PEER

Adding an Element into OpenSees

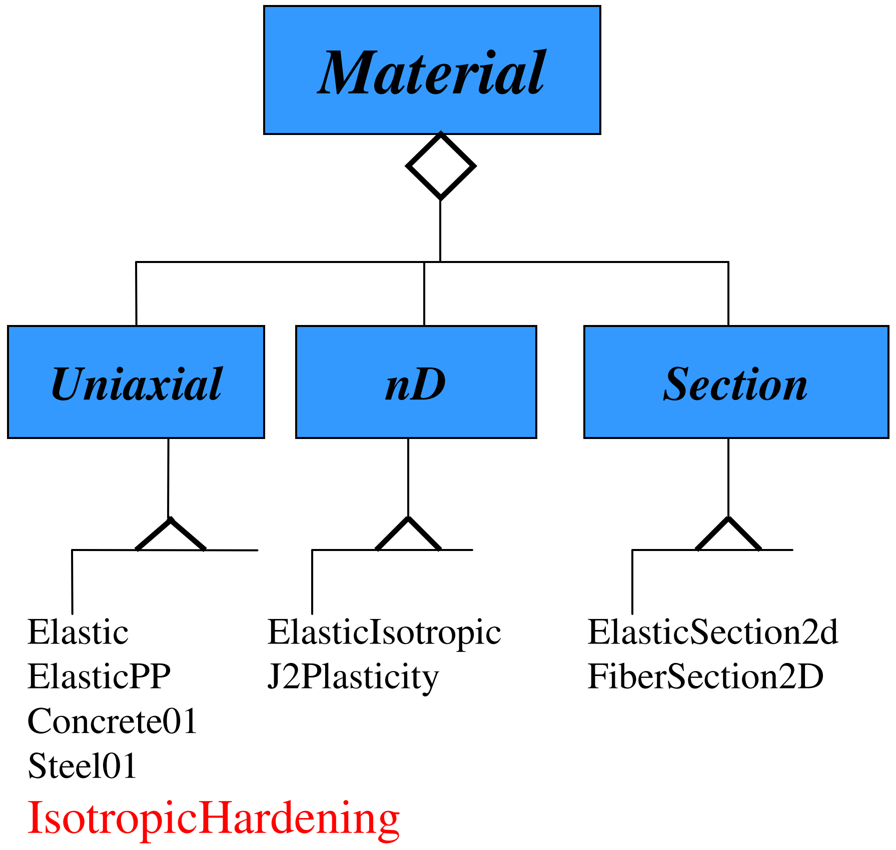
Frank McKenna

OpenSees Developers Workshop
Berkeley, CA
August 15, 2006

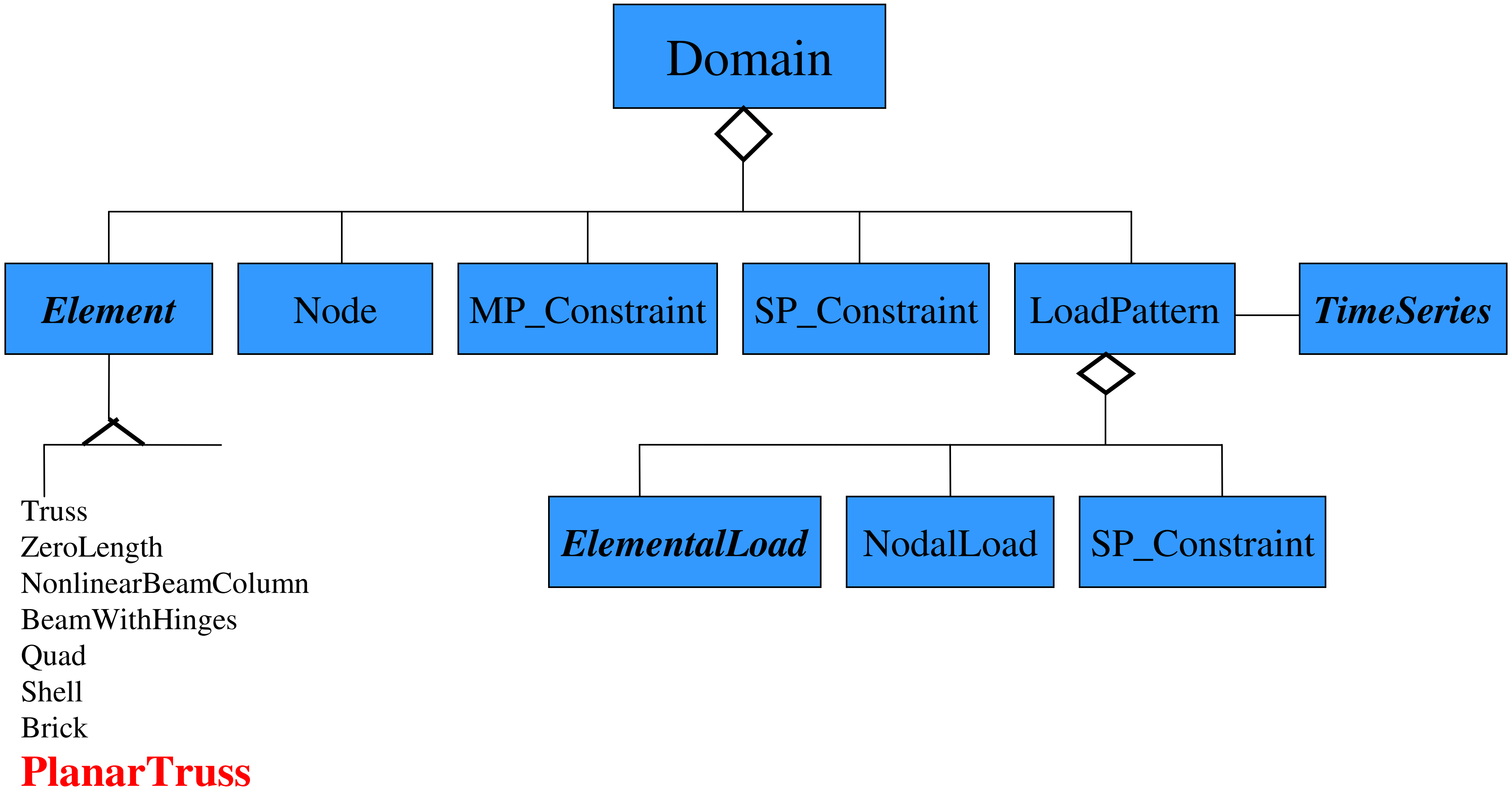
Sponsored by the National Science Foundation
through the Pacific Earthquake Engineering Research Center
and through NEESit

This Morning:

1. Added KinematicHardening Material to OpenSees Framework



This Afternoon!

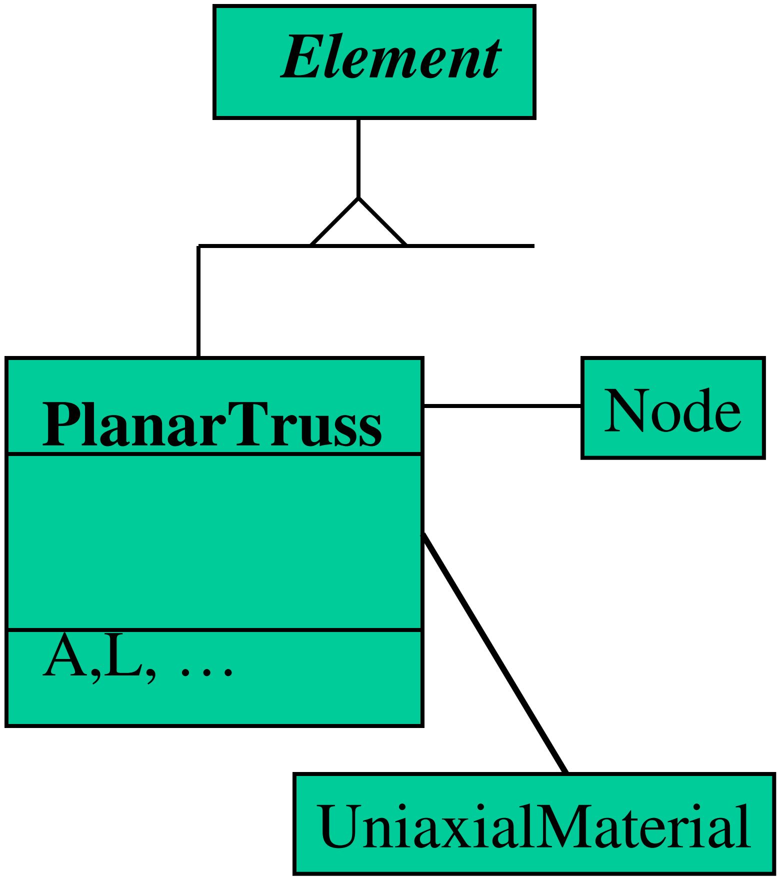


We are going to add a 2d truss to OpenSees!

Element Interface

```
class Element : public DomainComponent {
    public:
            Element(int tag, int classTag);
           virtual ~Element();
           virtual int getNumExternalNodes();
           virtual const ID &getExternalNodes()
           virtual Node **getNodePtrs;
           virtual int getNumDOF(void);
           virtual setDomain(Domain *theDomain);
           virtual int commitState(void);
           virtual int revertToStart();
           virtual int revertToLastCommit(void);
           virtual int update(void);
           virtual const Matrix &getTangentStiff(void);
           virtual const Matrix &getInitialStiff(void);
           virtual void zeroLoad(void);
           virtual int addLoad(ElementLoad *theLoad, double loadFactor);
           virtual int addInertiaLoadToUnbalance(const Vector &accel);
           virtual const Vector &getResistingForce(void);
           virtual const Vector &getResistingForceIncInertia(void);
           virtual Response *setResponse(const char *argv, int argc, Information &info);
           virtual int getResponse(int responseID, Information &info);
           void Print(OPS_Stream &ops, int flag=0);
 };
```

New PlanarTruss Element

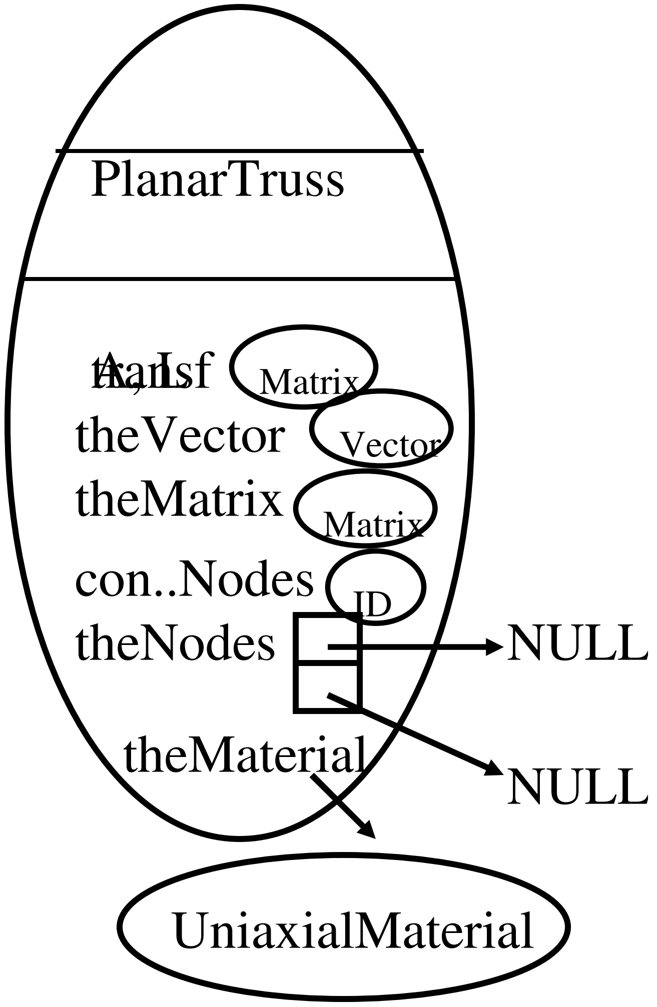


```
class PlanarTruss : public Element {
    public:
            PlanarTruss(int tag, int node1, int node2,
                   UniaxialMaterial &theMat,
                  double A);
            ~PlanarTruss();

                        …
    private:
            double A, L;
            Matrix transf;
            Node *theNodes[2];
            UniaxialMaterial *theMaterial;
            ID connectedExternalNodes;
            Vector theVector;
            Matrix theMatrix;
};
```

Constructor

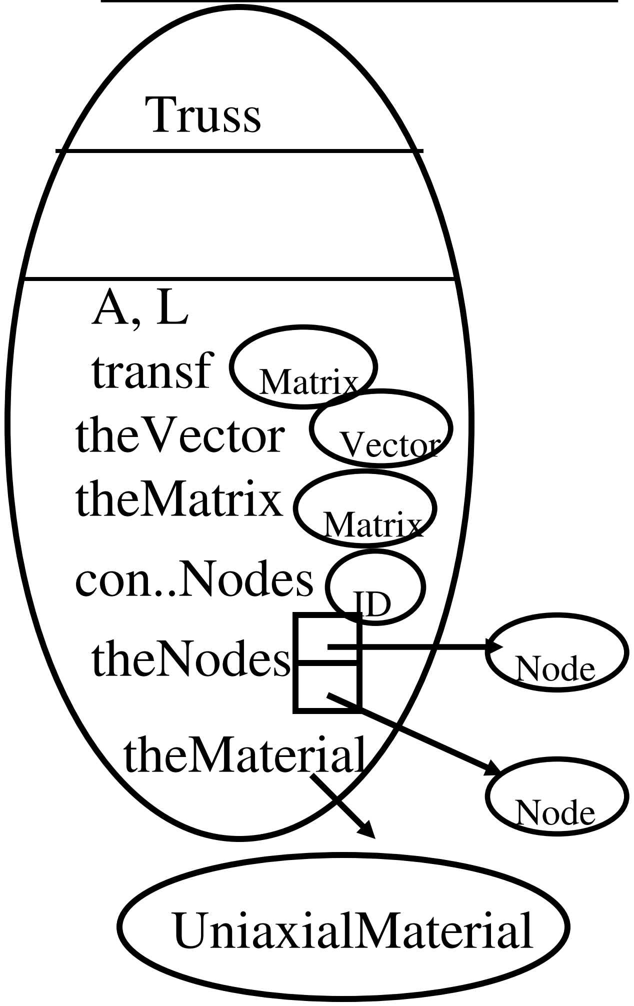
```
#define numNode 2
#define numDOF 4
PlanarTruss::PlanarTruss(int tag, int node1, int node2,
                    UniaxialMaterial &theMat,
                    double a)
:Element(tag, ELE_TAG_PlanarTruss),
 A(a), L(0), transf(1,4),
 connectedExternalNodes(numNode),
 theMatrix(numDOF,  numDOF),
theVector(numDOF)
{
            connectedExternalNodes(0) = node1;
            connectedExternalNodes(1) = node2;
            theMaterial = theMat.getCopy();
            theNodes[0] = 0;
            theNodes[1] = 0;
}
```



Destructor & setDomain

```
PlanarTruss::setDomain(Doman *theDomain)
{
  int node1 = connectedExternalNodes(0);
  int node2 = connectedExternalNodes(1);
  theNodes[0] = theDomain->getNode(node1);
  theNodes[1] = theDomain->getNode(node2);
  this->DomainComponent::setDomain(theDomain);
  const Vector &crd1 = end1Ptr->getCrds();
  const Vector &crd2 = end2Ptr->getCrds();
  double dx = crd2(0)-crd1(0);
  double dy = crd2(1)-crd1(1);
  L = sqrt(dx*dx + dy * dy);
  double cs = dx/L; double sn = dy/L;
  trans(0,0)=-cs; trans(0,1)=-sn;
  trans(0,2) = cs; trans(0,3)=sn;
  this->update();
}
```

```
Truss::~Truss()
{
    delete theMaterial;
}
```



Public Methods - some easy ones

```
int PlanarTruss::getNumNodes(void)
{
   return numNode;
}
```

```
int PlanarTruss::commitState(void)
{
    return theMaterial->commitState()
}
```

```
Node **PlanarTruss::getNodes(void)
{
    return theNodes;
}
```

Public Methods - more difficult!

```
const Matrix &PlanarTruss::getTangentStiff(void) {
    double E = theMaterial->getTangent();
    theMatrix = transf ^ transf;
    theMatrix *= A*E/L;
    return theMatrix;
}
```

```
const Matrix &PlanarTruss::getInitialStiff(void) {
    // one line needs to be changed from above
    // which one???
}
```

```
const Vector &PlanarTruss::getResistingForce(){
   double force = A*theMaterial->getStress();
   for (int I=0; I<4; I++)
       theVector(I) = transf(0,I)*force;
    return theVector;
}
```

Public Methods - most difficult!

```
int PlanarTruss::update(void)
{
   const Vector &disp1 = theNodes[0]->getTrialDisp();
   const Vector &disp2 = theNodes[1]->getTrialDisp();
   double dLength = 0.0;
   for (int i=0; i<2; i++)
       dLength -= (disp2(i)-disp1(i)) * trans (0,i);
  double strain = dLength / L;
  return theMaterial->setTrialStrain(strain);
}
```

```
void PlanarTruss::Print(OPS_Stream &out, int flag)
{
   out << “PlanarTruss  tag: “ << this->getTag() << endln;
   out << “resisting Force: “ << this->getResistingForce();
   theMaterial->Print(out, flag);
}
```

Now it’s your turn - steps to proceed:

1. In OpenSees/SRC/element there are 2 files: NewElement.h and NewElement.cpp. Make a copy of them and rename them PlanarTruss.h and PlanarTruss.cpp.
2. Open PlanarTruss.h and .cpp and make necessary changes. Hint start by doing a global replace of NewElement!
3. Open the TclElementCommands.cpp file. Search out the string “addTruss” and add the obvious addPlanarTruss additions. (hint: 2 locations)
4. Copy the file OpenSees/SRC/element/TclNewElement.cpp to be TclPlanarTruss.cpp. Make changes. (hint: look at TclTruss.cpp file in SRC/element/truss directory)
5. Add the files to the project, compile & link.
6. Does it Work? .. Test it (hint: use Example1.1.tcl)