{j,n-1} + \Delta\mathbf{U}_j$
- Arc Length $\Delta\mathbf{U}_n^T \Delta\mathbf{U}_n + \alpha^2 \Delta\lambda_n^2 = \Delta s^2$



StaticIntegrator

LoadControl

DispControl

ArcLength

MinUnbalDispNorm

TransientIntegrator

Newmark

HHT

CentralDifference (X3)

ConstraintHandler Classes:

- to specify how the constraints are enforced

$$U_c = C_{rc} U_r$$

$$C U = 0$$

$$T U_r = [U_r \ U_c]^T$$

ConstraintHandler

Transformation

handle()

Lagrange

handle()

Penalty

handle()

forall nodes in Domain
if not constrained
construct and add DOF_Group
else
construct and add TDOF_Group
forall elements in Domain
if a node not constrained
construct and add FE_Element
else
construct and add TFE_Element

forall nodes in Domain
construct and add DOF_Group
forall elements in Domain
construct and add FE_Element
forall SP_Constraints in Domain
construct and add LagrangeSP_FE and
LagrangeDOF_Group
forall MP_Constraints in Domain
construct and add LagrangeMP({}_FE
and a LagrangeDOF_Group

forall nodes in Domain
construct and add DOF_Group
forall elements in Domain
construct and add FE_Element
forall SP_Constraints in Domain
construct and add PenaltySP_FE
forall MP_Constraints in Domain
construct and add PenaltyMP({}_FE

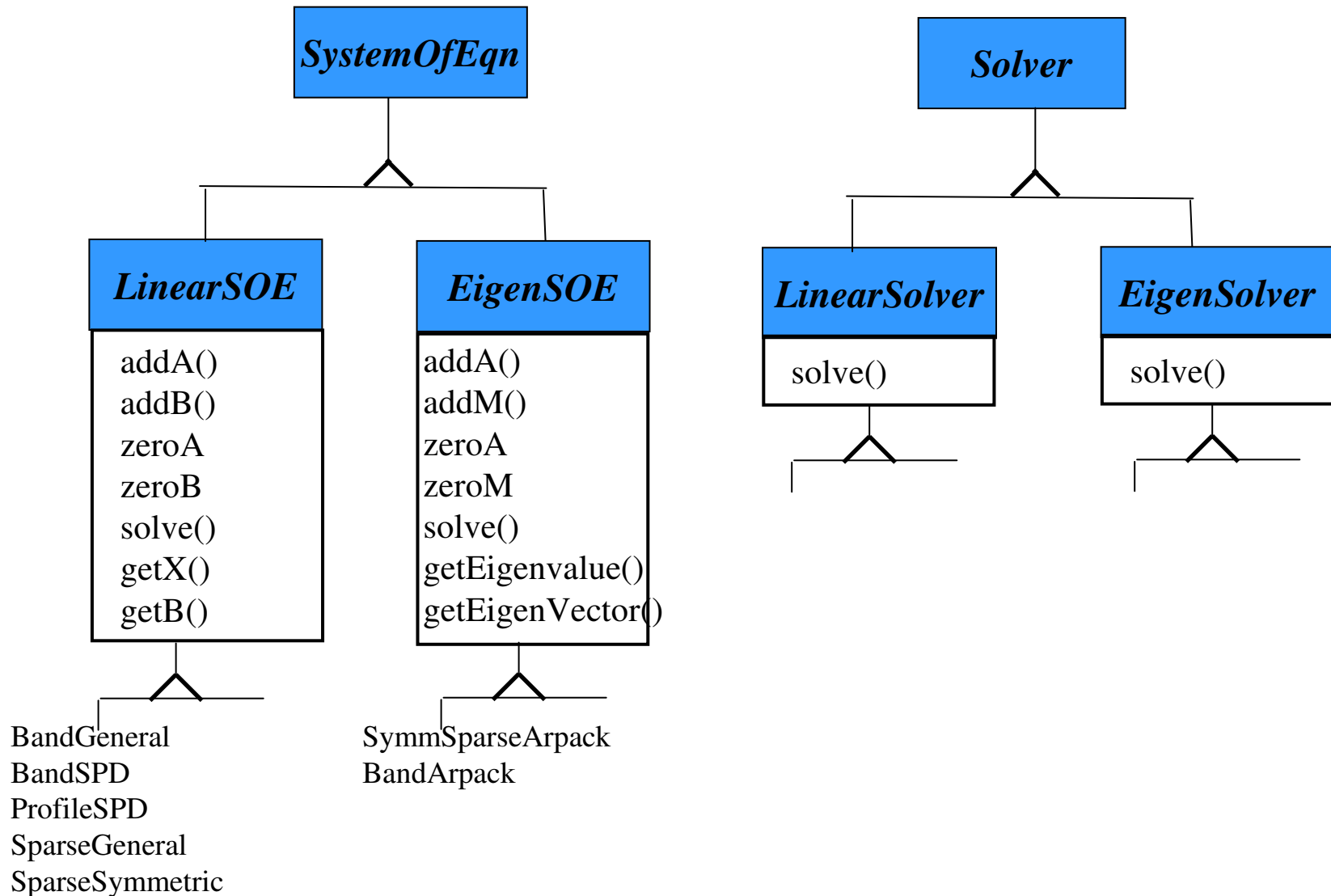
$$K^* U_r = R^*$$

$$\begin{bmatrix} K & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} U \\ \lambda \end{bmatrix} = \begin{bmatrix} R \\ Q \end{bmatrix}$$

$$[K + C^T \alpha C] U = [R + C^T \alpha Q]$$

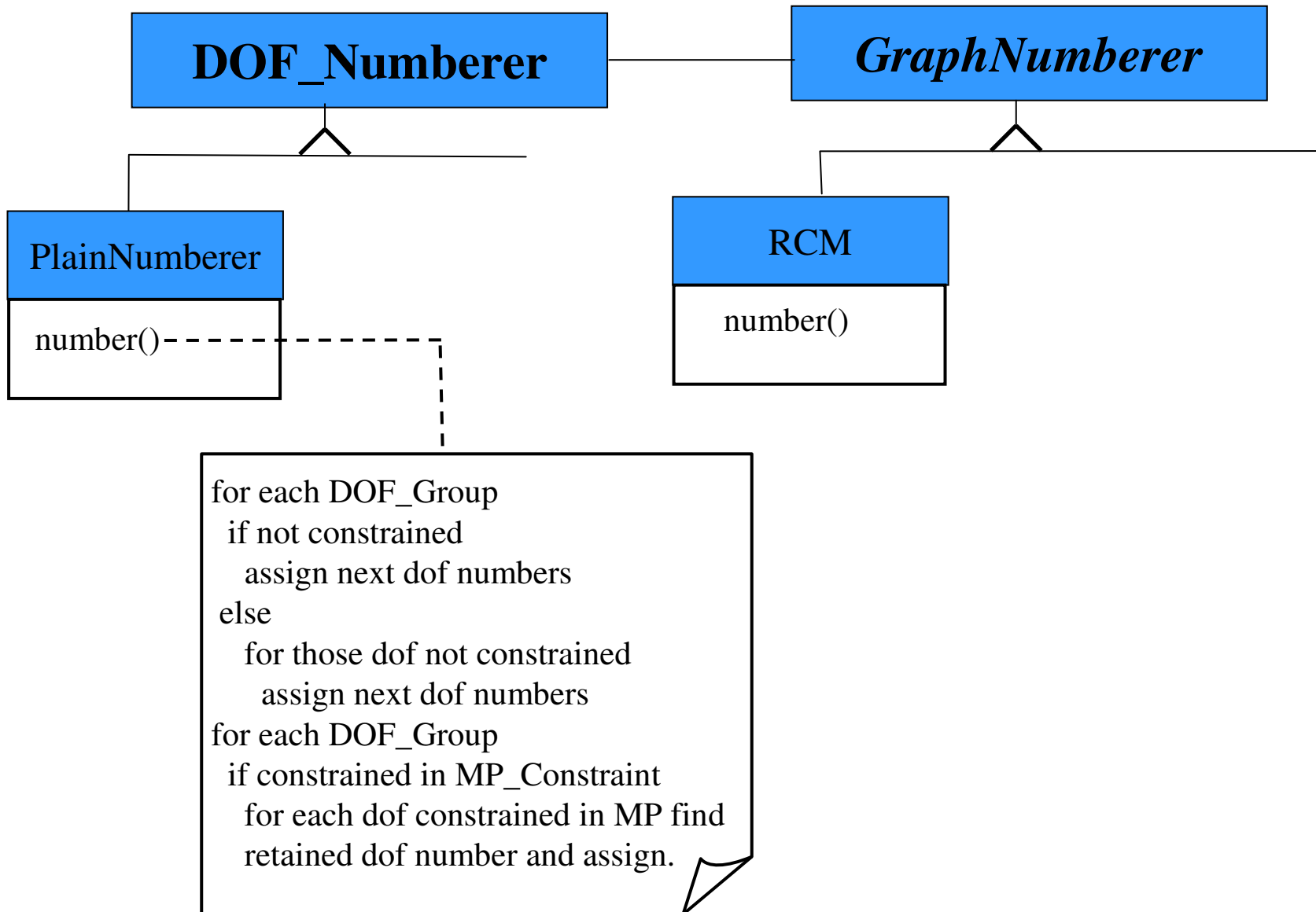
SystemOfEqn and Solver Classes:

- the SystemOfEqn classes store the matrix equations
- The Solver classes work on the SystemOfEqn classes to solve the eqn.



Numberer command:

- to specify how the degrees of freedom are numbered



ConvergenceTest Classes:

- to determine if convergence has been achieved

