Toward a Multi-Model Hierarchy to Support Multiscale Simulations

Mark S. Shephard and E. Seegyoung Seol

Scientific Computation Research Center

110 8th Street, Troy, NY 12180, U.S.A.

November 21, 2005

1 Introduction

There is a long history on the development of mathematical representations capable of providing behavioral predictions of physical parameters on the atomic, molecular, microscopic, and macroscopic scales. Over the past half century, simulation programs have been developed to support the computerized solution of these mathematical representations which, in some cases, are discretized with billions of degrees of freedom and solved on massively parallel computers with thousands of processors. Historically, scientists and engineers have applied these models (simulation programs) to solve problems on a single physical scale. However, in recent years it has become clear that to continue to make advances in the areas of nanotechnology and biotechnology and to develop new products and treatments based on those advances, scientists and engineers must be able to solve sets of coupled models active over multiple interacting scales. For example, the development of new materials will require the design of structure and function across a hierarchy of scales starting at the molecular scales to define nanoscale building blocks that are used to define mesoscale features that are combined into micron-scale weaves that used in the manufacturing of the complete part (Figure 1). Such capabilities are clearly central to the development of nanoelectronics devices and future drug delivery systems (Figure 2), as well as many of other future products. For example, consider new automotive skins made of nano-reinforced materials in which the material interfaces are strong at strain rates consistent with normal usage, leading to high stiffness so the little dents are avoided, while at high strain rates the interfaces demonstrate substantial local damage, thus providing high energy absorption under impact loading to keep the individuals in the passenger compartment safe.

<INSERT FIGURE 1 HERE>

<INSERT FIGURE 2 HERE>

Although there are a large number of models available to solve various single scale simulation problems, there is little or no support for the multi-model systems needed for multiscale problems. Considering the thousands of person-years of effort that has gone into the development of these existing models, the effective development of multiscale simulations requires the use of the existing and developing single-scale simulation models. Thus, the only practical approach is to construct a component-based multi-model system, where each model is developed as a component using clearly defined interfaces and functionality for sharing information with other models that will allow the effective integration of many models.

The successful definition of these component-based models must begin with the careful decomposition and abstraction of the basic functional models and a clear qualification of the functional and informational interfaces needed to support their interactions. The goal of this chapter is to present a high-level description of such a decomposition and abstraction to support multiscale simulation of problems defined over general space-time domains.

The next section considers the key functional and information hierarchies of a multiscale simulation. Section 3 discusses the overall design of the six functional components defined to support the full set of interactions and transformations needed by multiscale simulation. It is the combination of these six functional components with the existing single-scale models that will provide an operational multi-model multiscale simulation system. Section 4 presents two example simulations development building on initial prototypes of these interfaces and existing singl- scale models.

2 Functional and Information Hierarchies in Multiscale Simulation

The abstraction of multiscale simulation processes must consider the hierarchy of transformations required to go from a mathematical description of the physical behavior to computer models used to solve, at least approximately, those mathematical descriptions. The highest level in the hierarchy is the mathematical descriptions as a set of governing equations used to describe the typically-coupled physical behavior at the different scales including equations relating parameters between the scales. The other two levels in the hierarchy are the discretizations and numerical algorithms used to solve, often approximately, those equations using computer models. A key to the abstraction of this process is qualification of the information needed to support the models and the transformations required as information is shared by models. The information used in the processes can be placed in the following two groups:

- *Domain definitions*: The description of the domains over which the various physical descriptions apply. In the case of multiscale analysis, this includes appropriate definitions at each scale, and the spatial temporal interactions between them.

- *Physical parameter definitions*: The description of the physical parameters, defined over the appropriate domains, that are needed to qualify the specific instances of the governing equations to be solved.

The ability to properly support component-based multi-model multiscale simulation requires the specification of mathematical descriptions with associated domain and physical parameter definitions at the highest possible level meaningful to the execution of the process so that the full range of methods of solution and interaction between models can be supported.

2.1 Mathematical Physics Description Transformations and Interactions

At the highest level, a mathematical physics description is a set of governing equations that are assumed to govern the behavior at a particular scale over a particular domain. The physics descriptions are written in terms of a set of dependent variables and given parameters, and are a function of the coordinates of the domain of the problem of interest. To make this more concrete, consider the two most common forms of equations encountered in multiscale analysis which are partial differential equations (PDEs) that are defined at various continuum scales and molecular dynamics (MD) which is based on interatomic potentials that define the interactions of discrete atoms at atomic scales.

2.1.1 PDEs

PDEs may be written in terms of multiple sets of dependent variables where each of them can be tensor quantities of various orders. For the purposes of this discussion, consider the domain PDE:

$$\mathcal{D}^m(u, \sigma) - f = 0 \text{ in } \Omega$$

subject to boundary conditions

$$\mathcal{D}^i(u, \Psi) - g_i = 0 \text{ on } \Gamma_i, i = 0, 1, 2, \ldots, m-1$$

where

$\mathcal{D}^m$ represents the appropriate m^{th} order differential operator.

$u(x,t)$ represents one or more vector dependent variables which are functions of the independent variables of space, x, and time, t.

Ψ represents one of more scalar dependent variables which are functions of the independent variables of space, x, and time, t.

f represents the forcing functions.

Ω represents the domain over which the equation is defined.

$\mathcal{D}^i$ are the appropriate i^{th} order differential operators.

g_i are the given boundary conditions.

Γ_i are the portions of the boundary over which the associated boundary conditions act.

The computerized models of the PDEs typically use mesh-based methods in which the first set of transformations is a double discretization process in which the dependent variables are discretized over mesh entities that represent the primary transformation of the domain to be solved (see §2.2), either by direct operator discretization (e.g., difference equations) or in terms of a set of basis function substituted into a weak form of the PDEs. In both cases, this process specifies a set of distribution functions over the mesh entities for the variables written in terms of a yet to be determined set of multipliers, called degrees of freedom (dof). The dof can always be associated with a single mesh entity while the distribution functions are associated with one or more mesh entities. In the case where the distribution is associated with multiple mesh entities, that set is defined by rules associated with the discretization operator and can be supported by using mesh adjacency information. Three common cases that employ different combinations of interactions between the mesh entities, the dof, and the distributions are:

- *Finite difference based on a vertex stencil*: In finite difference methods the distribution functions are difference stencils where the dof are typically values of the dependent variables at vertices in a mesh.
- *Finite volume methods*: Finite volume methods are constructed in terms of distribution function written over individual mesh entities. In most cases the field being defined is C^{-1} and dof are not shared between neighboring mesh entities. The coupling of the dof from different mesh entities is

through operators acting over common boundary mesh entities.

- *Finite elements*: Finite element distribution functions are written over individual mesh entities, called elements. In cases where C^m, $m \geq 0$ continuity is required, the distribution functions associated with neighboring elements are made C^m, $m \geq 0$ continuous by having common dof associated with the bounding mesh entities common to the neighboring elements.

The application of the discretization operation over the mesh entities produces a local contributor which can be stated symbolically as:

$$k^c d^c = f^c$$

where k^c is the discretized matrix for contributor C that multiplies the vector of dof associated with the contributor, d^c.

These individual contributions are then assembled into a global algebraic system, $Kd = F$, based on an assembly operator defined by the relationships of the contributor level dof, d^c , with the assembled set of global dof, d.

2.1.2 Molecular Dynamics

In molecular dynamics (MD), the mathematical model is a potential function describing the forces between interacting atoms that depends on the relative position of the atoms [17]. An example of a common potential function is the

Lennard-Jones potential defining the force between two atoms given by

$$V_{LJ} = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$$

where σ and ϵ are the Lennard-Jones parameters for a given material and r is the inter-atomic distance. The parameters in the potential equations may be developed empirically or based on simulations performed on the finer *ab initio* scale. Because of the number of atoms involved in fill domains, MD simulation is typically performed over small subdomains where boundary conditions must be applied to the atoms on and/or near the boundary. Typical boundary conditions are free-surface, periodic and fixed boundary conditions. The output of MD simulations, the atom trajectories and forces on the atoms, are typically not of specific interest, but are needed to determine the parameters meaningful at the higher scales of interest. The extraction of those parameters often requires a set of statistical ensembles.

2.1.3 Interactions Between PDEs and MD

It is common for a simulation to require the solution of a set of coupled mathematical models where the coupling is defined by parameters assumed to be given in one model and are actually the dependent (to be solved for) parameters in another model. In some cases, the coupling simply requires solving the models in a given order so that the required given parameters are available when the model of interest is to be solved. In other cases the coupling is two way in that parameters are shared in both directions, thus necessitating the application of an appropriate coupling method.

Coupling on a single scale occurs when multiple models are used to solve for different sets of the physical parameters of interest. A common example is fluid structure interactions where the flow field is influenced by the geometry of the structure it is flowing over, and the geometry of the structure is a function of the forces on it created by the flow field going over it. The issues associated with the transfer of parameters between the models depend on what portions of the domain the interactions occur over, and how that portion of the domain has been discretized both in terms of its geometry (mesh) and the distributions and dof used.

The interactions of parameters between models solved on multiple scales must account for differences of the domain representation of the different scales, the models used to couple information between the scales, and the relationships between the parameters passed between the models on the different scales. Two broad classes of scale linking methods are information-passing and concurrent bridging [13]. In the concurrent bridging, both the fine and coarse scales are simultaneously resolved. In the information-passing methods, fine scales are modeled and their gross response is infused into the coarse scale, and the influences of coarse-scale fields on the fine scales are taken into account. For nonlinear problems, the models at different scales are coupled in both directions and the information continuously flows between the scales.

In many message-passing techniques the fine scale model is a representative unit cell subject to appropriate boundary conditions and the information passed to the larger scale is assumed to be a point on that scale. In the concurrent techniques the fine scale model acts over some small, but finite, portion of the domain on the coarse scale and the parameters are passed through the common boundary between the domains, or through some overlap portion of the domains.

In multiscale methods where entirely different models are used on the different scales, the relationship of the parameters between the scales is usually not direct and specific care must be taken to define the appropriate operators to relate them. In some cases, these operators act as filters to remove specific information content (e.g., the removal of high-frequency modes when up-scaling). In others they must account for relating discrete and continuum models (e.g., relating atomic-level deformations defined by atomic positions to a continuum displacement field). In some cases, operators are needed to relate quantities with different forms of definition (e.g., atomic-scale forces to continuum stresses).

The complication of properly relating information across multiple scales has led to the active development of methods for scale linking and their implementation as computer models. Representative information-passing methods include multiple-scale asymptotic techniques [14], variational multiscale methods [22], heterogeneous multiscale method [11], multiscale enrichment schemes based on partition of unity [12], discontinuous Galerkin discretizations [21], and the equation-free method [24]. Spatially-concurrent schemes are based on either multilevel [15] or domai- bridging methods [4, 8], while the concurrent schemes in the time domain are typically based on multistep methods [18].

2.2 Domain Definitions, Transformations and Interactions

The domains considered here are space-time domains. Since time is a linear progression that runs from an initial time to a final time, it is simple to represent. On the other hand, there are a number of general forms commonly used to provide a high level representation of spatial domains. To meet the needs of multiscale simulation the space-time domain representations must be able to:

- Support the transformation of the original domain definition into the sets of interacting representations used to support the discretization of the governing equations over the domain and maintain the relationship between each of the representations.
- Support the definition of the physical parameters (attributes) associated with the equations to be solved and the proper transformation of that information into any derived representation used by the models.
- Support the ability to address any domain interrogation required during

the execution of models involved with the simulation.

- Support the geometric interactions between related domains used in a multiscale simulation.

The definition of the domain is a function of the type of mathematical description used. For example, continuum domain definitions are needed in the case of PDEs while a discrete set of atomic positions are needed in MD.

2.2.1 Continuum Domains

There are multiple sources for domain definitions with CAD models, mesh models and image data being the most common. CAD systems and mesh models employ some form of boundary representation. Image data is defined using a volumetric form such as voxels. Except in cases of directly using the image data as the model, it is generally accepted that the use of a boundary representation is well suited for the domain definition in continuum level simulations. Common to all boundary representations is the use of the abstraction of topological entities and their adjacencies to represent the entities of different dimensions. The information defining the actual shape of the topological entities can be thought of as information associated with the entity. The ability to interact with the domain definition in terms of the topological entities provides an effective means to develop abstract interfaces to the domain definition allowing the easy integration of multiple domain definition sources.

In addition to the topological entities and associated shape information, geometric modeling systems maintain numerical tolerance information on how well the entities actually fit together. The algorithms and methods within the geometric modeling system are able to use the tolerance information to effectively define and maintain a consistent representation of the geometric domain. (The vast majority of what various geometry-based applications have referred to as dirty geometry is caused by a lack of knowledge or by improper use of the tolerance information [3].)

< INSERT FIGURE 3 HERE >

The abstraction of topology provides an effective means to develop functional interfaces to boundary-based modelers that are independent of the specific shape information. The developers of CAD systems have recognized the possibility of supporting geometry-based applications through general APIs. This has lead to the development of geometric modeling kernels such as ACIS and Parasolid. These geometric modeling APIs have been successfully used to develop automated finite element modeling processes [40, 43] and automatic mesh generators [3].

In the application of generalized numerical analysis processes, the geometric domain must be transformed into a mesh that approximates the domain.

To support a full set of operations needed for reliable multiscale analysis, the mesh must maintain an association with the continuum domain representation, and with the distribution functions and dof used in discretizing the PDEs (see §2.1.1). From the perspective of maintaining its relationship to the geometric domain, the use of an appropriate set of topological entities and their adjacency is ideal [2]. In cases where there is a higher-level structure associated with the definition of the mesh, a simplified representation based on that structure can provide the needed information and relationships back to the continuum domain definition.

A key component of supporting mesh-based simulation is the association of the mesh to the geometric model [2, 36]. This association can be defined as follows:

Classification: The unique association of mesh topological entities of dimension d_i, $M_i^{d_i}$ to the topological entity of the geometric model of dimension d_j, $G_j^{d_j}$ where $d_i \leq d_j$, on which it lies is termed classification and is denoted $M_i^{d_i} \sqsubset G_j^{d_j}$ where the classification symbol, $\sqsubset$, indicates that the left hand entity, or set, is classified on the right hand entity.

Reverse Classification: For each model entity, G_j^d, the set of equal order mesh entities classified on that model entity define the reverse classification information for that model entity. Reverse classification is denoted as

$$RC(G_j^d) = \{M_i^d \mid M_i^d \sqsubset G_j^d\}$$

Shape information can be effectively associated with the topological entities defining the mesh. In many cases this is limited to the coordinates of the mesh vertices and, if they exist, higher-order nodes associated with mesh edges, faces or regions. In addition, it is possible to associate other forms of geometric information with the mesh entities. For example, the association of Bezier curves and surface definitions with mesh edges and faces for use in p-version finite elements [28]. The mesh classification can be used to obtain other needed geometric information such as the coordinates of a new mesh vertex caused by splitting a mesh edge classified on a model face.

2.2.2 Discrete Domains

The domain definition for the discrete models are the positions of the entities for which the potentials are written to relate. For example, in the case of MD this is the position of atoms. In many cases it is possible to define the full set of discrete entity positions from a higher-level construct with appropriate transformations. In this case the highest-level domain definition consists of the geometry of the domain to be included, parameters defining the distribution of the discrete positions and the functions required to define those positions. The overall domain is often a representative volume that has portions of its boundary interior to a higher-level domain and may include knowledge of free surfaces.

The parameters and transformations used to define the atomic positions are a function of the type of material being defined. In the case of perfect crystals, the position of atoms within each crystal are defined by a set of lattice vectors. The definition of the geometric configuration of the crystal is a nontrivial process that can start with a statistical method to define an initial set of seed locations for crystals whose initial shape can then be defined as the Voronoi diagram of those points. To define more-realistic configurations various grain growth procedures that account for knowledge of the material system can be applied. There can be defects in the crystal systems [23] and by providing additional information about these defects the total number of particles, coordinates and velocities of the particles can have an initial adjustment applied to them. In the case of polymeric materials, the atomic positions must be defined by their position along a molecular chain where there are strong bounds between neighboring units in the chain. Statistically-based geometric constructs can be used to define these material-dependent chains in the simulation box.

One of the methods used to bridge scales is to combine sets of atoms into unit that they can be defined in terms of a smaller number of discrete points. One such approach well suited to lattice structures is the quasi-continuum methods where the movement of atoms over simple shapes (triangles, tetrahedra) is described to vary linearly [25, 30]. In the case of polymeric chains, the atoms along a chain are represented by a small number of beads placed along the chain

[29, 33].

2.2.3 Interactions of Domains

There are three general forms of domain interactions used in multiscale simulations. They are:

- *Disjoint domains* that share information across a common boundary
- *Overlapping domains* where the higher scale domain overlaps all or part of the finer-scale domain and the information is shared through the overlapped region
- *Telescoping domains* where the finite, but very small with respect to the higher scale domain, small-scale domain passes information to a point in the higher scale domain

In each case the operations used to transfer parameters between the scales must be consistent with the form of domain interaction.

2.3 Physical Parameter Definitions, Transformations and Interactions

The physical parameters used in the mathematical equations are tensor quantities [5] defined over various portions of the domain that can be general functions of the independent variables of space and time as well as other dependent variables. Knowledge of the order of a tensor and the dimension of the spatial domain it is defined over, defines the number of components needed to uniquely define the tensor. The symmetries, for tensors of order two or greater, define those components that are identical to, or negative of (anti-symmetric), other components. The components of the tensor are in general functions of the domain parameters as well as other problem parameters. The ability to understand and use a tensor at any particular instant requires knowledge of the coordinate system in which the components are written. Tensors can be represented in other coordinate systems of equal or lower order through appropriate coordinate transformations.

To support the full range of simulation needs, the tensors used to define the equations parameters must be related to the highest level of the geometric representation to which they can be defined. For example, in the case of solving PDE over continuum domains, the distribution of the given input tensors needs to be related to the entities in the geometric model. The model topological entities of regions, faces, edges, and vertices are ideally suited for supporting that specification in a general way.

The tensors associated with the dependent parameters are determined as part of the solution process. Therefore these tensors, referred to as fields, are understood with respect to the spatial and equation discretizations used in the simulation process. Since the spatial discretizations are required to maintain the relationship to the original domain definition (see §2.2), the fields can also be related to the highest-level domain definitions.

In multiscale simulation, a single tensor field can be used by a number of different analysis routines that interact and the field may be associated with multiple spatial discretizations (e.g., meshes) having alternative relationships between them. In addition, different distributions can be used by a field to discretize its associated tensor. The ability to have a specific tensor defined over multiple meshes and/or discretized in terms of multiple distributions can be handled by supporting the concept of multiple field instances.

3 Constructing a Multi-Model: Design of Functional Interfaces to Support Multiscale Simulations

In the design of a multi-model to support multiscale simulations, it is important to determine the basic information and operations required by the model used and provide components to support the information and operations on it. From a programming view point, each component is a library that can execute independently of other parts such that constructing a model is done in a plug-

and-play manner. Components interact with other components only through the interoperable APIs.

The creation of the multi-model needed to support multiscale simulation must:

- Focus attention on the needs of the models used in the simulations.
- Recognize and take direct advantage of the simulation steps that are well handled by existing single scale model programs.
- Identify the appropriate abstractions to effectively support the flow of information between components within the process such that any required transformations and models can be supported and any proper models can be inserted.

The goal of the components being designed is to support reliable multiscale simulations where explicit consideration is taken of controlling the approximation errors that arise within each step in the process. Since many of these errors cannot be controlled through *a-priori* means, it is necessary to provide adaptive feedback using *a-posteriori* information in the execution of each model.

A number of the models needed to perform specific simulation steps are well established and, because of the time and effort associated with their development, must be used to advantage. Two such models are generalized fixed mesh continuum PDEs solvers (finite element, finite volume, and finite difference) and discrete level models for solving discrete-potential systems (*ab initio*, molecular statics, molecular dynamics). The majority of the mature and heavily-used of these codes operate only though fixed structure input and output files. In some cases, the programs do support the addition of user-defined routines for specific functions. For example the ABAQUS [1] supports the development of user-defined material models and user-defined finite elements. Although limited, these two routines support the effective addition of the majority of the functionalities needed for ABAQUS to be an effective model in a multiscale simulation environment.

Another area where there are mature programs is the definition of geometric domains in terms of 3-D solids using boundary representations. The most commonly used of these systems are built on a functional API [35, 41] that is ideal for use to support component-based procedures. A number of models are also available for the generation of the mesh level discretizations of the geometric domains. The interfaces to these procedures range from file-based to API-based [3]. The API-based methodologies have been used in the development of adaptive mesh modification procedures [26] and complete adaptive PDE solvers [40, 43].

Considering the informational and functional needs of multiscale simulations in combination with the available models that can effectively support specific of the simulation tasks, the needed components are focused on information flow and transformation within a multi-model simulation. The components will support the definition and control of the

1. overall problem,
2. equation parameters,
3. geometric domains,
4. discretized geometric domains,
5. tensor fields, and
6. scale linking operations.

The concept of defining a similar set of functional components to support the interoperability of simulation models is a topic of current development for mesh-based continuum simulation methods both in terms of open-source code [9, 42] and commercial products [38]

< INSERT FIGURE 4 HERE >

Figure 4 illustrates a structure of a multi-model composed of the six components used in each model where n is the number of models constructing the multiscale multi-model. Each model uses instances of the six components. A multiscale simulation process requires the application of multiple interacting instances of the six functional components. As depicted in the figure, each instance of components interacts with instances of other components of different type within a model (dashed lines), and instances of component of the same type in other models (solid lines).

The explicit consideration of the relationships that must be supported between the information and operations associated with each component and the models they interact with is critical. Therefore, the API of functions of a component has the three categories:

- APIs for providing and modifying component-internal data (APIs which don't involve interactions with other instances),
- APIs for interaction between components, and
- APIs for interaction of components of the same type between models.

3.1 Overall Problem Definition

To support multiscale simulations, a high-level problem definition is needed. The problem definition must include all of the physical parts involved and support the specification of general sets of relationships between the parts and linkage to alternative representations of the parts. The structures and methods used to support the problem definition must be able to support the morphing into viewpoint-specific forms of the problem definition as needed for specific models. In these viewpoints, parts may be decomposed into additional pieces and additional information supporting the viewpoint added. For purposes of this discussion, the basic viewpoint of interest is multiscale simulation where parts may be decomposed to support the definition of problems at different scales or simulation idealizations needed by the various models.

Representations including simulation viewpoints have begun to be developed [39]. The definition of such representations can be supported by a graph-like structure similar to those used to define assembly and feature models in CAD systems [6, 19] with the extensions necessary to support hierarchal decompositions and multiple viewpoints [7, 19, 20, 31].

Its functioning API must include

1. a component-internal API: (*i*) part definitions, (*ii*) relationships of parts
2. an API for interacting with other components: (*i*) relationships of parts to domains, (*ii*) relationships of parts to parameters, (*iii*) relationships of parts to model functions and scale linking, and
3. an API for interacting with other instances of the overall problem definition: (*i*) viewpoint construction rules.

3.2 Equation Parameters

The parameters used in the mathematical equations represent physical quantities best described by tensors. The types of physical parameters these tensors define are material properties, loading functions, boundary conditions and initial conditions. Although the execution of any model requires the specific set of these tensors associated that the mathematical equations of that model, the most general means for defining these tensors is to state them in terms of the problem definition entities they are associated with. Generalized methods to define and manipulate these tensors have been defined [32, 37].

Its functioning API must include

1. a component-internal API: (*i*) parameter information queries, (*ii*) parameter instance information queries, (*iii*) parameter coordinate transformation, (*iv*) parameter reduction and modification,
2. an API for interacting with other components: (*i*) relation to problem parts, (*ii*) relation to model solution process, (*iii*) relation to fields, and
3. an API for interacting with other instances of the equation parameter component: (*i*) dependencies between parameters.

3.3 Geometric Domain

The geometric domain component is a functional unit to describe the multiscale simulation domain at a scale which is either a continuum domain or discrete domain. Within a multi-model, it supports the geometric interactions between instances.

Consider first the API of the continuum geometric domain component defined in a CAD modeler which uses boundary representations. Its functioning API must include

1. a component-internal API: (*i*) topological entity queries, (*ii*) shape information, (*iii*) geometric model tolerance information,

2. an API for interacting with other components: (*i*) association with parts in the problem definition, (*ii*) association of equation parameters with the geometric domain, (*iii*) association with domain discretizations (meshes), (*iv*) association with scale linking, and

3. an API for interacting with other instances of the geometric domain component: (*i*) geometric interactions relating domains on different scales through boundaries and/or overlaps.

For atomic-scale models:

1. a component-internal API: (*i*) the definition of atom layouts,

2. an API for interacting with other components: (*i*) obtain potentials, (*ii*) provide forces, and

3. an API for interacting with other instances of the geometric domain component: (*i*) placement of domain with respect to larger scale domains, (*ii*) geometric interactions relating domains of different scales through boundaries and/or overlaps.

3.4 Discretized Geometric Domains

The discretized geometric domain component is a piecewise geometric representation of the corresponding geometric domain component in terms of a mesh or atomistic layout.

The API must include

1. a component-internal API: (*i*) topological entity queries for meshes, atom queries such as position of atoms and distance between atoms for atomistic,
2. an API for interacting with other components: (*i*) mesh shape information, (*ii*) association with the geometric domain, (*iii*) association with fields, and
3. an API for interacting with other instances of discretized domain: (*i*) mesh to mesh interaction, (*ii*) mesh to atomistic interactions, (*iii*) discrete to atomistic interactions.

3.5 Tensor Fields

The tensor field is the discretization of a tensor over a domain which is the description of physical parameters determined as part of the simulation processes. The field discretizes a tensor over the discretized domain. The tensor field component provides functions to obtain the information needed for error estimation and to support the transfer of solution fields during the simulation.

To be able to support a specific tensor defined over multiple discretized domains, and/or discretized in terms of multiple distributions, the field component has a set of field instances where a single field instance has a single set of distributions over a given discretized domain.

The fields API includes

1. a component-internal API: (*i*) field information queries, (*ii*) field instance information queries, (*iii*) field coordinate transformation, (*iv*) field reduction and modification

2. an API for interacting with other components: (*i*) association of field with the discretized geometric domain entities, (*ii*) association with quantities determined by model solution processes, (*iii*) relation to parameters, and

3. an API for interacting with other fields of the component: (*i*) solution transfer between field instances.

3.6 Scale-Linking Operators

The function of the scale-linking operators is to transform parameters between scales being addressed by different models. The parameters to be transformed are tensors that are typically defined as fields on the appropriate discretization. The definition of any scale linking operator is a function of

- the domains on the two scales and the form of domain interactions,
- the domain discretization used for the interacting fields,
- the distribution functions and dof used to represent the interacting fields over the discretized domains, and
- the functional operations associated with transforming the field information on the one scale to the other scale.

The methods used should allow the scale-linking operators to be defined at the highest level of problem definition with additional qualification as needed to account for specific forms of domain discretizations and of the field distributions used.

Its functioning API must include

1. a component-internal API: (i) definition of the linking operations,
2. an API for interacting with other components: (i) relationship to parts and domains and interactions, (ii) relationship to fields and parameters, and
3. an API for interacting with other instances of scaling linking operator component: (i) coupling interactions between scale linking operations.

4 Example Multi-Model Simulation Procedures

The examples of automated adaptive simulation procedures presented in this section employ prototype implementations of the functional components outlined Section 3. The first example is an automated adaptive single-scale procedure that is being used in industry. The second is an adaptive atomistic/continuum multiscale procedure currently under development.

4.1 Automated Adaptive Mesh-Based Simulation

A large number of codes are used for the solution of PDEs on a given fixed mesh. Although these codes are capable of providing results to the required levels of accuracy, the vast majority lack the ability to automatically control the mesh discretization errors through the application of adaptive methods. Using interoperable components discussed in Section 3 in conjunction with existing fixed-mesh finite element models and a mesh modification component [26], multiple adaptive analysis procedures have been built.

One such example was created for 3-D forming simulations in which the deformable parts undergo large plastic deformations that result in major changes in the analysis domain geometry. The meshes of the deforming parts typically need to be frequently modified to continue the analysis due to large element distortions, mesh discretization errors and/or geometric approximation errors. In these cases, it is necessary to replace the deformed mesh with an improved mesh that is consistent with the current configuration. Procedures using the two domain and field components are employed to determine a new mesh size field considering each of these factors which is provided to a local mesh modification [43] that creates the adapted mesh. The tensor field component is also used to transfer history-dependent field variables as each mesh modification is performed [43] so that the full set of information needed for the next set of analysis steps can be provided to the analysis model which is the commercial finite element code DEFORM-3D [16].

Figure 5 shows the setup, initial mesh, and final adapted meshes for a steering link manufacturing problem solved using this multi-model capability. A total stroke of 41.7mm is taken in the simulation. The initial work piece mesh consists of 28,885 elements. The simulation is completed with 20 mesh modification steps producing a final mesh with 102,249 elements.

< INSERT FIGURE 5 HERE >

4.2 Adaptive Atomistic/Continuum Adaptive Multiscale Simulation

A concurrent adaptive multiscale simulation capability is being developed for the modeling of fracture problems in metallic structures [10]. The key analysis engines for this multi-model application are non-linear finite element models for the continuum level and molecular statics models to address the discrete-level aspect of dislocation formation and growth. Part of the simulation viewpoint in this case is the indication of the sets of behaviors that can be associated with the parts that indicates that both linear- and non-linear continuum behavior can be considered and that atomistic regions can be superimposed at the locations of failure initiation, like crack tips. The equation parameters include the continuum material properties, loads and boundary conditions, and the atomistic potentials. The geometric domains include the full part geometry and atomistic overlays, including defect locations, for the specific locations where that are adaptively determined to be needed. The discretized representations of these two regions are a finite element mesh and atomistic positions taking account of the defects, respectively. The tensor fields include overall and local deformations and stresses on the continuum level, and atom positions and forces on the atomistic level. Since the atomistic and continuum levels overlap, the options for the scale-linking operators include the relating of local deformations and forces either though the common boundary or through the overlap region. In both cases, the atomistic deformations must be smoothed before being transferred to the continuum level and the discrete inter-atom forces must be transformed into stress like quantities to relate them to the continuum level stresses.

Figure 6 shows an example of adaptive atomic continuum simulations for the definition and growth of dislocations at a crack tip. In this case the cracked macro-domain was defined in a solid modeler and the finite elements were automatically generated. Based on a two-component error indication procedure, atomic lattice overlays are defined in the critical regions. As the defects form, the atomic fields automatically adjust.

< INSERT FIGURE 6 HERE >

5 Closing Remarks

The focus of this paper has been an examination of the process of performing adaptive multiscale simulation with the goal of defining an appropriate set of high-level abstractions that can support the construction of multi-model simulations taking advantage of established models that can effectively address specific aspects of these simulations. This abstraction process has led to the definition of the six functional components needed to support the transformation and transfer of information to the various models associated with these multi-model simulations. These ideas are demonstrated through two multi-model automated adaptive simulation examples building on initial prototypes of the six functional components.

References

[1] ABAQUS Inc. http://www.abaqus.com.

[2] Beall, M.W., and Shephard, M.S. 1997. A General Topology-Based Mesh Data Structure. *Int. J. Num. Meth. Engng* 40(9):1573-1596.

[3] Beall, M.W., Walsh, J., and Shephard, M.S. 2004., Accessing CAD Geometry for Mesh Generation. *Engineering with Computers* 20(3):210-221.

[4] Belytschko, T., and Xiao, S.P. 2003. Coupling methods for continuum model with molecular model. *Int. J. Multiscale Comput. Eng* 1:115-126.

[5] Beju, I., Soos, E., and Teodorescu 1983. *Euclidean Tensor Calculus with Applications*. Abacus Press.

[6] Bidarra, R., and Bronsvoort, W.F. 2000. Semantic feature modeling. *Computer-Aided Design* 32:201-225.

[7] Bronsvoort, W.F., and Jansen, F.W. 1993. Feature modeling and conversion - key concepts to concurrent engineering. *Computers in Industry* 21(1):61-86.

[8] Broughton, J.Q., Abraham, F.F., Berstein. N., and Kaxiras, E. 1999. Concurrent coupling of length scales: Methodology and application. to appear in *Phys. Review B* 60:2391-2403.

[9] Chand, K.K., Diachin, L.F., Li. X., Ollivier-Gooch, C., Seol, E.S., Shephard, M.S., Tautges, T. and Trease, H. 2006. Toward interoperable mesh, geometry and field components for PDE simulation development. *Engineering With Computers*

[10] Datta, D., Picu, R.C., and Shephard, M.S. 2004. Composite Grid Atomistic Continuum Method: An adaptive approach to bridge continuum with atomistic analysis. *Journal of Multiscale Computational Engineering* 2(3):401-420.

[11] E, W., and Enquist, B. 2002. The heterogeneous multi-scale methods. *Com. Math. Sci.* 1:87-132.

[12] Fish, J., and Yuan, Z. 2005. Multiscale Enrichment based on the Partition of Unity. *Int. J. Num. Meth. in Engng* 62(10):1341-1359.

[13] Fish, J. 2006. Discrete to Continuum Multiscale Bridging. to appear in *Multiscaling in Molecular and Continuum Mechanics*.

[14] Fish, J., Chen, W., and Nagai, G. 2002. Nonlocal dispersive model for wave propagation in heterogeneous media: One-Dimensional Case and Multi-Dimensional Case. *Int. J. Num. Meth. in Engng* 54(3):331-363.

[15] Fish, J., and Belsky, V. 1995. Multigrid method for a periodic heterogeneous medium. *Comp. Meth. Appl. Mech. Eng* 126:1-38.

[16] Fluhrer J 2004 *DEFORM-3D Version 5.0 User's Manual*. Scientific Forming Technologies Corporation.

[17] Frenkel, D., and Smit, B. 2002. *Understanding Molecular Simulations: From Algorithms to Applications*. 2^{nd} ed. Academic Press.

[18] Gravouil, A., and Combescure, A. 2001. Multi-time-step explicit method for nonlinear structural dynamics. *Int. J. Num. Meth. in Engng* 50:199-225.

[19] Hoffmann, C.M., and Joan-Arinyo, R. 1998. CAD and the product master model. *Computer-Aided Design* 30(11):905-918.

[20] Hoffmann, C.M., and Joan-Arinyo, R. 2000. Distributed maintenance of multiple project views. *Computer-Aided Design* 32:421-431.

[21] Hou, T. Y., and Wu, X. 1997. A multiscale finite element method for elliptic problems in composite materials and porous media. *J. Comput. Phys.* 134:169-189.

[22] Hughes, T.J.R., Mazzei, L., and Jansen, K.E. 2000. Large-eddy simulation and the variational multiscale method. *Computing and Visualization in Science* 3:47-59.

[23] Hull, D., and Bacon, D.J. Eds 1975, "Introduction to Dislocations", Butter worth-Heinemann.

[24] Kevrekidis, I.G., et al 2003. Equation-free coarse-grained multiscale computation: enabling microscopic simulators to perform system-level tasks. *Comm. Math. Sciences* 1(4): 715-762.

[25] Knap, J., and Ortiz, M. 2001. An analysis of the quasicontinuum method. *J. Mechanics and Physics of Solids* 49:1899-1923.

[26] Li, X., Shephard, M.S., and Beall, M.W. 2002. Accounting for curved domains in mesh adaptation. *Int. J. Num. Meth. in Engng* 58:246-276.

[27] Li, X., Shephard, M.S., and Beall, M.W. 2005. Anisotropic mesh adaptation by mesh modifications. *Comp. Meth. Appl. Mech. Engng.* 194(48-49):4915-4950.

[28] Luo, X., Shephard, M.S., Remacle, J.-F., O'Bara, R.M., Beall, M.W., Szab, B.A., and Actis, R. 2002. p-Version Mesh Generation Issues. 11th *International Meshing Roundtable* 343-354.

[29] Mavrantzas, V.G., Boone, T.D., Zervopoulou, E., and Theodorou, D.N. 1999. End-Bridging Monte Carlo: An Ultrafast Algorithm for Atomistic Simulation of Condensed Phases of Long Polymer Chains. *Macromolecules* 32:5072-5096.

[30] Miller. R.E., and Tadmor, E.B. 2002. The Quasicontinuum Method: Overview, Applications and Current Directions. *J. of Computer-Aided Materials Design* 9:203-209.

[31] Noort, A., Hoek, G.F.M., and Bronsvoort, W.F. 2002. Integrated part and assembly modeling. *Computer-Aided Design* 34:899-912.

[32] O'Bara, R.M., Beall, M.W., and Shephard, M.S. 2002. Attribute Management System for Engineering Analysis, *Engineering with Computers* 18(4):339-351.

[33] Padding, J.T., and Briels, W.J. 2002. Time and length scales of polymer melts studied by coarse-grained molecular dynamics simulations. *J. Chem. Phys.* 117(2):925-943.

[34] Pandofi, A., and Ortiz, M. 2002. An efficient procedure for fragmentation simulations. *Engineering With Computers* 18(2):148-159.

[35] Parasolid Inc. http://www.ugs.com/products/open/parasolid.

[36] Shephard, M.S., and Georges, M.K. 1992. Reliability of Automatic 3-D Mesh Generation. *Comp. Meth. Appl. Mech. and Engng.* 101:443-462.

[37] Shephard, M.S. 1985. Finite element modeling within an integrated geometric modeling environment: Part II - Attribute specification, domain differences, and indirect element types. *Engineering with Computers* 1:72-85.

[38] Simmetrix Inc., http://www.simmetrix.com/, 10 Halfmoon Executive Park Drive, Clifton Park, NY 12065.

[39] Shephard, M.S., Beall, M.W., O'Bara, R.M., and Webster, B.E. 2004. Toward simulation-based design. *Finite Elements in Analysis and Design* 40:1575-1598.

[40] Shephard, M.S., Flaherty, J.E., Jansen, K.E., Li, X., Luo, X.-J., Chevaugeon, N., Remacle, J.-F., Beall, M.W., and O'Bara, R.M. 2005. Adaptive

mesh generation for curved domains. *J. for Applied Numerical Mathematics* 50(2-3):251-271.

[41] Spatial Inc. http://www.spatial.com/components/acis.

[42] TSTT Software, http://tstt-scidac.org/software/software.html.

[43] Wan, J., Kocak, S, Shephard, M.S., and Mika, D. 2005. Automated adaptive 3-D forming simulations. to be appear *Engineering with Computers.*

[44] Weiler, K.J. 1998. The radial-edge structure: A topological representation for non-manifold geometric boundary representations. In *Geometric modeling for CAD applications*, 3-36. North Holland.

Figure 1. Multi-model hierarchy used in the design of a composite material system.

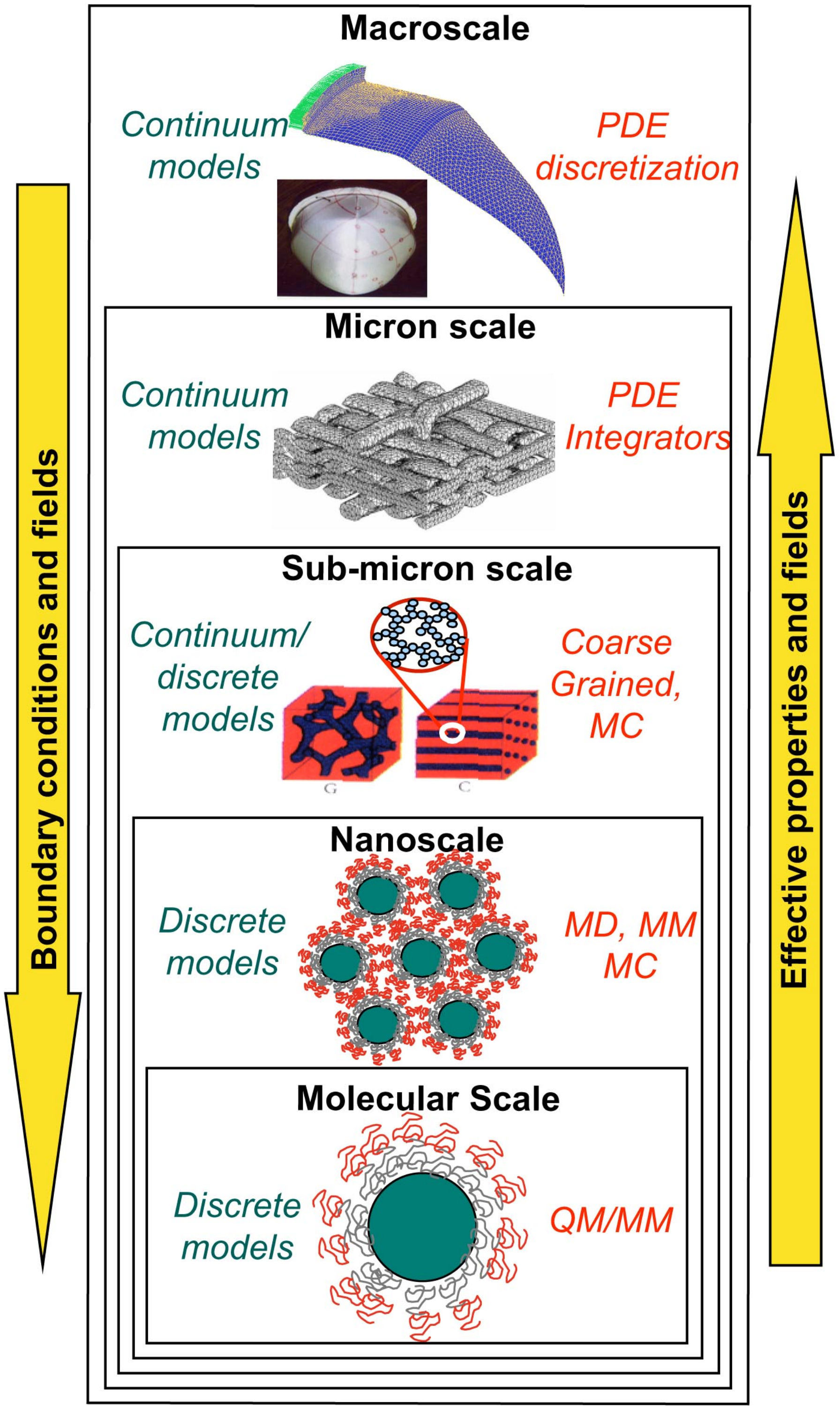
Figure 2. Multi-model hierarchy needed for a drug delivery system.

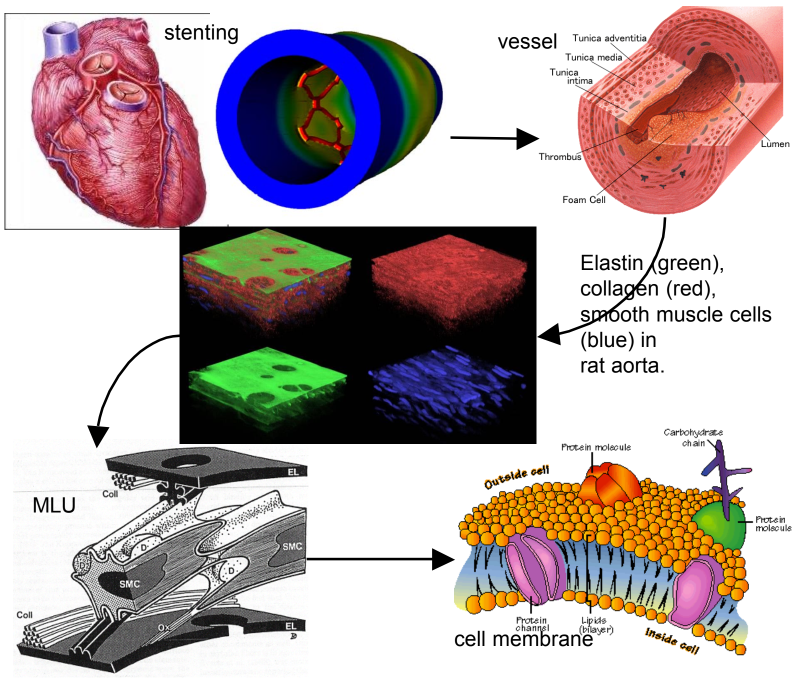
Figure 3. Example of a non-manifold model.

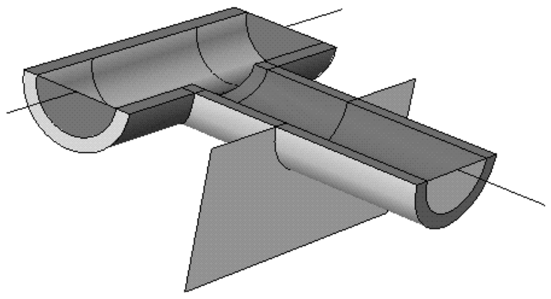
Figure 4. Interactions between components.

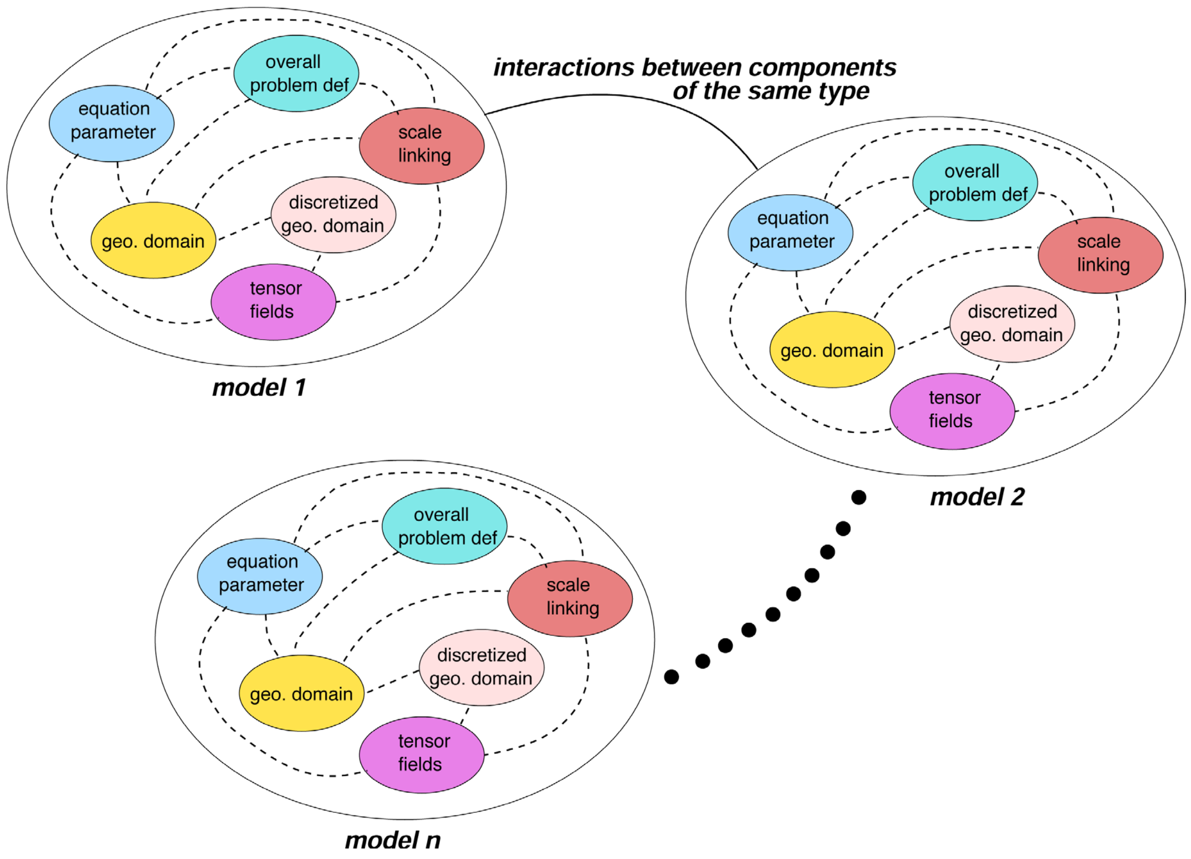
Figure 5. Adaptive forming simulation example.

Figure 6. Adaptive molecular/continuum multiscale simulation.

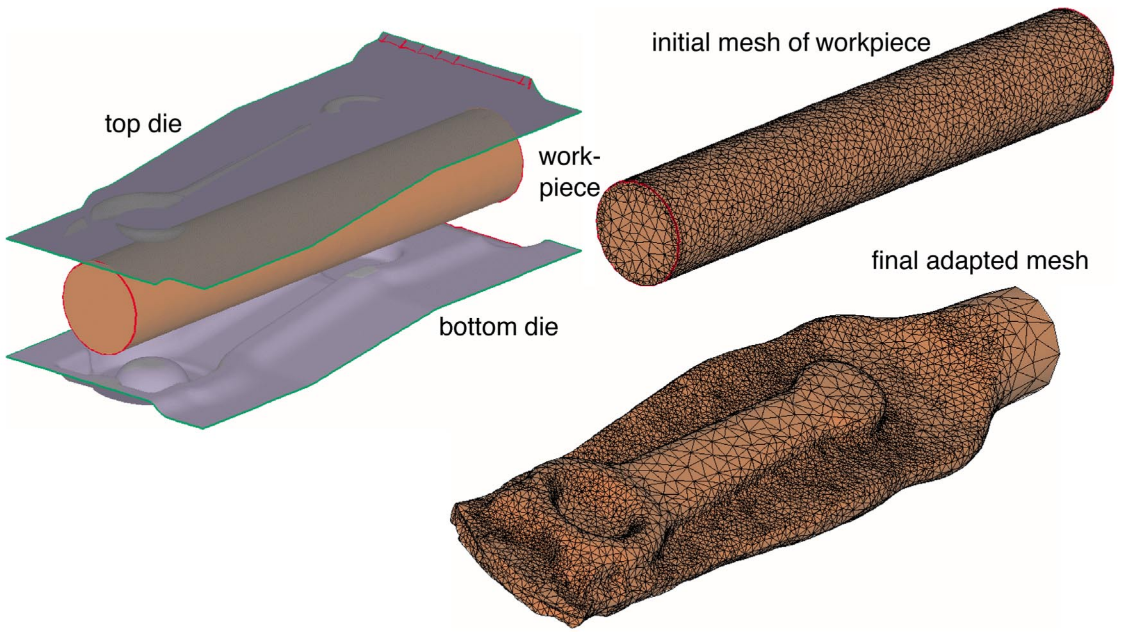

Macroscale
Continuum models
PDE discretization
Micron scale
Continuum models
PDE Integrators
Sub-micron scale
Continuum/ discrete models
Coarse Grained, MC
G
C
Nanoscale
Discrete models
MD, MM MC
Molecular Scale
Discrete models
QM/MM
Boundary conditions and fields
Effective properties and fields

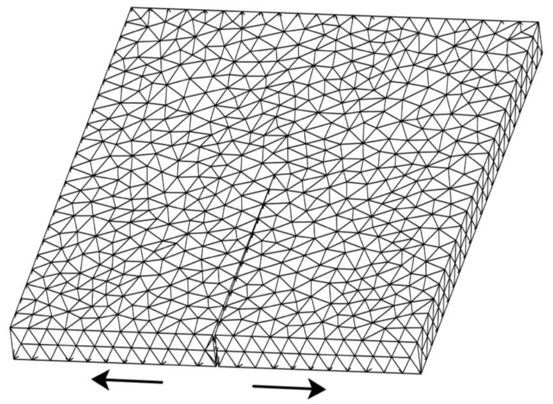

stenting
vessel
Tunica adventitia
Tunica media
Tunica intima
Lumen
Thrombus
Foam Cell
Elastin (green), collagen (red), smooth muscle cells (blue) in rat aorta.
MLU
Coll
EL
D
SMC
Ox
Protein molecule
Carbohydrate chain
Outside cell
Protein molecule
Protein channel
Lipids (bilayer)
Inside cell
cell membrane



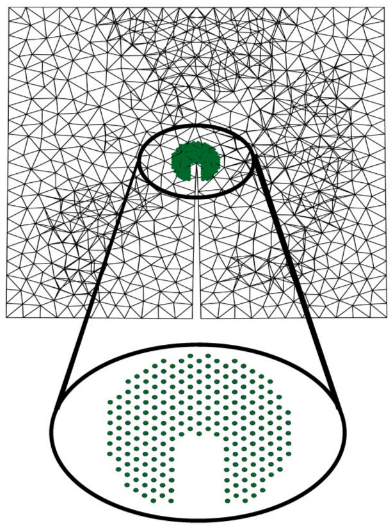


interactions between components of the same type
overall problem def
equation parameter
scale linking
geo. domain
discretized geo. domain
tensor fields
model 1
overall problem def
equation parameter
scale linking
geo. domain
discretized geo. domain
tensor fields
model 2
overall problem def
equation parameter
scale linking
geo. domain
discretized geo. domain
tensor fields
model n

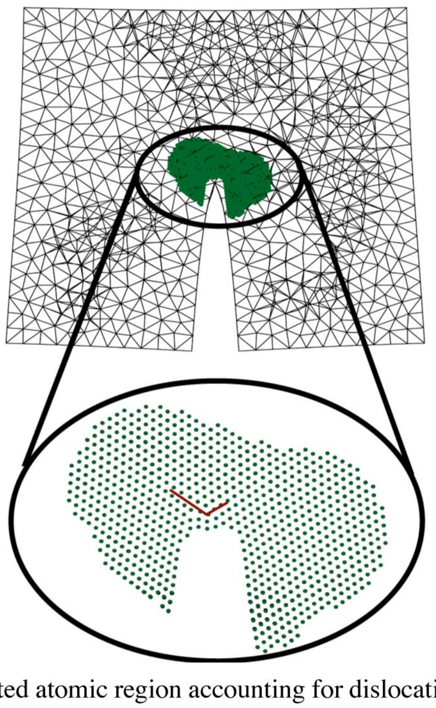

top die
work-
piece
bottom die
initial mesh of workpiece
final adapted mesh



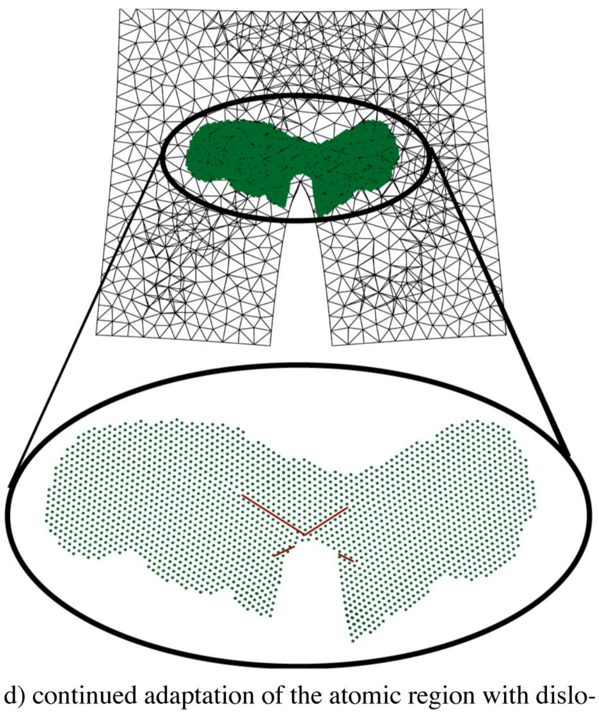
a)3-D domain with a crack and macroscale mesh



b) adaptively superimposed atomic region (in green)



c) adapted atomic region accounting for dislocation formation and growth



d) continued adaptation of the atomic region with dislocation growth