 CMC, vol.3, no.2, pp.77-95, 2006

Object-Oriented Modeling of Solid Material in Nonlinear Applications

Hamid Sharifi[1] **and Augustin Gakwaya**[1]

Abstract: In this paper, an object-oriented modeling of solid material constitutive behavior using the UML notation is presented. Material properties are first classified into large and small deformation kinematical models. In the small deformation package, we keep classes such as Elastic, ElastoPlastic, ViscoElastic and ViscoPlastic. In the large deformation package, we store classes such as ElastoPlastic, HyperElastic, HyperPlastic, HyperViscoElastic, HyperViscoPlastic and so on. The hierarchical structure, the association relationships as well as key attributes and methods of these classes are presented. We used a C++ implementation of the above model for developing HyperElastic, HyperElastoPlastic and Contact applications in the Diffpack environment.

keyword: Material modeling, Hyperelastoplasticity, Finite Element, Object-Oriented, UML, Diffpack.

1 Introduction

The object-oriented programming appears to be the method of choice for recent finite element code developments [Dubois-Pélerin(1993), Bettig(1996), Zabaras(1999), Cross(1997), Masters(1997), Toukourou(2001), Nikishkov(2006)]. There is a tendency to separate the material behavior from finite element objects in recent finite element and boundary element codes [Foerch(1997), Besson(1997), Sharifi(2005)]. In this perspective to access to the material properties, a geometrical finite element is associated to a class of material. For example, in the calculation of the consistent tangent matrix or the elasticity tensor, needed in the finite element calculation, at each Gauss point, material properties can be accessed by the element via an association link between the element and the material class. Here, elements do not inherit material characteristic (as proposed in [Kong(1995)]) and therefore, there is no need to have numerous combined geometrical-material element classes, such as: T3 triangular elastic element, T3 triangular plastic element, T3 triangular viscoplastic element and so on. These elements can be replaced with a T3 triangular element that has access to different material classes. This separation is more useful when one works with a large number of nonlinear material constitutive laws [Foerch(1997)]. As a result, the object-oriented model of material can be considered separately, and it does not depend on other finite element or boundary element objects.

Beside, the object-oriented modeling of material characteristics gives us the opportunity to model and to classify the material properties in a convenient way. It is a useful tool to present the kinematical as well as the hierarchical relationships between different types of materials. An object-oriented model is not only valuable to be used for software engineering purposes, but also it is a useful tool in educational and research intentions. A good object-oriented model presents each material using its specific characteristic properties (attributes of the material object) and its behaviors in actual application (methods of the material object). It helps us to show similarities and relations among materials, and it permits to collect the common material properties in a more general class (a super class). This kind of modeling gives us a global overview of materials. It helps us to present when a specific type of material behaves differently from its general category (polymorphism).

A successful solid material model must cover linear and nonlinear, as well as isotropic and anisotropic material properties. In the technical literature, solid material properties are classified into different categories [ABAQUS(2005)], from which the following can be considered as classical categories of solid properties:

- General properties such as material density and initial temperature;
- Mechanical behavior, Stress/Strain relationship (Elastic, Inelastic properties,);
- Thermal properties;
- Damage properties;

[1] Department of Mechanical eng. Laval University, Quebec, Canada

- Dynamic properties such as acoustic; and
- Electrical properties.

Some of these properties, such as the conductivity and the elasticity, are totally unrestricted and they can be used alone or together. Some other properties, for example the elasticity and the plasticity, are not exclusive (a plastic solid may also have the elastic behavior in a small strain range). On the other hand, some properties cannot be combined; for example if we chose the small deformation model we cannot use the large deformation properties. Northwest Numerics and Modeling Inc. [Northwest(2005)] defines object BEHAVIOR as the collection of:

- primal (prescribed) variable (*i.e.* strain) and dual (associated) variable (*i.e* stress),
- set of state dependent variables,
- set of auxiliary variables (used in post-processing),
- external parameters (defined by the user such as temperature and external load), and
- material parameters.

In the object-oriented modeling, we can present above properties and variables. We can specify for each class of material different attributes (properties) and different operations over these attributes (behaviors). More important, we can construct a global architecture to relate these classes together. As one realizes from the previous discussion, the classical categories of material properties by themselves are not enough to lead us to an appropriate object-oriented model. We need a natural or an inherent way to classify the material properties.

The constitutive equations of a class of materials are usually defined by a set of first order ordinary differential equations (ODE) expressed in terms of state variables, material parameters and boundary conditions. As a result, in each class of materials there must be methods to integrate these ODEs and to find the material property matrix (the consistent tangent matrix) or the elasticity tensor. If it is not easy to perform the direct evaluation of an integral, for example in a plasticity problem, explicit or implicit methods of integration can be used. For example, we can consider the Runge-Kutta method (explicit) or the Newton θ method (implicit).

The UML (Unified Modeling Language) notation [Muller(1997), Graham(2001), Wampler(2002)] is increasingly used in the object-oriented modeling. We have chosen this notation to present our solid material model. The UML, which is actually a fusion of Booch, OMT and OOSE methods, is accepted as the standard notation for the object-oriented application design. Rational Rose software is a powerful tool that can be used for object-oriented modeling with UML [Rational(2006)]. Unfortunately, Rational rose is a commercial software, therefore we have used open source ArgoUML [ArgoUML(2006)], which is good as well, to present our UML model.

Input/output objects as well as several numerical method utility objects, i.e., Matrix, Vector, linear and nonlinear solvers, are needed to write an object-oriented finite element (FE) application. To develop our implementation examples, we have taken advantage of the Diffpack package. The Diffpack package is efficient object-oriented software written in C++. It consists of a set of useful libraries that makes input/output and utilities objects available [Langtangen(1999), Diffpack(2006)].

2 Deformation / time curve of solid material

In a search for characteristics to be used in classification of material properties, which is also suitable for our object-oriented modeling, we consider the behavior of a solid material under loading. Generally, the deformation versus time curve of a solid material under a gradually increasing load, which starts from zero and goes higher than the yield limit, gives us a deformation history, which is similar to one presented in Figure 1. In this experience, we load the material so that the yield limit of the material is over passed, but it stays below the fracture limit. Thereafter, suddenly, we remove the load (point $\boldsymbol{B}$).

As we increase the load, the material is deformed from the state $\boldsymbol{A}$ to the state $\boldsymbol{B}$. If in the point $\boldsymbol{B}$ we remove the load, part of material deformation recovers instantly, from $\boldsymbol{B}$ to $\boldsymbol{C}$, this recoverable deformation is called the *elastic* part. As the time goes on, another part of deformation is recovered, from $\boldsymbol{C}$ to $\boldsymbol{D}$. This time dependent nonlinear elastic recoverable part is called the *anelastic* part. After the point $\boldsymbol{D}$, we observe that there is a deformation part that cannot be recovered even with time, from $\boldsymbol{D}$ to $\boldsymbol{E}$, this unrecoverable part could be due to *the damage, the plastic, the viscous or the viscoplastic* property of the material.

Depending on the type of the solid material and the nature of our application, some of these deformation behaviours are more important than the others. For instance, in the small deformation, it is not necessary to consider the unrecoverable deformation part, while for some large deformation applications; the effect of elastic part can be neglected.

Considering the above observations and focusing our attention to the application nature, we can categorize our solid behavior properties in two grand kinematics models, which are the **small** and the **large** deformation models. In the small deformation, we can model the material behavior by introducing the linear elastic and the weak nonlinear behavior parts. On the other hand in the large deformation, we model the material behavior and characteristics by introducing large elastic and large inelastic or permanent deformation parts.

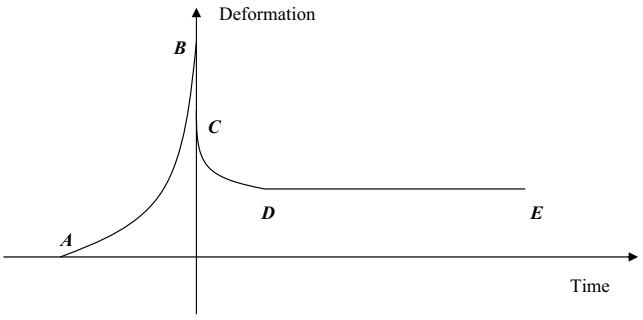

Figure 1 : Deformation/Time curve of a typical solid material

Since the behavior of materials in each part of the above curve is very important in our application, we can further refine our classification of material properties in the following rheological categories

1. Rigid;
2. Elastic;
3. Anelastic;
4. Damage;
5. Plastic and
6. Viscous.

We have added the rigid category, in order to have more general categories and to allow covering the rigid boy motions of a material.

Using the above rheological based classification and also taking into account the thermodynamic formulations of material constitutive equations, from now on, we will present our object-oriented model using the UML notation.

3 Organization of packages

Figure 2, presents the organization of packages of our object-oriented model of material. As one can see, we put all of the material classes into a package, called *Material*. The *utility* package contains some of the numerical analysis utility classes, for example, classes which make tensor, matrix and vector operations needed in our calculations available. Inside the Material package, we have two sub-packages; they are the *Fluid* and the *Solid* packages. The Fluid package has also been added to our modeling in order to construct a more general model and to store the fluid properties for the future development of the model.

The Solid package has six internal packages. These are: *Small_deformation*, *Large_deformation*, *Isomorphism, ThermoElectric, ThermodynamiquePotential* and *Loading_Surface*. Classes that are in relation with the large deformation behavior of the material are kept in the Large_deformation package. In Small_deformation, we keep classes, which are in relation with the small deformation behavior of a material. The link between Large_deformation and Small_deformation emphasize the possibility of the access to the small deformation classes from the large deformation classes. The Loading_Surface package has two interior packages. They are *YieldCriterion* and *DamageCriterion* packages. For each yield or damage criterion model, we can have a class inside the YieldCriterion or DamageCriterion packages, for example, VonMises, Drucker and Hill are three classes inside YieldCriterion package. We thermodynamically consider the consistent material behavior, which is based on the concept of observable and internal state variables; therefore we have to deal with the deformation potential in small and large deformation calculations. As a result, the *ThermodynamiquePotential* package has also been inserted into the solid package to store different large deformation potential modes such as: Plastic and Damage dissipation potentials, and different Elastic energy

potentials such as Ogden, Hill, Polynomial, and so on. The objects inside this package have access to the Loading_Surface package, used in calculation of the damage rule and the plastic flow rule.

Isomorphism is another package inside the Solid package. This package is used to keep different material isomorphism or symmetry models. For example, isotropic, orthotropic and transversal isotropic are three different isomorphism models that we have considered in our project. Finally, to store different classes in relation with thermal and electrical properties of the material, the *ThermoElectric* package has also been added inside the Solid package.

After being familiarized with the package organization, we now pay attention to the main classes and to their relationships within our object-oriented model.

4 General overview of the main classes of material properties

The main classes of the material properties are shown in Figure 3. Material is our root super-class. We have divided material properties into two general categories, solid and fluid. Each category is presented by a derived class of Material; they are the Solid and the Fluid classes. Note that, in order to have a more general model we have added the fluid properties class to our model, but we only consider the solid properties and leave the fluid properties for the future development of our model.

As we consider the solid behavior from two kinematical viewpoints (small and large deformation models), we keep the solid derived classes in two packages which are the Small_deformation and Large_deformation packages. As we mentioned in the previous sections, a solid could have different types of response. i.e., Rigid, Elastic, Anelastic, ViscoElastic, ElastoPlastic, ViscoPlastic and so on. Rigid, Elastic, ViscoElastic or Anelastic, ElastoPlastic and ViscoPlastic are classes created in Small_deformation package that are used to model solid behaviors in the small deformation range. These classes are the sons of the Solid class; they have access to general solid properties from their super-classes (Solid and Material).

The isomorphism properties of a solid or the thermal properties of a material are available from Solid and Material classes. The ViscoPlastic class is a child of ElastoPlastic and ViscoElastic classes (multiple inheritances) to give it the permission to use the information stored in these two classes.

Large deformation simulation classes are kept in the Large_deformation Package. As we are interested in hyper elastic models in the large deformation range, we have created the HyperElastic class as the local root in this package. This class is a child of the Elastic class; as a result, it has access to elastic properties of the material, as well as other general properties of the material via its super-classes.

To deal with different material behavior in the large deformation model, we have created HyperViscoElastic, HyperPlastic and HyperViscoPlastic classes in the Large_deformation package. They are all children of the HyperElastic class. Each of these classes is also a child of its equivalent class in the small deformation package to give them access to the small deformation behavior and properties.

In the next section, we look in details at some of these classes and the relations among them and other associated classes.

5 Material and Solid associated classes

General thermal properties of material, such as the initial temperature and the specific heat capacity, are stored in the ThermalProperty class. ElectricalProperty is a class that keeps general electrical properties of material. These two classes are stored in the ThermoElectric package which is inside the Solid package. The associated classes to the Material and Solid classes are presented in Figure 4. As one can see, the Material class has the links to the ThermalProperty class and to the ElectricalProperty class. Using these links, a material object has access to the information inside above classes. Note that an object of type Material has also access to the information accessible from these classes.

Damage and Isomorphism are classes that must be accessible to the Solid class. Using the associations between the Solid class and these classes, a solid material can use these properties. The Damage class is used to store damage properties of a solid such as damage limit, decay rate and so on. The Isomorphism class is the superclass of the isomorphism solid types, which we discuss in the next section.

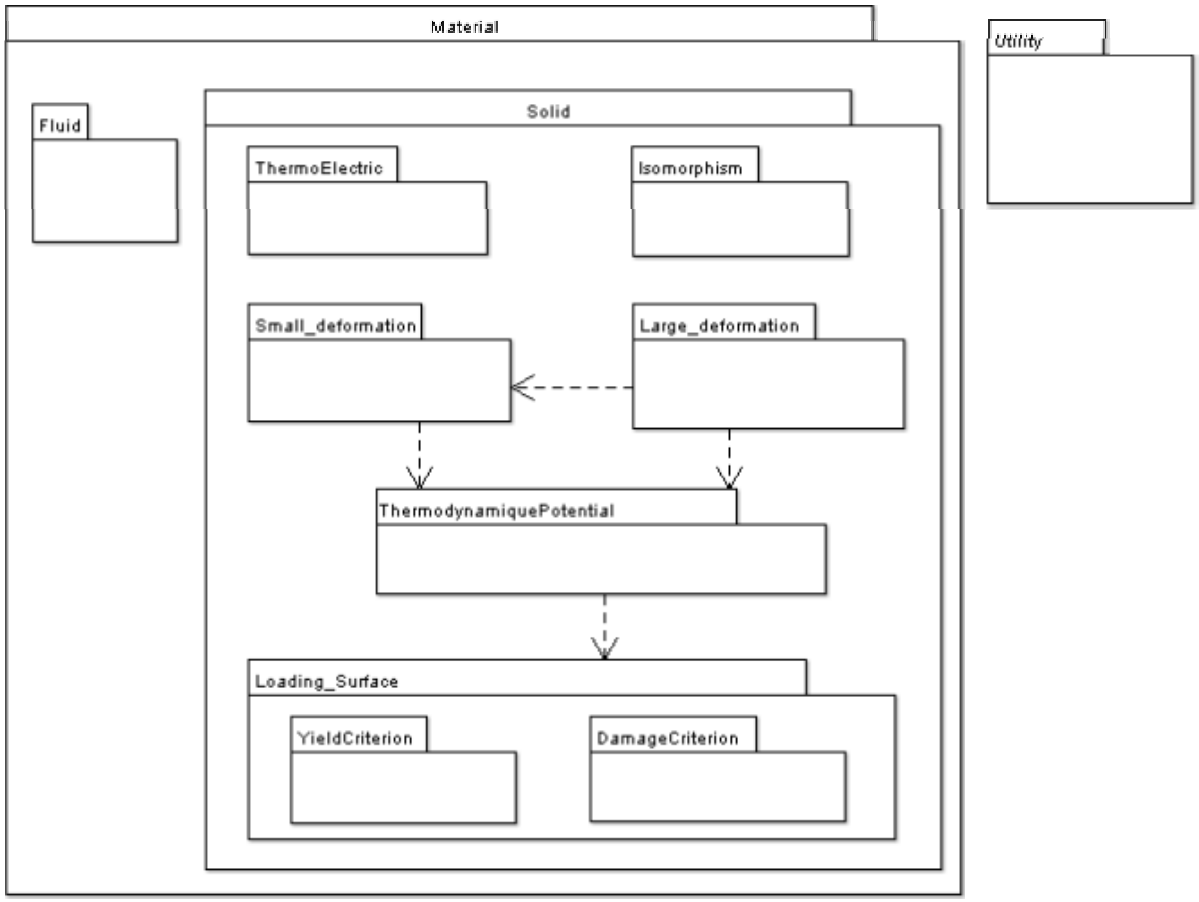

Figure 2 : Main packages organization

6 Isomorphism types

The object-oriented model of the isomorphism types of a solid is presented in Figure 5. We have considered three general isomorphism mechanical properties types of a solid. For each type, we created a derived class (child) of the Isomorphism superclass. For isotropic solids, having spherically symmetric properties, we keep the Young modulus E, the Poisson ratio ν and the shear modulus G (other isotropic properties can also be added), these attributes are defined in the IsotropicSolid class. For a transversally isotropic solid, having a plane of isotropy, we have created the TransversalIsotropicSolid class, and we have defined the Young modulus E_1 & E_2, the Poisson ratios ν_{12} & $\nu_{31}$and the shear modulus G_{12} & G_{31}in two directions. For an orthotropic solid, having different properties in each of the three principal directions, the OrthotropicSolid class has been created, and we defined the Young modulus E_1, E_2 & E_3 the Poisson ratios ν_{12}, ν_{23} & $\nu_{31}$and the shear modulus G_{12}, G_{23} & $G_{31}$of the planes of symmetry (some other key information can also be added).

In case of the thermal properties of Solid, we can also observe that different isomorphism types. We do not mix the thermal and the mechanical properties together, therefore we have created three isomorphism derived classes for the ThermalProperty superclass to keep the thermal isomorphism data. These classes are ThermalIsotropic, ThermalTransIsotropic and ThermalOrthotropic classes. For example, for the thermal conductivity k, we have defined one k attribute in the ThermalIsotropic class, while in the ThermalTransIsotropic class we defined two attributes, k_1 & k_2, and in the ThermalOrthotropic class we defined three attributes k_1, k_2 & k_3, one for each principal direction of the material.

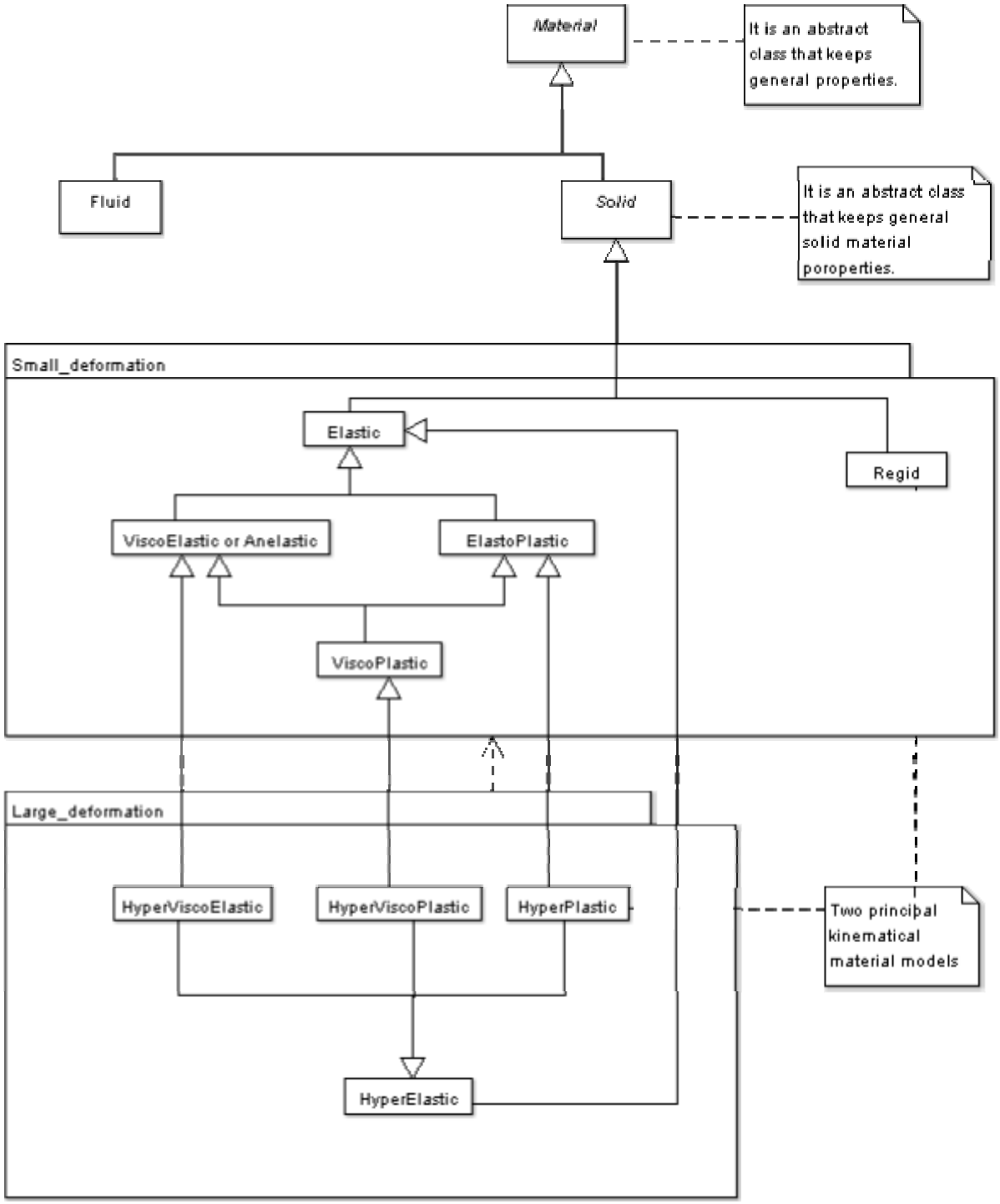

Figure 3 : Classes general overview.

7 Small deformation classes

As an example, some detail information of certain small deformation classes are shown in Figure 6. As one can realize, since the Solid class is an abstract class, it has pure virtual methods such as "*reqMatrixD()*", it can not be used to define a solid material object by itself. The method "*reqMatrixD()*" computes the material property matrix, because this method is defined as a pure virtual

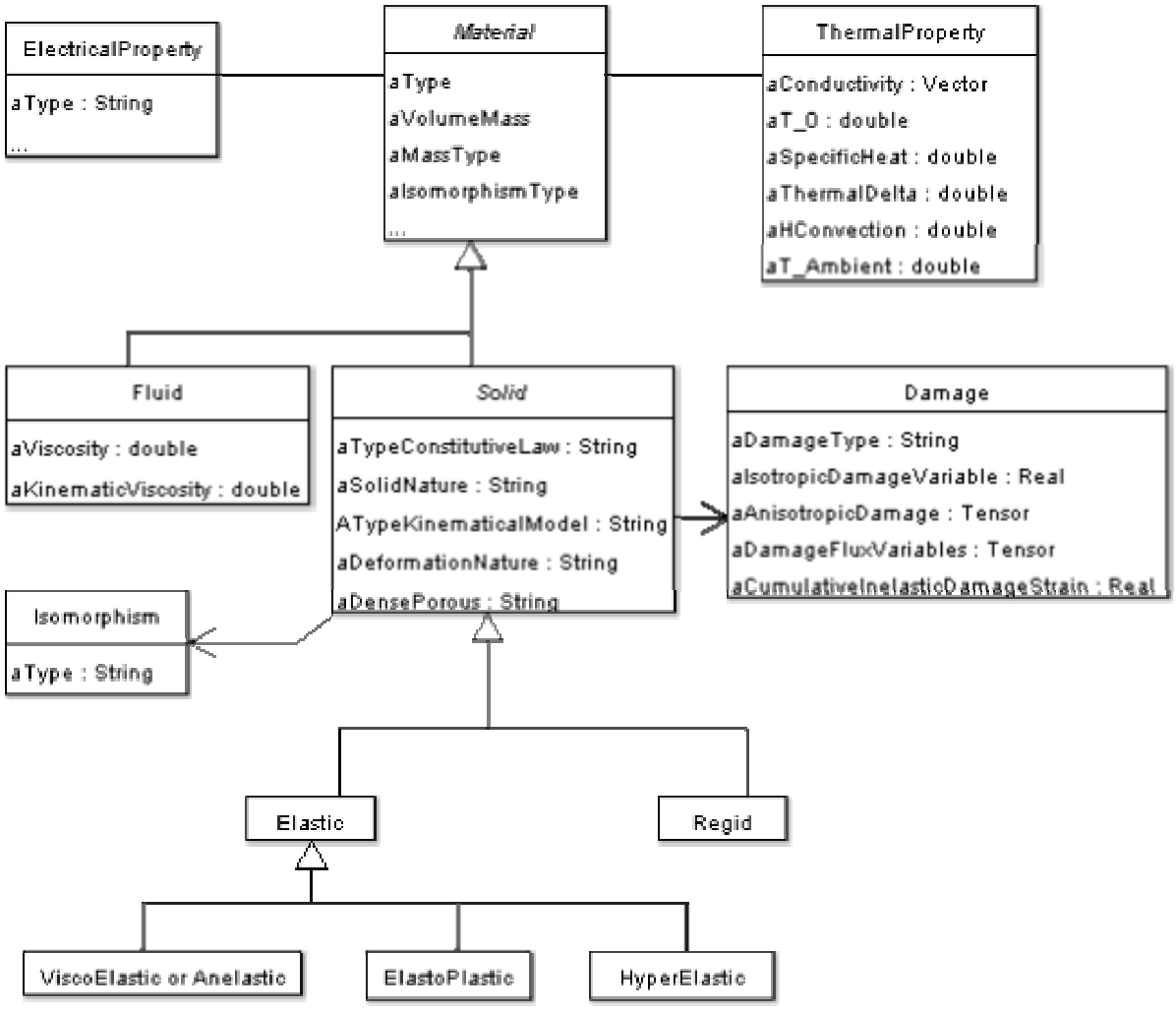

Figure 4 : Material and Solid associated classes

method in the Solid class, it has to be redefined in each of the derived classes. Elastic, ViscoElastic and ElastoPlastic are three children of the Solid class, and for each of these classes, the above method must be redefined with respect to the material type. Information in the Solid class and its associated classes are almost sufficient for the definition of the Elastic class, and the computation of the material tangent matrix or the elasticity tensor, but these attributes and methods are not sufficient in case of the ViscoElastic and the ElastoPlastic classes. Therefore, the missing material properties have been added to the appropriate classes. For example, the viscosity and the relaxation parameters have been added as the private attributes of the ViscoElastic class. The shear modulus, the bulk modulus, the yield limit and the type of yield criterion are also defined and placed in the ElastoPlastic class.

ViscoPlastic is another class in the small deformation kinematical group. As it needs both the viscoelastic and the elastoplastic properties, we have defined it as a child of ViscoElastic and ElastoPlastic. Abstract methods such as *reqMatrixD()* and *reqTensorElasticity()* are redefined in this class to present the behavior of this type of material. The ElastoPlastic class needs to have access to the yield criterion models. In Some application this class needs the information about the hardening characteristics, the kinematical calculations and the deformation history of the material. As a result, we have created the HardeningMechanism class to keep the hardening mode of the material, and the Kinematics class to perform the kinematical operations. The YieldCri-

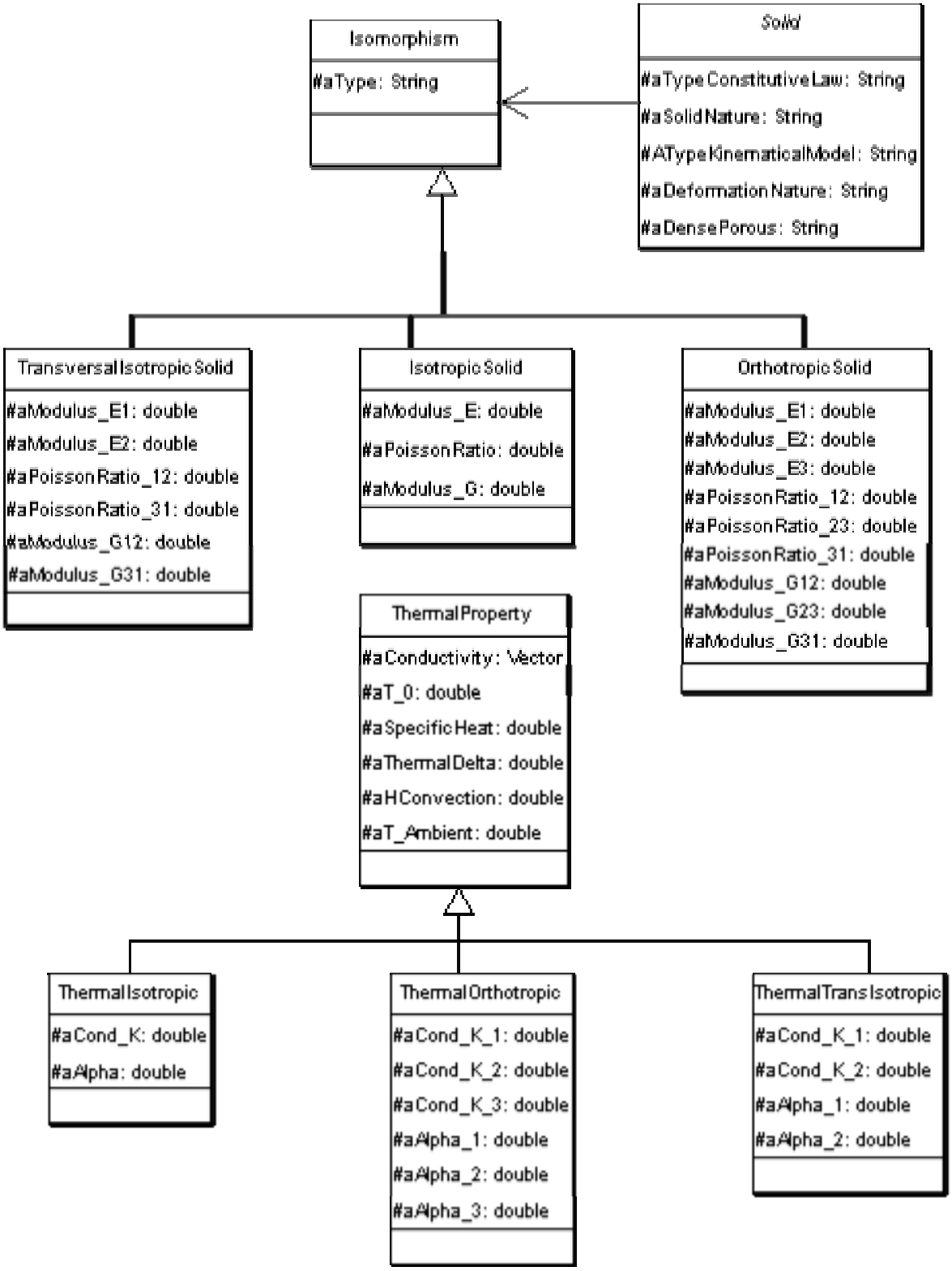

Figure 5 : Detailed diagram of isomorphism relations.

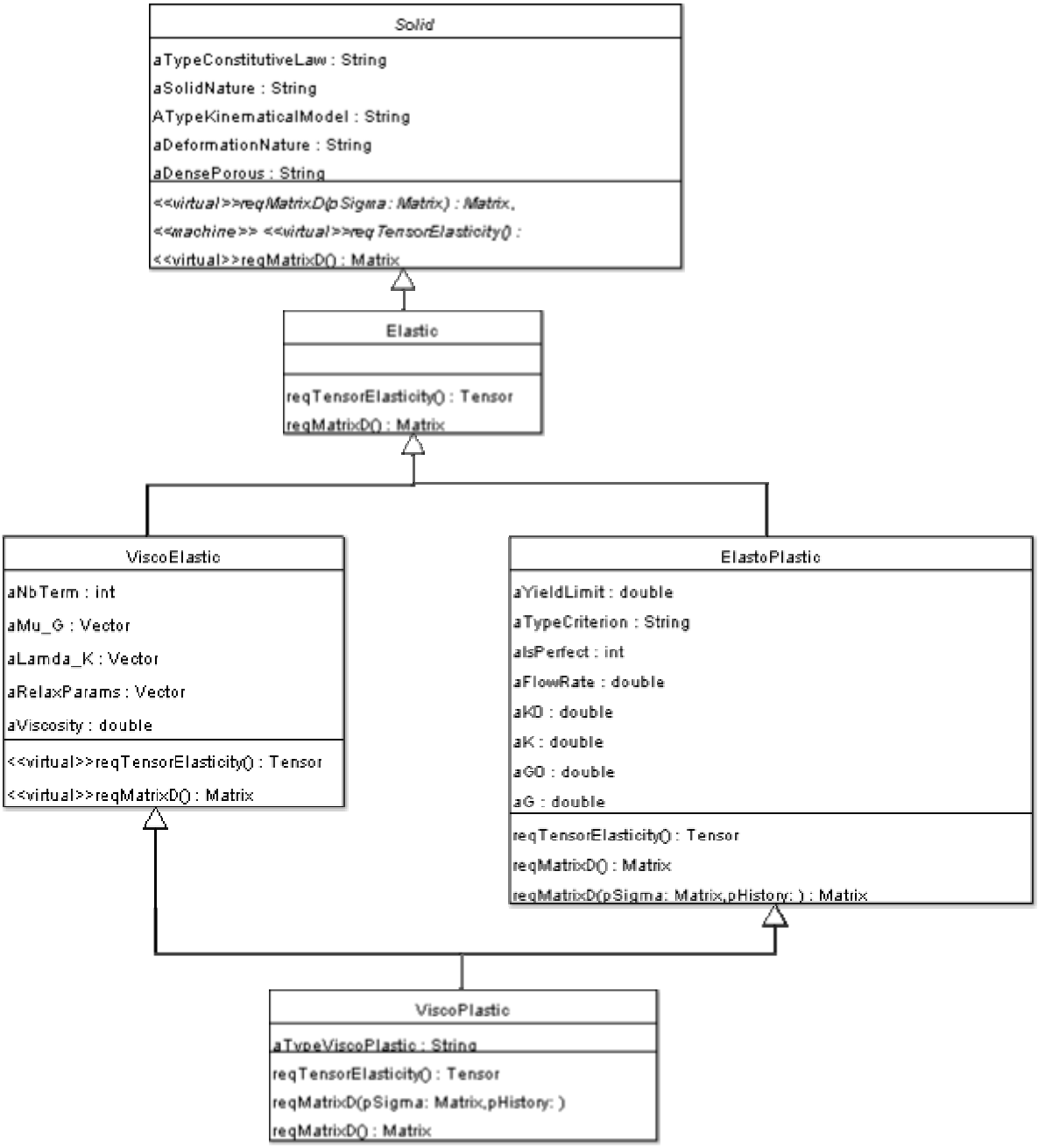

Figure 6 : A detailed diagram of the small deformation class hierarchy

teria classes are used to store information about plastic loading (or yield) function and its parameters. Several yield criterion classes have also been created for the classical yield criterion models, for example VonMises, Hill, Tresca, Drucker and PowderCap are defined in our project. These classes are specializations (derived classes) of the YieldCriterion class, they are inside the YieldCriterion package inside the Loading_Surface package. In Figure 7, we illustrate the relation between the ElastoPlastic, Hardening, MaterialHistory and Yield Criterion classes.

Association among the ElastoPlastic class and the Yield criterion classes are exclusive. This means that we can have only one of the yield criterion classes in relation with the ElastoPlastic class at a time. For example, we cannot have Capmodel and VonMises objects at the same

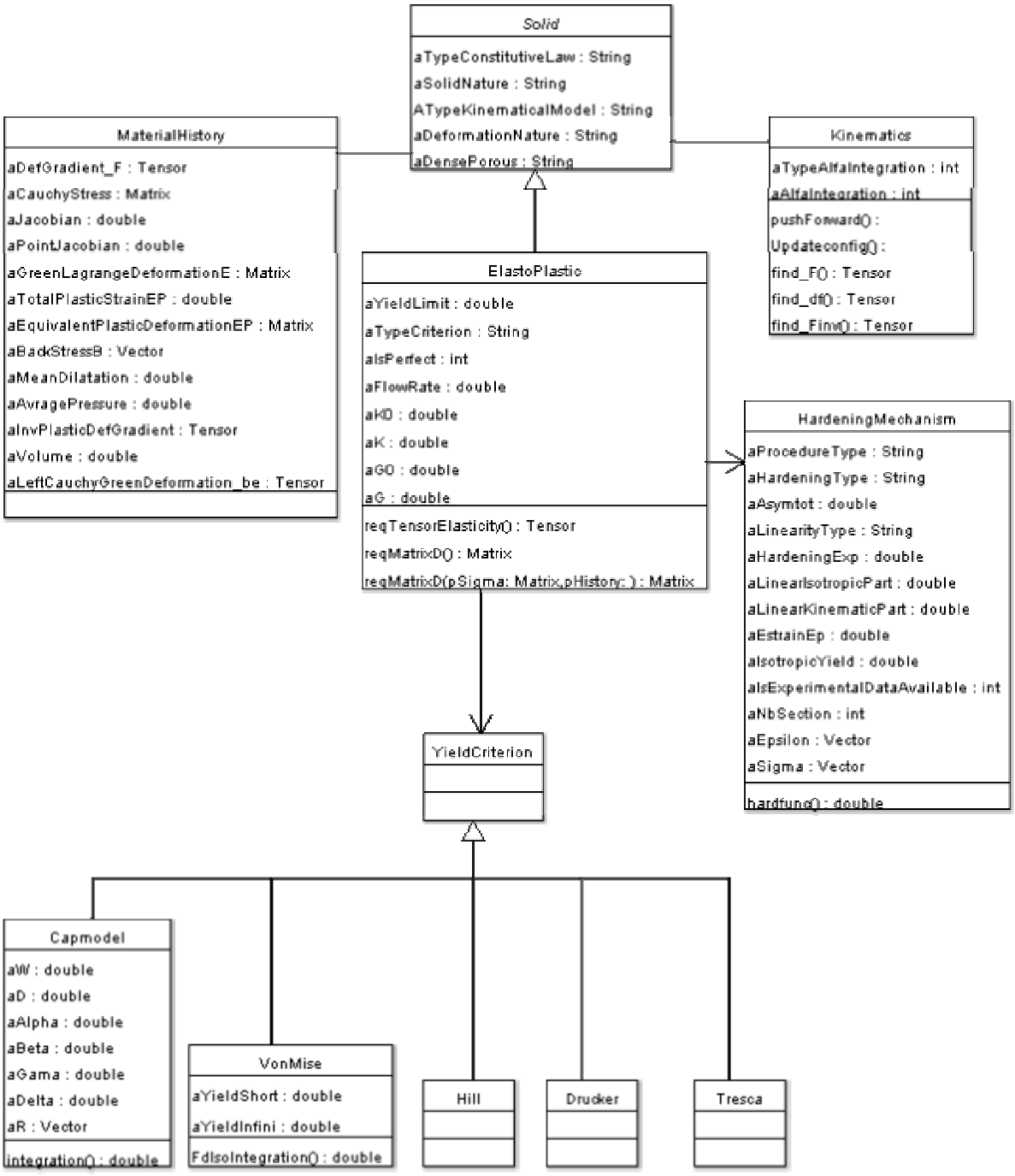

Figure 7 : Diagram of hardening relations

time in association with an ElastoPlastic object.

MaterialHistory is a class that is created to keep the deformation history of a material at each Gauss point or integration point. In this class, we keep several state variable properties in two instances of time: in the current and in the previous time steps. Therefore, each attribute must be associated with two values in this class (i.e., by using an array of two elements or by repeating the attributes with "N" postfix). As it is possible that we do not need to use all of MaterialHistory variables in a specific application, then to use the less memory space in case of the vector, the matrix or the tensor attributes, we

only keep pointers to these attributes. Most of the attributes of this class are used by the HyperElastic class of the LargeDeformation package.

8 Large deformation classes

Figure 8 shows some of the large deformation classes in relation with the small deformation classes. As it can be noticed, to each of the material types (classes) in the large deformation kinematics package, there is a corresponding father (superclass) in the small deformation kinematics package. The local root class of the large deformation classes is the HyperElastic class, as we have chosen the hyper-elastic model as a base of our modeling of the large deformation phenomena. The HyperElastic class itself is a derived class of the Elastic class; therefore it has access to all of material properties accessible to its superclasses. To have access to the history evolution of a material and to the kinematical operations, the HyperElastic class via the Solid class has a link to the MaterialHistory and the Kinematics classes, the same as the ElastoPlastic class in the previous section. For example, the left Cauchy Green deformation tensor (b^e) is needed in the hyperplastic calculation. This tensor at the current and at the previous time (N) is stored in the *aLeftCauchyGreenDeformation_be* attributes of the MaterialHistory class.

The Polymorphism methods such as *reqMatrixD()* and *reqTensorElasticity()* are also redefined in the Hyperelastic classes to return the appropriate material tangent matrix and the elasticity tensor to the calling program. As the hyperelastic formulation is driven from the hyperelastic potential, and since we thermodynamically consider the consistent material behavior, the elastic energy potentials and the dissipation potentials are essential in our calculation. We have created several elastic energy potential classes such as Hill, Ogden, Polynomial and Neohookean. Figure 9 shows the association relationships among the HyperElastic class and the hyperelastic potential classes. The FreeEnergy class has the multiples heritage, because the total energy is the sum of the elastic, the plastic and the damage energies. These classes are inside the ThermodynamiquePotential package that is inside the Solid package.

Figure 10 shows the relations between the plastic and the damage dissipation potential classes, the damage and the plastic criterion classes, and the Damage and the HardeningMechanism classes. The PlasticDamagePotential class is the son of the PlasticPotential and the DamagePotential classes. In this class the coupled plastic and damage effects are defined. The link between the Damage class and the DamagePotential class, and the link between the HardeningMechanism class and the PlasticPotential class show that objects of type Damage or HardeningMechanism have access to objects of type DamagePotential or PlasticPotential respectively. These associations can be used, for example, in the calculation of the damage rule or the plastic flow rule. The DamagePotential and the PlasticPotential classes have access to the YieldCriterion and DamageCriterion respectively as the loading surface functions can be used in formulation of dissipation potentials.

9 Thermoelectric classes

In some applications, we need access to the thermal, the electrical or the thermoelectrical behavior of a solid. To deal with such applications, we have created two template classes: ElectroSolid and ThermoSolid (presented in Figure 11). The type of the template parameter of these classes is a solid class; as a result we can create several thermo-solid or electro-solid classes. For example, the ThermoPlastic class is created from the ThermoSolid class by choosing the ElastoPlastic class as the template parameter.

The ThermoElectroSolid class is a class that combines thermal and electrical effects. As one can see from Figure 11, it has the aggregation associations with ThermoSolid and ElectroSolid templates. Therefore, it is also a template class that is formed from the combination of these two templates. Moreover, we can also add some other appropriate attributes or methods to this class.

All of the above classes can be put in the ThermoElectro package inside the Solid package. Piezoelectric class is another interesting class inside this package. In this class, we have defined special piezoelectric characteristics. It is also in association with the ElectroSolid class to give it access to the general electro-solid methods and attributes.

10 Integration of constitutive equations:

As mentioned in previous sections, the "*reqMatrixD()*" method is the polymorphism method which computes the material property matrix (tangent moduli). The "*reqTensorElasticity()*" method is another polymorphism method that calculates the elasticity tensor. These methods must be redefined in each non-abstract material type

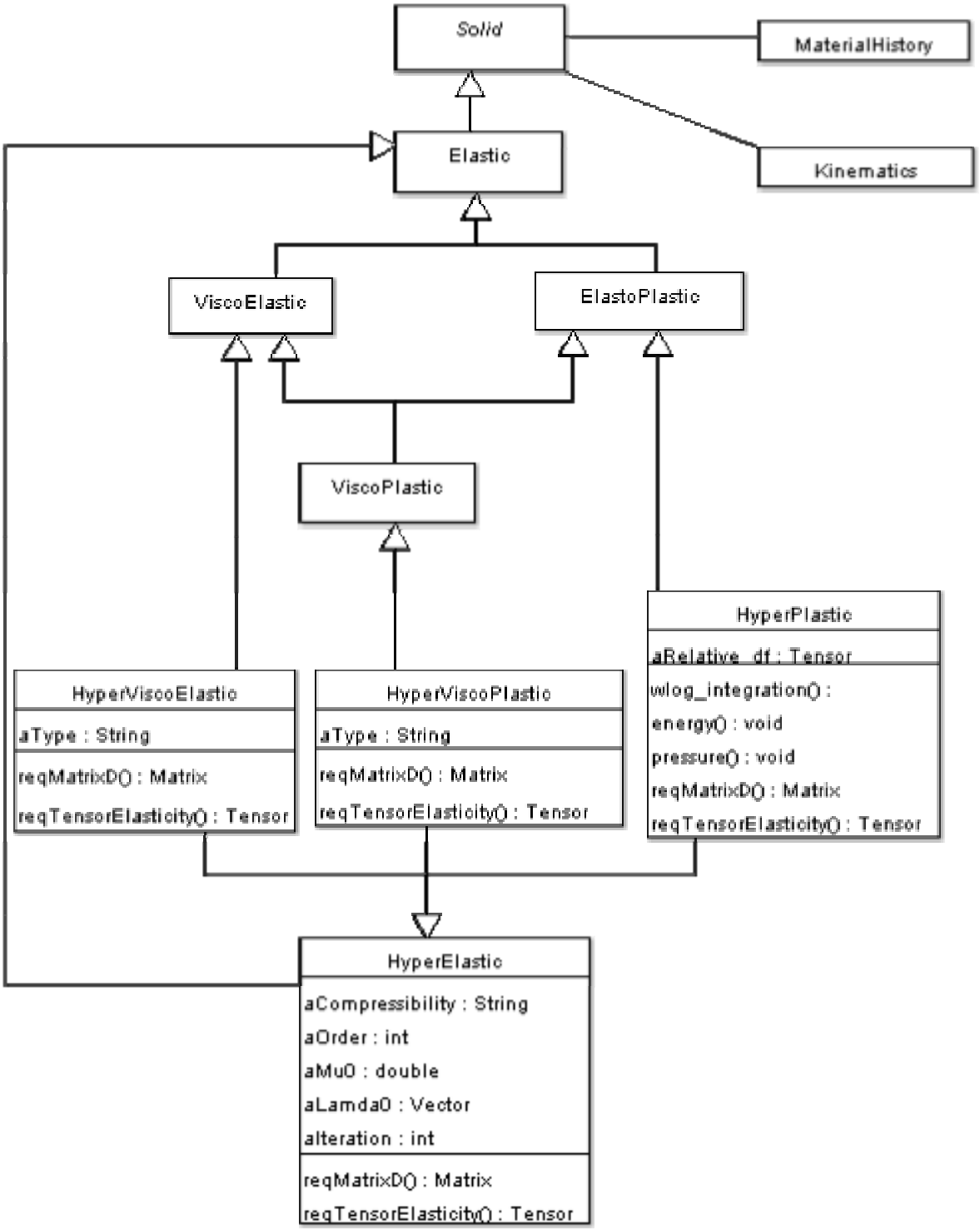

Figure 8 : Detailed diagram for large deformation class.

classes. In these methods, we really integrate the evolution laws of the constitutive equations to find the material tangent matrix or elasticity tensor. As an example, in this section we consider the algorithm of this method in the case of the HyperPlastic class.

Integration of the constitutive laws has been con-

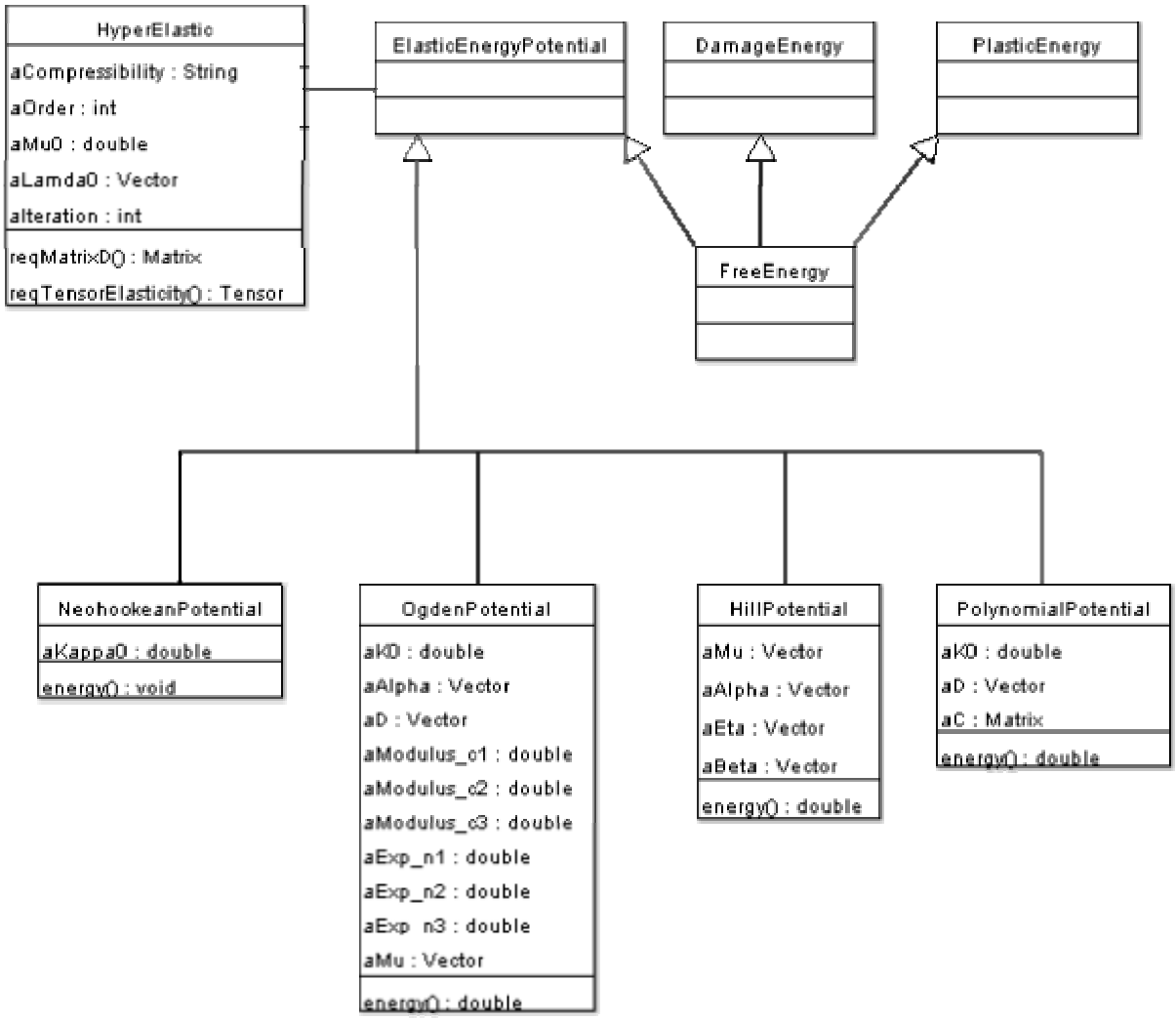

Figure 9 : Large deformation potentials

sidered in several papers [Ortiz(1986), Simo(1991), Simo(1992), Nikishkov(1993), Gharzeddine(1999), Liu(2004), Sun(2004) Akamatsu(2005)]. The algorithm that is used for implementation of the "*reqMatrixD()*" method in the Hyperplastic class is based on the general return mapping in principal axes with an implicit integration method. Our implementation, we used the multiplicative decomposition proposed by Simo [Simo(1985), Simo(1988), Simo(1991), Simo(1992)] for large strain elastoplasticity.

Given a typical time step t_n and displacement increment Δu_{n+1} in a typical time interval $[t_n, t_{n+1}]$, we first assume plastic flow is frozen and evaluate a trial elastic state. Then the trial stress state is tested to see if it is inside or outside of the yield surfaces. If it is outside the elastic zone, the true stress is then found by a closest-point-projection of the trial stress onto the yield surfaces. The problem to be solved can the be stated as follows: consider a typical time sub-interval $[t_n, t_{n+1}]$, and a specific Gauss point $X \in \Omega$, (Ω is the domain of material). Assuming that X has the following history of deformation gradient, left Cauchy-Green tensor and strain like internal variables vector: $\{F_n, b^e_n, \xi_n\}$ at time t_n. We would like to compute: true stress state $\sigma_{n+1}$and history variables $\{F_{n+1}, b^e_{n+1}, \xi_{n+1}\}$ at time t_{n+1} for a given a prescribed incremental displacement u_{n+1}.

11 Algorithm

A. Kinematical Calculation:

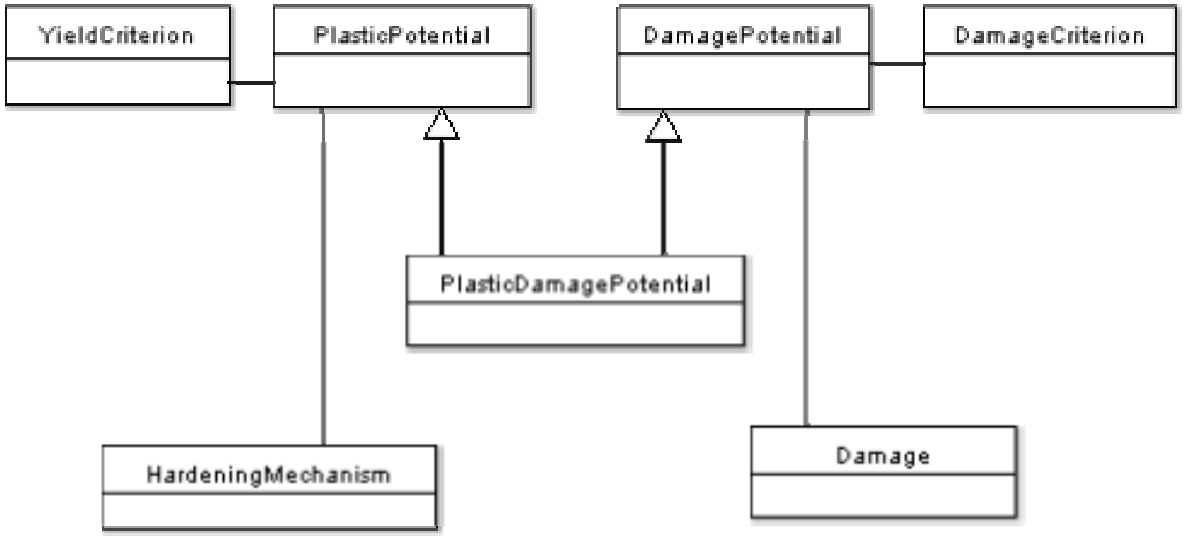

Figure 10 : Large deformation dissipation potential associations

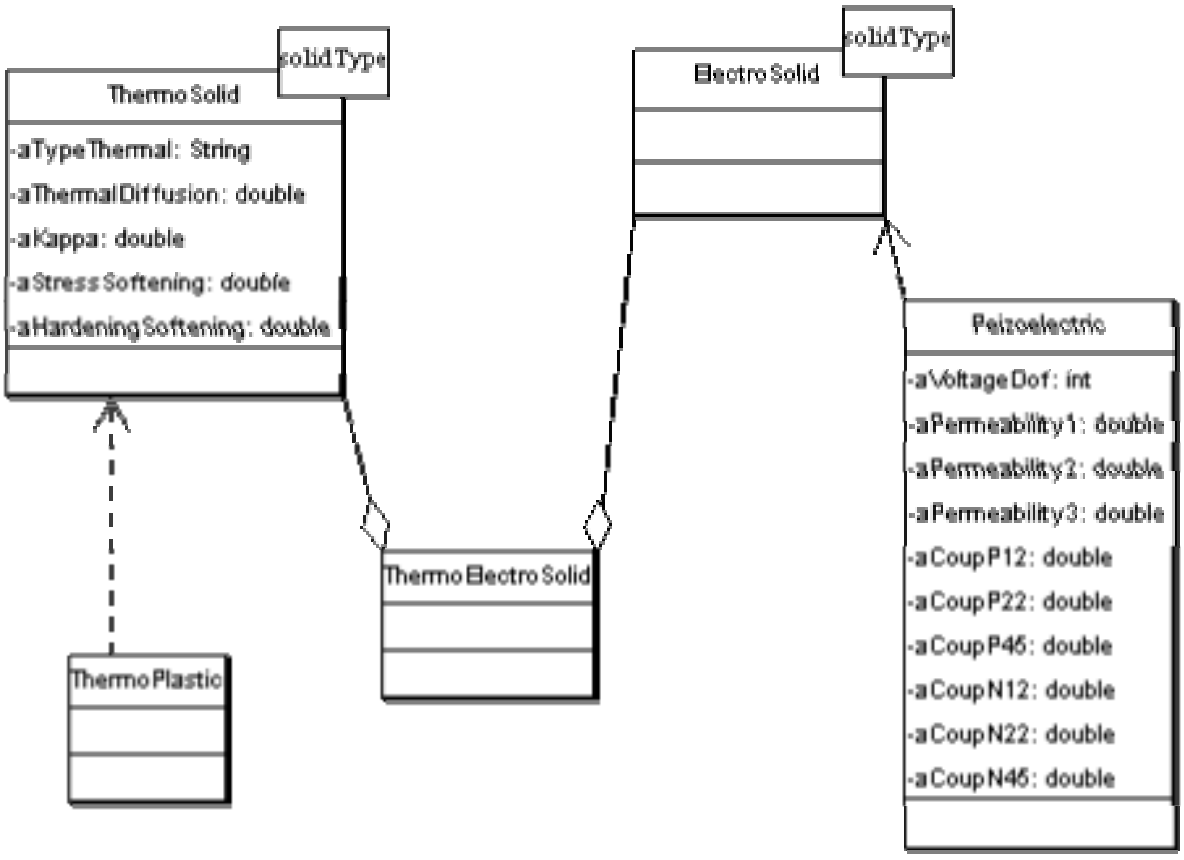

Figure 11 : ThermoElectric class diagram

(using methods of the Kinematics and the MaterialHistory classes)

- Compute the relative deformation gradient:

$$F_u^{n+1} = I + \nabla u_{n+1}$$

I is an identity matrix.

- Evaluate total deformation gradient and its determinant:

$F_{n+1} = F_u^{n+1} F_n$

$J_{n+1} = \det F_{n+1}$

B. Apply the predictor/corrector algorithm for the multiplicative decomposition.

1. elastic trial state (first freeze plastic flow and compute trial elastic response)

1.1 compute the trial left Cauchy-Green tensor

$$b_{n+1}^{etrial} = F_{n+1} b_n^e F_{n+1}^T$$

1.2 perform its spectral decomposition of b_{n+1}^{etrial} by solving an eigen value problem:

- compute invariants I_1, I_2, I_3 :

$$I_1 = tr[b_{n+1}^{etrial}]$$
$$I_2 = \frac{1}{2}(I_1^2 - tr[b_{n+1}^{etrial}]^2)$$
$$I_3 = \det[b_{n+1}^{etrial}]$$

- form the characteristic equation:

$$-(\lambda_A^e)^6 + I_1(\lambda_a^e)^4 - I_2(\lambda_A^e)^2 + I_3 = 0$$

- find principal values λ_A^e, A = 1,...,3

- compute rank-one principal directions basis matrices [Simo(1991)]:

$$n^{(A)} \otimes n^{(A)} = m^{(A)}$$
$$= \frac{(b_{n+1}^{etrial})^2 - (I_1 - (\lambda_A^e)^2) b_{n+1}^{etrial} + I_3 (\lambda_A^e)^{-2} \overleftrightarrow{g}}{2(\lambda_A^e)^4 - I_1(\lambda_A^e)^2 + I_3(\lambda_A^e)^{-2}};$$
$$\text{for } \lambda_1^e \neq \lambda_2^e \neq \lambda_3^e$$
$$n^{(3)} \otimes n^{(3)} = m^{(3)} = \frac{b_{n+1}^{etrial} - (\lambda_1^e)^2 \overleftrightarrow{g}}{(\lambda_3^e)^2 - (\lambda_1^e)^2};$$
$$m^{(1)} = \overleftrightarrow{g} - m^{(3)} \text{ for } \lambda_1^e = \lambda_2^e \neq \lambda_3^e$$
$$n^{(1)} \otimes n^{(1)} = m^{(1)} = \lambda^e \overleftrightarrow{g} \text{ for } \lambda_1^e = \lambda_2^e = \lambda_3^e = \lambda^e$$

here $\overleftrightarrow{g}$ is the metric tensor.

- hence spectral decomposition b_{n+1}^{etrial} becomes:

$$b_{n+1}^{etrial} = \begin{cases} \sum_{A=1}^{3} (\lambda A^e)^2 m^{(A)}, \\ (\lambda_1^e)^2 \overleftrightarrow{g} + ((\lambda_3^e)^2 - (\lambda_1^e)^2) m^{(3)} \\ (\lambda_1^e)^2 \overleftrightarrow{g} \end{cases}$$

$$\det [b_{n+1}^{etrial}] = (\lambda_1^e)^2 (\lambda_2^e)^2 (\lambda_3^e)^2$$

1.3 evaluate the trial logarithmic principal stretches:

$$\varepsilon_{A,n+1=}^{etrial} \ln \lambda^e{}_A,$$

$$\ln J^e = tr[\varepsilon_A^e],$$

$$J^e = (\lambda_1^e)(\lambda_2^e)(\lambda_3^e)$$

1.4 compute elastic principal trial Kirchhoff stresses:

$$\sigma = 2\frac{\partial \psi}{\partial b^e} b^e => \quad \{\tau_{n+1}^e{}^{trial}\} = [a]\{\varepsilon_{n+1}^e{}^{trial}\}$$

where [a] = $\kappa \vec{1} \otimes \vec{1} + 2\mu(I - \frac{1}{3}\vec{1} \otimes \vec{1})$, κ is the bulk modulus and μ is the shear modulus. Due to isotropy, the principal directions of the Kirchhoff stress τ and of the elastic left Cauchy-Green tensor b^e coincide. Hence, in term of the principal Kirchhoff stresses, we can write:

$$\tau = \sum_{A=1}^{3} \beta_A m^A$$

where β_A, A=1,2,3 are principal value of Kirchhoff stress tensor, with deviatoric part:

$$\overline{\beta}_{n+1}^{trial} = 2\mu \overline{\varepsilon}_{n+1}^e{}^{trial}; \quad \overline{\varepsilon}^e = dev[\varepsilon^e] = \varepsilon^e - \frac{1}{3} tr[\varepsilon^e]$$

and pressure:

$$p_{n+1=}^{trial} J_{n+1} \frac{dU(J_{n+1})}{dJ}$$

$U(\lambda_A^e, \xi), A = 1, ..., 3$, is volumetric free energy function.

1.5 Evaluate the trial yield function; (using YieldCriterion classes)

$$\phi_{n+1}^{trial} = \phi(\tau_{n+1}^{trial}, q_{n+1}^{trial})$$

in case of Von Mises:

$$\phi(\tau, q)_{n+1}^{trial} = \left\| dev[\tau_{n+1}^{trial}] \right\| - \sqrt{\frac{2}{3}}[\sigma_y + h(\xi_{n+1}^{trial})]$$

where σ_y is the yield stress, q is stress-like internal variable and h(.) is an isotropic hardening function.

1.6 Testing:

- if trial yield function <= 0 then we have an elastic response:

.perform a trivial updating: (.) $_{n+1}$ = (.)^{trial}

. exit.

- else if trial yield function > 0 , we have a plastic response

. perform a one step return mapping i.e.

. continue to step 2

2. Plastic Correction Phase (perform a plastic corrector). Perform the return mapping algorithm in principal axes to evaluate the elastic stretches ε_{n+1}, total stresses σ_{n+1} and internal variables ξ_{n+1} in a way that is consistent with the plasticity constitutive laws: (using the HardeningMechanism class)

2.1 evaluate the consistency parameter $\Delta\gamma_{n+1}$ at time t_{n+1} using Euler-bakward method:

- evaluate the plastic flow direction from the trial state:

$$\nu_{n+1} = \nu_{n+1}^{trial} = \frac{\beta_{n+1}^{trial}}{\|\beta_{n+1}^{trial}\|}$$

- use updating formula for : stresses and equivalent plastic strain(for isotropic hardening)

$$\tau_{n+1}^{A} = \beta_{n+1} = \beta_{n+1}^{trial} - 2\mu\Delta\gamma_{n+1}\nu_{n+1}$$
$$\xi_{n+1} = \xi_{n+1}^{trial} + \sqrt{\frac{2}{3}}\Delta\gamma_{n+1} \quad (1)$$

- use expression of current value of the yield surface at t_{n+1} and find a combined non linear scalar equation in $\Delta\lambda_{n+1}$ which becomes in case of isotropic hardening:

$$\phi_{n+1} = \phi_{n+1}^{tr} - 2\mu\Delta\gamma_{n+1} - \sqrt{\frac{2}{3}}[h'(\xi_n + \sqrt{\frac{2}{3}}\Delta\gamma_{n+1}) - h'(\xi_n)] = 0;$$

$$h(\xi) = H\xi + [\sigma_y^{\infty} - \sigma_y](1 - \exp[\delta\xi])$$

solve it using a local Newton-Raphson scheme and find $\Delta\lambda_{n+1}$.Here, H is the kinematical hardening coefficient (=0 here) and σ_y^{∞} and σ_y are the yield limit at infinity and the current yield limit of the material.

2.2 Once $\Delta\lambda_{n+1}$ is found , then perform updating of strains and stresses at t_{n+1}:

(using the MaterialHistory class)

- first compute elastic principal stretches:

$$\varepsilon_{n+1=}^{e} \varepsilon_{n+1}^{e\ trial} - \Delta\lambda_{n+1}\nu_{n+1}^{trial}$$

- update internal plastic variables:

$$\xi_{n+1} = \xi_{n+1}^{trial} - \Delta\lambda_{n+1}\partial_q\phi_{n+1}$$

- compute the principal deviatoric Kirchhof stresses:

$$\overline{\beta}_{n+1} = \beta_{n+1}^{tr} - 2\mu\Delta\gamma_{n+1}\nu_{n+1}$$

- recover the total Kirchhof stress tensor from the spectral decomposition:

$$dev[\tau] = \overline{\tau}_{n+1} = \begin{cases} \sum_{A=1}^{3} \beta_{n+1}^{(A)} m_{n+1}^{(A)} \\ \beta_{n+1}^{(1)} \overleftrightarrow{g} + (\beta_{n+1}^{(3)} - \beta_{n+1}^{(1)}) m_{n+1}^{(3)} \\ \beta_{n+1}^{(1)} \overleftrightarrow{g} \end{cases}$$

Add the pressure term and deviatoric part to obtain:

$$\tau_{n+1} = dev[\tau_{n+1}] + J_{n+1}p_{n+1}$$

2.3 Update the intermediate configuration:

- compute the updated left Cauchy-Green tensor

$$b_{n+1}^{e} = \begin{cases} \sum_{A=1}^{3} (\exp\varepsilon_{A,n+1}^{(e)})^2 m_{n+1}^{(A)} \\ (\exp\varepsilon_{1,n+1}^{e})^2 \overleftrightarrow{g} + [(\exp\varepsilon_{3,n+1}^{e})^2 - (\exp\varepsilon_{1,n+1})^2] m_{n+1}^{(3)} \\ (\exp\varepsilon_{1,n+1}^{e})^2 \overleftrightarrow{g} \end{cases}$$

using the principal basis direction and updated logarithmic stretches

- update the right Cauchy-Green plastic deformation tensor C^p on the intermediate configuration:

$$C_{n+1}^{p\ -1} = F_{n+1} b_{n+1}^{e} F_{n+1}^{-1}$$

3. Compute the consistent algorithmic tangent moduli:

3.1 Ignoring the thermal effect, the assumed decoupled volumetric/isochoric form of the free energy can be written as: (using elastic energy classes)

$$\psi = \psi_{vol} + \psi_{isoch}$$

$$\psi_{isoch}(\overline{b}^e) = \frac{1}{2}\mu[J^{e-\frac{2}{3}} tr[\overline{b}^e] - 3];$$

$$\psi_{vol}(J^e) = U(J^e) = K[\frac{1}{2}((J^e)^2 - 1) - \ln J^e]$$

3.2 compute the spatial elasticity tensor:

$$c^e = c^e_{vol} + c^e_{dev}$$

$$c^e_{vol} = (J^e U')' J^e 1 \otimes 1 - 2J^e \mathrm{I}$$

$$c^e_{dev} = 2\overline{\mu}[\mathrm{I} - \frac{1}{3}1 \otimes 1] - \frac{2}{3}||s||[\vec{n} \otimes 1 + 1 \otimes \vec{n}]$$

where: $s = \mu dev[\overline{b}^e]; \vec{n} = \frac{s}{||s||}; \overline{m} = \mu\frac{1}{3} tr[\overline{b}^e]$
then the elastoplastic part is given by:

$$c^{ep} = 2\partial_g dev[\tau_{n+1}]|_{g=1}$$

$$c^{ep} = c^{etrial}_{dev} + c^p_{dev}$$

where:

$$c^{etrial}_{dev} = 2\overline{\mu}[\mathrm{I} - \frac{1}{3}\vec{1} \otimes \vec{1}] - \frac{2}{3}\left\|dev[\tau^{tr}]\right\|(\vec{1} \otimes \vec{n} + \vec{n} \otimes \vec{1}]$$

$$c^p_{dev} = -\alpha^p\{\beta_1 c^{etr} + 2\overline{\mu}\beta_3 \vec{n} \otimes \vec{n} + 2\overline{\mu}\beta_4 \vec{n} \otimes dev[\vec{n}^2]\}$$

where α^p is a plastic loading flag :
α^p = 1 for $\phi^{trial} => 0$;
α^p = 0 for $\phi^{trial} < 0$
the various coefficients β_i , i= 0,1,2,3,4 are defined as:

$$\beta_0 = 1 + \frac{1}{2\mu}h''(\xi_n + \sqrt{\frac{2}{3}}\Delta\gamma);$$

$$\beta_1 = \frac{2\overline{\mu}\Delta\gamma}{\|dev[\tau^{tr}]\|};$$

$$\beta_2 = [1 - \frac{1}{\beta_0}]\frac{2}{3}\left\|dev[\tau^{tr}]\right\|\frac{\Delta\gamma}{\overline{\mu}};$$

$$\beta_3 = \frac{1}{\beta_0} - \beta_1 + \beta_2;$$

$$\beta_4 = (\frac{1}{\beta_0} - \beta_1)\frac{||dev[\tau^{tr}]||}{\overline{\mu}}$$

4. Transfer from Kirchhoff stress to true Cauchy stresses:

$$\sigma_{n+1} = \tau_{n+1}/J_{n+1}$$

$$c^{ep}_{c\ n+1} = c^{ep}_{n+1}/J_{n+1}$$

End of algorithm.

12 Practical use of our model in real application

We convert our UML object-oriented model into a C++ program software package. We use our software in hyperelastic and in hyperplastic Diffpack contact applications. In the hyperelastic application we used an isotropic hyperelastic material with the NeoHookean potential. As a result, we have chosen the HyperElastic class of Large_Deformation package as the type for the material. The HyerElastic object is using the IsotropicSolid class through its grand parent (Solid) to find its isomorphism properties. It also uses its associations with the MaterialHistory and the NeoHookeanPotential objects to find other properties which are needed for the its computation. Figure 12 shows one of our result diagrams. It is the equivalent stress for a matrix in contact with a punch.

In our applications, we have not needed to write the program for the material properties, as the hyperelastic/hyperplastic models have already written in our Material package. As a result, the development time is reduced considerably for developing a new application. Of course, we expend time to develop our material model but this model could be reused for any of our solid material applications using the finite element or the boundary element method.

The difference of the execution time between the object-oriented and the procedural approach is not significant and it is ignorable. As lot of time is used for solving the nonlinear system, the time of access to the data through the object hierarchy is not considerable.

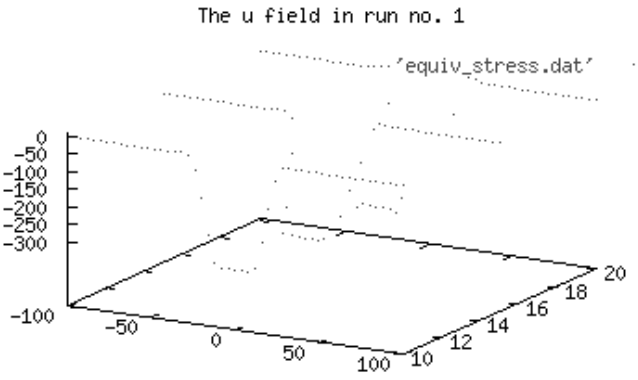

Figure 12 : equivalent stress of matrix in contact to a punch.

13 Conclusion

In this paper, it is shown that the object-oriented modeling helps us to model and to classify the material properties in a convenient way. We consider the behavior of a material under loading to classify the material properties. This type of classification is used in our object-oriented model. We have categorized material properties in two large and small deformation kinematical models. In the small deformation, we model the material behavior in elastic range. On the other hand, the behavior and the characteristic of permanent deformation range are considered in the large deformation model. We have also classified material properties using the rheological categories, namely, Rigid, Elastic, Anelastic, Plastic and Viscous.

The kinematical as well as the hierarchical relationships between different types of materials properties are presentable in our object-oriented model. Our model covers the linear and the nonlinear as well as the isotropic and the anisotropic material. Classes such as Elastic, ElastoPlastic, ViscoElastic and ViscoPlastic have been put in the small deformation package. In large deformation package, we store classes such as HyperElastic, HyperPlastic, HyperViscoElastic, HyperViscoPlastic.

We have presented our model graphically using the UML notation, which is more understandable than a C++ code. It makes our object-oriented model not only usable for the software engineering, but also it can be used in the educational and the research objectives. Using C++ programming language, we implemented our model in a hyperelastic/hyperplastic Diffpack application. It is considerably reduced the programming effort to develop the nonlinear material contact applications.

References

ABAQUS/Standard user's manual (2005): ABAQUS/Standard user's manual. Hibbitt,

Akamatsu, M.; Nakane, K.; Ohno, N. (2005): An Implicit Integration Scheme for a Nonisothermal Viscoplastic, Nonlinear Kinematic Hardening Model, CMES:Computer Modeling in Engineering & Sciences, vol. 10, no. 3, pp. 217-228.

ArgoUML (2006): ArgoUML home page. http://argouml.tigris.org/index.html.

Besson, J.; Foerch, R. (1997): Large Scale Object-Oriented Finite Element Code Design, Computer Methods in Applied Mechanics and Engineering. vol. 142, pp. 165-187.

Bettig, B. P.; Han, R. P. S. (1996) : An object-oriented framework for interactive numerical analysis in a graphical user interface environment. International Journal for Numerical Methods in Engineering, vol. 39, pp. 2945-2971.

Cross, J. T.; Masters, I.; Sukirman, Y.; Lewis, R.W. (1997): Object-oriented Programming Techniques for Finite Element Methods in Heat Transfer. In R.W. Lewis and J. T. Cross (eds.) Proc. 10th Int. Conf. for Num. Meth. Thermal Problems, Swansea, pp. 757-766.

Diffpack (2006): Diffpack home page. http://www.diffpack.com/products/prod_main. html.

Dubois-Pélerin, Y.; Zimmermann, T. (1993): Object-oriented finite element programming: III. An efficient implementation in C++. Comp. Meth. Appl. Mech. Eng., vol. 108, pp. 165-183.

Foerch, R.; Besson, J.; Cailletaud, G.; Pilvin, P. (1997): Polymorphic Constitutive Equations in Finite Element Codes. Computer Methods in Applied Mechanics and Engineering. vol. 141, pp. 355-372.

Gharzeddine, F.; Ibrahimbegovic, A.; Chorfi, L.; Gakwaya, A. (1999): Formulation des théories de grandes déformations dans les axes principaux et leur implantation numérique. 4ème Colloque National en Calcul des Structures.

Graham, I. (2001): Object-Oriented Methods, Principles & Practice. Third Edition, Addison-Wesely.

Kong, X.A.; Chen, D.P. (1995): An object-Oriented design of FEM programs. Computer & Structures, vol. 57, no. 1, pp. 157-166.

Langtangen, H. P. (1999): Computational Partial Differential Equations, Numerical Methods and Diffpack Programming. Lecture Notes in Computational Science and Engineering, vol. 2, Springer-Verlag.

Liu, C. S.; Chang, C. W. (2004): Lie Group Symmetry Applied to the Computation of Convex Plasticity Constitutive Equation, CMES: Computer Modeling in Engineering & Sciences, vol. 6, no. 3, pp. 277-294.

Masters, I.; Cross, J. T.; Lewis, R. W. (1997): A Brief Review of Object-Oriented Finite Element Methods. In R.W. Lewis and J. T. Cross (eds.) Proc. 10th Int. Conf. for Num. Meth. Thermal Problems, Swansea, pp. 766-

776.

Muller, P. A. (1997) : Instant UML. Wrox Press.

Nikishkov, G. P.; Atluri, S.N. (1993): Implementation of a Generalized Midpoint Algorithm for Integration of Elasto-Plastic Constitutive Relations for Von Mises' Hardening Material. Computers and Structures, vol. 49(6), pp. 1037-1044.

Nikishkov, G. P. (2006): Object Oriented Design of a Finite Element Code in Java, CMES: Computer Modeling in Engineering & Sciences, vol. 11, no. 2, pp. 81-90.

Northwest Numerics (2005): Constitutive Equations. In http://www.nwnumerics.com/ PDFs/info/z-matdetail.pdf, web site of Northwest Numerics and Modeling Inc.

Ortiz, M.; Simo, J. C. (1986): An analysis of a new class of integration algorithms for constitutive relations. Int. J. Meth. Eng., vol. 23, pp. 353-366.

Pawtucket, R.I.: Karlsson & Sorensen, Inc. Volume I, Version 5.8.

Rational Rose (2006): Rational Rose Software home page. http://www-306.ibm.com/ software/awdtools/developer/modeler/, IBM Inc.

Sharifi, H.; Gakwaya, A. (2005): Object-oriented Modeling of Field Boundary Element Method in Nonlinear Solid Mechanics with Applications. In Advances in Boundary Element Techniques VI, Editors: A P Selvadurai, C L Tan, M H Aliabadi, EC Ltd United Kingdom, pp. 317-325.

Simo, J. C,; Taylor, R.L. (1991): Quasi-incompressible finite elasticity in principal stretches: continuum basis and numerical algorithms. Comp. Methods Appl. Mech, Eng., vol. 85, pp. 273-110.

Simo, J. C. (1985): On the computational significance of intermediate configuration and hyperelastic stress relations in finite deformation elastoplasticirty. Mech. Mater. vol. 4, pp. 439-451.

Simo, J. C. (1988): A framework for finite strain elastoplasticity based on maximum plastic dissipation and multiplicative decomposition: Part I. Continuum formulation; Part II. Computational aspects. Comput. Methods Appl. Mech. Eng., vol. 66, pp. 199-219, and vol. 68, pp. 1-31.

Simo, J.C. (1992): Algorithms for static and dynamic multiplicative plasticity that preserve the classical return mapping schemes of the infinitesimal theory. Comp. Meth. Appl. Mech. Eng. vol. 99, pp. 61-112.

Simo, J.C.; Govindjee, S. (1991): Non-linear B-stability and symmetry preserving return mapping algorithms for plasticity and viscoplasticity. Int. J. Numer. Meth. Eng., vol. 31, pp. 151-176.

Sun, X. S.; Huang, L. X.; Liu Y. H.; Cen, Z. Z. (2004): Elasto-plastic Analysis of Two-dimensional Orthotropic Bodies with the Boundary Element Method ,CMC: Computers, Materials, & Continua, vol. 1, no. 1, pp. 91-106.

Toukourou, M. M.; Gakwaya, A.; Yazdani, A. (2001): Object-oriented finite implementation of large deformation frictional contact problems and applications. Computational Fluid and solid Mechanics. pp. 365-368.

Wampler, B. E. (2002): The Essence of Object-Oriented Programming with Java and UML. Addison-Wesley.

Zabaras, N.; Srikanth, A. (1999): Using Objects to Model Finite Deformation Plasticity. Engineering with Computers. vol. 15, pp. 37–60.