Toward a Multi-Model Hierarchy to Support Multiscale Simulations

Mark S. Shephard E. Seegyoung Seol

Benjamin FrantzDale

Scientific Computation Research Center

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

June 20, 2006

1 Introduction

There is a long history of developing 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 scale to define nanoscale building blocks that will be used to construct mesoscale features that may be combined into micron-scale weaves that could be used in the manufacturing of complete parts (figure 1). Such capabilities are clearly central to the development of nanoelectronics devices and future biomedical device design, as well as many of other future products.

<INSERT FIGURE 1 HERE>

As an example of the potential impact of multiscale simulation on biomedical device design, consider a drug-eluting stent (figure 2). Drug-eluting stents are hybrid device–drug medical products that release therapeutic drugs and provide a scaffold to maintain arterial lumen size after angioplasty. The design of these devices requires consideration of the mechanical function of holding open a diseased artery, and of the pharmacological function of delivering the appropriate drug in the appropriate concentration for the requisite length of time to prevent in-stent restenosis. The mechanical simulations involved with device deployment includes continuum-scale models of the stent and blood vessel, which employ complex material models. The complex nature of the blood vessel requires the application of multiscale methods to determine those material

models. Consideration of drug delivery from the stent coating requires a model that includes continuum-level modeling of blood flow coupled with molecular-level diffusion and transport of drug molecules, which needs to be coupled to cell- and molecular-level models of drug diffusion into blood vessels through cell membranes. Similar models and methods are central to many other applications. For example, similar models and model coupling is needed in the consideration of new automotive skins made of nano-reinforced materials in which the material interfaces are strong at strain rates consistent with normal use, while at high strain rates demonstrating substantial local damage, leading to high stiffness so that little dents are avoided, but providing high energy absorption under impact loading to keep passengers safe.

<INSERT FIGURE 2 HERE>

There are many available models to solve various single-scale simulation problems, but while the development of multiscale methods is an active research area, there has been limited attention paid to the development of general multiscale modeling techniques. One procedure that does address the complex issue of bridging from atomistic to continuum physics, including the ability to adaptively control the model selection over the domain, is the quasicontinuum method (Knap and Ortiz 2001; Miller and Tadmor 2002). Since this procedure is based on a single method to define and link the physical scales, its development has not needed to address the inclusion of flexible methods or the insertion of alternative models and scale-linking methods. Other efforts have limited the range of models included to similar models such as the OCTA soft-

ware, which includes four discrete mesoscale models (OCTA 2006). Considering the thousands of person-years of effort that has gone into the development of the existing single-scale models that operate at each of the physical scales needed, the effective development of multiscale simulations will be greatly facilitated by the development of methods that can easily integrate and use existing and developing single-scale models. This chapter outlines the overall structure of a component-based multi-model approach in which each model uses clearly defined interfaces and functionality for sharing information. Since these methods are early in the development phase, this chapter provides one view of how this complex problem could be addressed with the goal of opening a constructive dialog between the software engineers developing multimodel methods and the computational engineers developing multiscale methods.

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

2 Functional and Information Hierarchies in Multiscale Simulation

In abstracting multiscale simulation processes, one must consider the hierarchy of transformations required to go from a description of physical behavior to a set of mathematical and computational models capable of simulating the desired behaviors. The highest level in the hierarchy is an overall problem definition related to mathematical descriptions used to describe the behavior, typically-coupled, at various scales, including equations relating parameters between the scales. The other two levels in the hierarchy are the discretizations of the mathematical models and the numerical algorithms used to solve the discretized mathematical models. A key to abstracting this process is to qualify 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 three groups:

Mathematical models: The description of the mathematical equations used as a description of the physical behavior on the various scales and mathematical equations that relate behaviors between scales.

Domains: The description of the domains over which the various mathematical equations apply. In the case of multiscale analysis, this includes appropriate definitions at each relevant scale of space and time, and the spatial and temporal interactions between them.

Physical parameters: The description of the various physical parameters used in the mathematical equations, defined over the appropriate domains as required to qualify the current instances of the governing equations to be solved.

The ability to properly support component-based 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 solution methods interaction modes can be supported.

2.1 Mathematical Physics Description Transformations and Interactions

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 equations 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. To make this more concrete, consider the two most common forms of equations encountered in multiscale analysis: partial differential equations (PDEs), which are defined at various continuum scales, and molecular dynamics (MD), which is based on interatomic potentials that define the interactions between molecules and atoms at the small scales for which continuum equations over the domain are not applicable.

2.1.1 PDEs

PDEs may be written in terms of multiple sets of dependent variables where each set can contain tensors of various orders that vary over the space–time domain. For the purposes of this discussion, consider the PDE

$$\mathcal{D}^m(u,\sigma) - f = 0 \quad \text{on} \quad \Omega \tag{1}$$

subject to boundary conditions

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

where

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

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

σ represents one or more dependent scalar 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.

Computerized models of the PDEs typically use mesh-based methods in which a two-part discretization process is used to transform the mathematical model into numerical systems which are solved. The first part of the discretization is the decomposition of the space–time domain into a set of mesh entities with simple shapes in space and time. The second part of the discretization is to discretize the shapes of the functions. A set of basis functions related to a "weak form" of the governing equation, and/or to difference relations for the differential operators, are used to discretize the dependent variables over the individual mesh entities in terms of a set of to-be-determined parameters, called degrees of freedom (dofs). The dofs can always be associated with a single mesh entity whereas the distribution functions (basis functions or difference relations) are associated with one or more mesh entities. In the case that 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 among the mesh entities, the dofs, and the distributions are:

Finite difference based on a vertex stencil, in which the dofs are typically values of the dependent variables at vertices of a mesh and the distribution functions are difference stencils defined in terms of vertex values for mesh vertices for the appropriate set of topologically adjacent mesh entities.

Finite volume methods are constructed in terms of distribution function written over individual mesh entities. In most cases the field being defined is discontinuous between elements (that is C^{-1}). Therefore, dofs are not shared between neighboring mesh entities. The coupling of the dofs from different mesh entities is through operators acting over boundary entities between neighboring mesh entities.

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 dofs associated with the bounding mesh entities common to the neighboring elements.

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

$$k^c d^c = f^c \tag{3}$$

where k^c is the discretized matrix for contributor c that multiplies the vector of dofs 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 dofs, d^c, with the assembled set of global dofs, d.

2.1.2 Molecular Dynamics

In molecular dynamics (MD), the mathematical model is a potential function describing the forces among interacting atoms and which depends on the relative position of the atoms (Frenkel and Smit 2002). An a common potential function is the Lennard–Jones potential which approximates the force between two atoms as

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

where σ and ϵ are Lennard–Jones parameters for a given material and r is the inter-atomic distance. The parameters in potential equations may be developed empirically or based on simulations performed on a finer *ab initio* scale using an electron model. Due to the large number of atoms required to 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, or fixed ("Dirichlet"). The direct outputs of an MD simulation, atom trajectories and forces on the atoms, are typically not of specific interest, but rather are needed to determine the meaningful higher-scale parameters of interest. The extraction of those higher-scale parameters is often done by taking 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 but which are actually the results of another model. In some cases, the coupling simply requires solving the models in a given order so that parameters are available when required. In other cases, parameters are shared in both directions, 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, in which the flow field is influenced by the geometry of the structure over which it flows and the geometry of the structure is a function of the forces on it caused by the flow field. The issues associated with the transfer of parameters between models depend on the portions of the domain over which the interactions occur and on how those portions have 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 at the different scales, for the models used to couple information between the scales, and for 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" (Fish 2006). With information-passing methods, fine scales are modeled and their gross response is infused into the coarse scale; the influences of coarse-scale fields on the fine scales are taken into account as boundary conditions and/or forcing functions on the fine scale. With concurrent bridging, the fine and coarse scales are simultaneously resolved. For nonlinear problems, the models at different scales are coupled in both directions

and information continuously flows between the scales.

In many information-passing techniques, the fine-scale model is a representative unit cell subject to appropriate boundary conditions and information passed to the larger scale is considered to be at a point on the larger scale. In concurrent techniques, the fine-scale model acts over some small finite portion of the coarse-scale domain 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 at each scale, the relationships of parameters between scales is usually not direct and care must be taken to define the appropriate operations to relate them. In some cases, these operations act as filters to remove information (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, operations are needed to relate quantities with different forms of definition (e.g., atomic-scale forces to continuum stresses) or to define terms not defined at a given scale (e.g., defining continuum-level temperature in terms of atomic scale motions).

The complication of properly relating information between scales has led to the active development of methods for scale linking and to computer implementation of these methods. Representative information-passing methods include multiple-scale asymptotic techniques (Fish et al. 2002), variational multiscale

methods (Hughes et al. 2000), heterogeneous multiscale methods (E and Enquist 2002), multiscale enrichment schemes based on partition of unity (Fish and Yuan 2005), discontinuous Galerkin discretizations (Hou and Wu 1997), and equation-free methods (Kevrekidis et al. 2003). Spatially-concurrent schemes are based on either multilevel (Fish and Belsky 1995) or domain-bridging methods (Belytschko and Xiao 2003; Broughton et al. 1999), while concurrent schemes in the time domain are typically based on multistep methods (Gravouil and Combescure 2001).

2.2 Domain Definitions, Transformations and Interactions

The domains considered here are space–time domains. Time is a linear progression that runs from an initial time to a final time and is typically discretized using a well-accepted set of methods based on time increments. 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 domain representation must support the transformation of an original domain definition into representations that can support a discretization of the governing equations over the domain. For example, the original definition of a domain may be a feature-based model of a domain over which a mesh-based simulation is to be performed. The process of creating the mesh in this case requires the transformation of the feature model into a non-manifold geometric model upon which an automatic mesh generation procedure can be applied to generate the desired mesh (Shephard, Beall,

O'Bara, and Webster 2004). In addition, the transformations needed to construct the required domain representations; it is necessary to maintain the relationship between the entities in each of the representations.

- The domain representation must 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 to be used by the models. For example, the ability to map the components of a tensor with a given distribution onto a model entity, such as a surface of the geometric domain.

- The domain representation must support the ability to address any domain interrogation required during the execution of models involved with the simulation. Most of these interrogations can be limited to pointwise evaluations (e.g., determine the normal vector at a given point on a surface).

- The domain representation must support geometric interactions between related domains used in a multiscale simulation. For example, in a concurrent multiscale model, to determine the mesh entities in the continuum domain which overlap with the atomic region.

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

An important consideration in selecting a boundary representation is its ability to represent the classes of domain needed. In the most general case, domains can be general combinations of 0-, 1-, 2-, and 3-D entities where lower-order entities are not required to bound higher-order entities. Figure 3 shows a typical analysis domain of this type which would be appropriate for structural analysis. The boundary representations that can fully and properly represent such geometric domains are referred to as a non-manifold boundary representations (Weiler 1998).

In addition to the topological entities and associated shape information, geometric-modeling systems maintain numerical tolerance information on how

well the entities fit together. The algorithms and methods within a geometric modeling system are able to use such 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 of, or improper use of, the tolerance information (Beall, Walsh, and Shephard 2004).)

< INSERT FIGURE 3 HERE >

Abstracting topology is an effective way to allow for the development of functional interfaces to boundary-based modelers that are independent of specific shape information. The developers of CAD systems have recognized the possibility of supporting geometry-based applications through general application program interfaces (APIs) where functions that provide entity adjacencies, calculate geometric information such as surface normals, etc. are keyed to topological entities. This has lead to the development of geometric modeling kernels such as ACIS (Spatial Inc.) and Parasolid (Parasolid, Inc.) which have been successfully used to develop automated finite element modeling processes (Shephard et al. 2005; Wan et al. 2005) and automatic mesh generators (Beall et al. 2004).

In the application of generalized numerical analysis processes, a meshed approximation must be created from a geometric domain. To support the full set of operations needed for reliable multiscale analysis, a mesh must maintain an association with its continuum-domain representation and with the distribution functions and number of dofs used in discretizing the PDEs (see

section 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 (Beall and Shephard 1997).

A key component of supporting mesh-based simulation is the association of a mesh to its geometric model (Beall and Shephard 1997; Shephard and Georges 1992), which indicates the mesh entities that represent particular model entities. This association is used for operations such as ensuring the mesh entities on the boundary of a model are properly curved when needed, associating boundary conditions defined at the model entity level with the appropriate mesh entities, etc. This association can be defined as follows:

Classification: The unique association of the i^{th} mesh topological entity (with dimension d_i), $M_i^{d_i}$, to a topological entity of the geometric model of dimension d_j, $G_j^{d_j}$, on which it lies, where $d_i \leq d_j$. This 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 defines the reverse classification information for that model entity. Reverse classification is denoted as

$$\text{RC}(G_j^d) = \{M_i^d \mid M_i^d \sqsubset G_j^d\} \tag{5}$$

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 Bézier curves and surface definitions with mesh edges and faces for use in high-order curved finite elements (Luo et al. 2002). The mesh classification can be used to obtain other needed geometric information such as the coordinates of a new mesh vertex formed by splitting a mesh edge classified on a model face.

2.2.2 Discrete Domains

The domain definition for the discrete domains 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 which provide information defining the positions of atoms. 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 can be applied which account for knowledge of the material system. There can be defects in the crystal systems (Hull and Bacon 1975). With additional information about these defects and the total number of atoms, coordinates, and velocities of the atoms can have an initial adjustment applied to them. In the case of polyms, an atom's position must be defined by its position along its 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. Methods like those just outlined, which can take a compact definition of discrete domains and produce a proper set of atomistic positions, are required. In some cases these methods will be purely geometric while in others will can require the execution of a full atomistic relaxation model.

One approach to bridging scales is to interpolate the behavior of a large set of atoms in terms of a small subset of them. One such approach, well-suited to lattice structures, is the quasi-continuum method in which the position of atoms over simple shapes such as triangles and tetrahedra is described to vary linearly between known atom positions (Knap and Ortiz 2001; Miller and Tadmor 2002). In the case of polymer chains, atoms along a chain can be represented by a small number of "beads" placed along the chain (Mavrantzas et al. 1999; Padding

and Briels 2002).

2.2.3 Interactions of Domains

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

Disjoint domains, which share information across a common boundary.

Overlapping domains, which have portions of the overall domain represented at more than one scale and the information is shared through the overlapped region.

Telescoping domains, which represent microstructure by many small-scale domains, which have essentially zero size with respect to the higher-scale domain. Thus, each 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 (Beju et al. 1983) 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 possible. For example, in the case of solving a 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 section 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 given tensor defined over multiple meshes and/or discretized in terms of multiple distributions can be handled by supporting multiple field instances.

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

In the design of a multi-model system to support multiscale simulations, it is important to determine the information required by the models and the transformations to be applied to provide the information in the needed form. Within the multi-model multiscale simulation environment, functional APIs are defined to support the various classes of information transformations needed. Employing the APIs provided by components makes it straightforward to combine various single-scale models to construct multi-models for multiscale simulations. Each of the various models interact with other models only through the component's API. For example, in a concurrent model, part of the scale-linking component would be a function linking atomistic to continuum using statistical averaging of atomic displacements on the boundary of the atomistic region thereby providing

boundary deformations to the continuum model.

A key goal of this design is to build multiscale simulation procedures by using adaptive solution strategies to control existing time-tested single-scale models thereby ensuring the reliability of simulation in terms of providing predictions of the desired parameters to the required degree of accuracy. The only way to provide this reliability is to explicitly consider the approximation errors that can arise within each step executed by a model or transformation performed by a component. Since many of these errors cannot be controlled through *a priori* means, it is necessary to support adaptive feedback processes that use *a posteriori* information to control the execution of each model step and transformation.

A number of the models needed to perform specific simulation steps are well-established and should be used. Two examples are generalized fixed-mesh continuum PDE 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 widely-used software operates only though input and output files. In that case, the components will be a facade, crating the input files and interacting with the output files. Although this case does not take full advantage of the components, advantages gained are the easy substitution of other models, including ones that can more-directly interact with the components.

Some programs support the addition of user-defined functionality. For ex-

ample, ABAQUS (ABAQUS Inc.) supports user-defined material models and user-defined finite elements. Although limited, these two features facilitate the majority of the functionality needed for ABAQUS to be an effective model in a multiscale simulation environment.

Another area in which mature models exist to support multiscale modeling is the definition of geometric domains of 3-D parts using boundary representations. Most existing systems provide a functional API (Parasolid, Inc.; Spatial Inc.), which is ideal for creating a component for a multiscale simulation. These geometric-modeling APIs provide the capabilities needed to represent continuum-level domains in multiscale simulations. There are also many existing programs which can generate mesh-level discretizations of geometric domains, the interfaces of which range from file- to API-based (Beall et al. 2004). API-based interfaces have been used in the development of adaptive mesh modification procedures (Li et al. 2002) and complete adaptive PDE multi-model simulations (Shephard et al. 2005; Wan et al. 2005), and are well-suited for the needs of multiscale simulation.

In designing a multi-model system to support multiscale simulation, we must identify appropriate levels of abstraction to support the flow of information between models such that information can be provided to procedures which execute any required transformations. The components defined to support multi-model multiscale simulations are:

1. problem definition,

2. equation parameters,

3. geometric domains,

4. discretized geometric domains,

5. tensor fields, and

6. scale-linking operations.

A subset of similar functional components are being defined 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 (Chand et al. 2006; TSTT Software 2006) and commercial products (Simmetrix Inc. 2006).

< INSERT FIGURE 4 HERE >

Figure 4 illustrates the structure of a multiscale multi-model in which five functional components are used by the single-scale models in the multi-model of a given multiscale simulation. Additional scales can be added by adding another scale-linking component and model to either side of the diagram. Each instance of a component utilizes other components in the same scale to do its job. Information flow is indicated by the arrows. Furthermore, components will only share information through scale linking with the component of the same type in the other model. Based on this, the functions within a component need to consider the following three modes of interaction:

1. providing and modifying component-internal data (procedures which don't involve interactions with other components),

2. providing information between components within a model, and

3. providing information between like components for different models.

3.1 Problem Definition

To consider multiscale simulations, we need an overall problem definition. A problem definition must include all of the relevant physical parts, must define the relationships between the parts, and must facilitate the creation of alternate representations of the parts. In addition, a problem definition must provide any other information needed to construct and execute the desired solution process. A problem definition must also allow for the creation of viewpoint-specific interfaces as needed by the solution strategy and simulation models to be used (for example, considering an atomic region as though it is a continuum).

Representations, including simulation viewpoints, are under development (Shephard et al. 2004). These representations can be defined in terms of graph structures similar to those used to define assembly and feature models in CAD systems (Bidarra and Bronsvoort 2000; Hoffmann and Joan-Arinyo 1998), with the extensions to support hierarchal decompositions and multiple viewpoints (Bronsvoort and Jansen 1993; Hoffmann and Joan-Arinyo 1998; Hoffmann and Joan-Arinyo 2000; Noort et al. 2002). In the single-scale case, a problem-definition component would then support the following interaction modes:

Component-internal: part definitions, relationships of parts, and mathematical equations governing part behavior.

Inter-component: relationships of parts to domains, relationships of parts to parameters, relationships of parts to model functions, and model-level

simulation strategy information.

Interacting with like components of other models: viewpoint construction rules, relation of mathematical equations on related parts in the different models, and multiscale simulation strategy information.

In the multiscale case, the overall problem definition is inherently multiscale. As such, it will depend on a scale-linking component, as well as single-scale models which compose it, complete with their own subproblem definitions.

3.2 Equation Parameters

The parameters in the mathematical equations representing physical quantities are, in general, 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 a specific set of these tensors associated with the mathematical equations of that model, the parameters can be stated abstractly in terms of the highest-level problem-definition entities. For example, a boundary heat flux of $1\,\mathrm{kW/m^2}$ defined on surface 1 of mesh element 2 might be abstractly stated as the same heat flux applied to the bottom surface of a heat sink. This boundary conditions on the heat sink would be mapped onto a corresponding concrete solid model, and in turn onto a mesh. By defining conditions on the highest-possible level, it becomes easy to change the mesh, or even the solid model itself, without having to start from scratch. Generalized methods to define and manipulate such a structure of models have been defined (O'Bara et al. 2002; Shephard 1985).

The equation parameters component must support:

Component-internal: parameter information queries, parameter instance information queries, parameter coordinate transformations, and parameter reduction and modification functions.

Inter-component: relation to problem parts and geometric domains, relation to model solution process, and relation to fields.

Interacting with like components of other models: dependencies between parameters for different models.

3.3 Geometric Domain

A geometric-domain component is a functional unit to describe a multiscale simulation domain at a particular scale (e.g., continuum domain or atomistic domain). Within a multiscale model, a geometric domain supports geometric interactions between other geometric domains.

Consider, first, a continuum geometric domain defined in a CAD modeler in terms of a boundary representation. This component must support:

Component-internal: topological entity information, shape information, geometric-model tolerance information, and geometric model modification.

Inter-component: association with parts in the problem definition, association of equation parameters with the geometric domain (and, through that, association with domain discretizations—meshes), and association with scale linking.

Interacting with like components of other models: geometric interactions relating domains on different scales through boundaries and/or overlaps.

Atomic-scale models must support:

Component-internal: the definition of atom layouts, and the geometric relationships among atoms.

Inter-component: obtain potentials and provide forces.

Interacting with like components of other models: placement of the domain with respect to larger-scale domains, and geometric interactions relating domains of different scales through boundaries and/or overlaps.

3.4 Discretized Geometric Domains

A discretized geometric-domain component is a piecewise-geometric approximation of a corresponding geometric-domain component in terms of a mesh or idealized atomistic layout. This component must support:

Component-internal: topological entity queries for meshes, geometric shape of mesh entities, and geometric queries such as position of atoms and distance between atoms or mesh entities.

Inter-component: mesh Jacobian information, association with the geometric domain, and association with fields.

Interacting with like components of other models: mesh-to-mesh interaction, mesh-to-atomistic interactions, and discrete-to-atomistic interactions.

3.5 Tensor Fields

A tensor-field component is a discretization of a tensor field over a discretized domain. To support a tensor field defined over multiple discretized domains, the tensor-field component must support a collection of field instances for a single field where one field instance is defined over each of the discretized domains. The tensor field-component must support:

Component-internal: field information queries, field coordinate transformation, and field reduction and modification.

Inter-components: association of field with the discretized geometric domain entities, association with quantities determined by model solution processes, and relation to parameters.

Interacting with like components of other models: field transfer between field instances, and field transformation or modification to meet the model needs.

3.6 Scale-Linking Operators

Scale-linking operators exist to transform parameters among different single-scale models. A scale-linking operator is defined in terms of

- the domains at each scale and the form of the domain interactions,
- the domain discretization used for the interacting fields,

- the distribution functions and number of dofs 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 scale-linking operations to be defined at the highest level of problem definition, with additional qualification as needed to account for specific forms of domain discretizations and field distributions used. As such it must support:

Component-internal: the definition of the linking operations.

Inter-component: the relationship to parts and domains, and to other fields and/or parameters.

Interacting with models: using single-scale components' interfaces to transfer information between scales.

By combining components as described, and as shown in figure 4, each component will have minimal dependence on the rest of the system. This reduces software complexity and will allow the components to be easily interchanged.

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 in section 3. The first example is an automated adaptive single-scale pro-

cedure that is currently used in industry. The second is an adaptive atomistic–continuum multiscale procedure under development.

4.1 Automated Adaptive Mesh-Based Simulation

Many programs are used for the solution of PDEs on a given fixed mesh. Although these they are capable of providing results to required levels of accuracy, the vast majority lack the ability to automatically control mesh discretization errors through adaptive methods. Using the interoperable components discussed in section 3 in conjunction with existing fixed-mesh finite element models and a mesh-modification component (Li et al. 2002), 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 need to be frequently modified to continue the analysis due to significant 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, which is provided to a local mesh modification procedure (Wan et al. 2005) which creates an adapted mesh. A tensor-field component is also used to transfer history-dependent field variables as each mesh modification is performed (Wan et al. 2005) so that the full set of information needed for the next set of analysis

steps can be provided to the analysis model, the commercial finite element program DEFORM-3D (Fluhrer 2004).

Figure 5 shows the setup, initial mesh, and final adapted meshes for a steering-link manufacturing problem solved using this multi-model capability. This simulation shows a total stroke of 41.7 mm. The initial mesh of the work-piece consists of 28,885 elements. The simulation was 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

Concurrent adaptive multiscale simulation capabilities are being developed for modeling fracture in metallic structures (Datta et al. 2004). 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 atomistic aspects of dislocation formation and growth. Part of the simulation viewpoint in this case is the indication of the set of behaviors that can be associated with the parts which indicate that both linear- and non-linear continuum behavior can be considered, and that atomistic regions can be superimposed at locations of dislocation formation such as crack tips. The equation parameters include the continuum material properties, loads and boundary conditions, and the atomistic potentials. The geometric domain includes the full part geometry and atomistic overlays, including defect locations for the locations that are

adaptively determined to require an atomistic overlay. The computational representation of these two regions includes a finite element mesh and atomistic positions, taking account of the defects. The tensor fields include overall and local deformations as well as and stresses at the continuum level, and atom positions and forces on the atomistic level. Since the atomistic and continuum levels overlap, the options for a scale-linking operator include relating local deformations and forces either though a common boundary or through an 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 so they can be related to continuum level stresses.

Figure 6 shows an example of an adaptive atomic continuum simulation 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 generated automatically. Atomic overlay regions are defined in the critical areas based on an error indicator; as defects form, the atomic domains 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 components to support the construction of multi-model simulations taking advantage of established models that can effectively address specific aspects of these simulations. Six functional components have been defined which support the transformation and transfer of information to the various models used by these multi-model simulations. The application of these components has been demonstrated through two multi-model automated adaptive simulation examples building on an initial prototype of six functional components.

References

ABAQUS Inc. (2006). http://www.abaqus.com.

Beall, M. and M. Shephard (1997). A general topology-based mesh data structure. *International Journal for Numerical Methods in Engineering 40*(9), 1573–1596.

Beall, M., J. Walsh, and M. Shephard (2004). Accessing cad geometry for mesh generation. *Engineering with Computers 20*(3), 210–221.

Beju, I., E. Soos, and Teodorescu (1983). *Euclidean Tensor Calculus with Applications*. Abacus Press.

Belytschko, T. and S. Xiao (2003). Coupling methods for continuum model with molecular model. *International Journal for Numerical Methods in Engineering* 1, 115–126.

Bidarra, R. and W. Bronsvoort (2000). Semantic feature modeling. *Computer-Aided Design* 32, 201–225.

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

Broughton, J., F. Abraham, N. Berstein, and E. Kaxiras (1999). Concurrent coupling of length scales: Methodology and application. *Physical Review B* 60, 2391–2403.

Chand, K., L. Diachin, X. Li, C. Ollivier-Gooch, E. Seol, M. Shephard, T. Tautges, and H. Trease (2006). Toward interoperable mesh, geometry and field components for pde simulation development. *Engineering With Computers*.

Datta, D., R. Picu, and M. Shephard (2004). Composite grid atomistic continuum method: An adaptive approach to bridge continuum with atomistic analysis. *Journal of Multiscale Computational Engineering 2*(3), 401–420.

E, W. and B. Enquist (2002). The heterogeneous multi-scale method. *Communications in Mathematical Sciences* 1, 87–132.

Fish, J. (2006). Discrete to continuum multiscale bridging. *Multiscaling in Molecular and Continuum Mechanics*.

Fish, J. and V. Belsky (1995). Multigrid method for a periodic heterogeneous medium. *Computer Methods in Applied Mechanics and Engineering* 126, 1–38.

Fish, J., W. Chen, and G. Nagai (2002). Nonlocal dispersive model for wave propagation in heterogeneous media: One-dimensional case and multi-dimensional case. *International Journal of Numerical Methods in Engi-*

neering 54(3), 331–363.

Fish, J. and Z. Yuan (2005). Multiscale enrichment based on the partition of unity. *International Journal of Numerical Methods in Engineering 62*(10), 1341–1359.

Fluhrer, J. (2004). *DEFORM-3D Version 5.0 User's Manual*. Scientific Forming Technologies Corporation.

Frenkel, D. and B. Smit (2002). *Understanding Molecular Simulations: From Algorithms to Applications* (Second ed.). Academic Press.

Gravouil, A. and A. Combescure (2001). Multi-time-step explicit method for nonlinear structural dynamics. *International Journal of Numerical Methods in Engineering* 50, 199–225.

Hoffmann, C. and R. Joan-Arinyo (1998). Cad and the product master model. *Computer-Aided Design 30*(11), 905–918.

Hoffmann, C. and R. Joan-Arinyo (2000). Distributed maintenance of multiple project views. *Computer-Aided Design* 32, 421–431.

Hou, T. and X. Wu (1997). A multiscale finite element method for elliptic problems in composite materials and porous media. *Journal of Computational Physics* 134, 169–189.

Hughes, T., L. Mazzei, and K. Jansen (2000). Large-eddy simulation and the variational multiscale method. *Computing and Visualization in Science* 3, 47–59.

Hull, D. and D. Bacon (1975). *Introduction to Dislocations*. Butter worth-

Heinemann.

Kevrekidis, I. G., C. W. Gear, J. M. Hyman, P. G. Kevrekidis, O. Runborg, and C. Theodoropoulos (2003). Equation-free coarse-grained multiscale computation: Enabling microscopic simulators to perform system-level tasks. *Communications in Mathematical Sciences 1*(4), 715–762.

Knap, J. and M. Ortiz (2001). An analysis of the quasicontinuum method. *Journal of Mechanics and Physics of Solids* 49, 1899–1923.

Li, X., M. Shephard, and M. Beall (2002). Accounting for curved domains in mesh adaptation. *International Journal of Numerical Methods in Engineering* 58, 246–276.

Luo, X., M. Shephard, J.-F. Remacle, R. O'Bara, M. Beall, B. Szab, and R. Actis (2002). p-version mesh generation issues. In 11^{th} *International Meshing Roundtable*, pp. 343–354.

Mavrantzas, V., T. Boone, E. Zervopoulou, and D. Theodorou (1999). End-bridging monte carlo: An ultrafast algorithm for atomistic simulation of condensed phases of long polymer chains. *Macromolecules* 32, 5072–5096.

Miller, R. and E. Tadmor (2002). The quasicontinuum method: Overview, applications and current directions. *Journal of Computer-Aided Materials Design* 9, 203–209.

Noort, A., G. Hoek, and W. Bronsvoort (2002). Integrated part and assembly modeling. *Computer-Aided Design* 34, 899–912.

O'Bara, R., M. Beall, and M. Shephard (2002). Attribute management system

for engineering analysis. *Engineering with Computers 18*(4), 339–351.

OCTA (2006). Meso-scale simulation programs. http://octa.jp/.

Padding, J. and W. Briels (2002). Time and length scales of polymer melts studied by coarse-grained molecular dynamics simulations. *Journal of Chemical Physics 117*(2), 925–943.

Parasolid, Inc. (2006). http://www.ugs.com/products/open/parasolid.

Shephard, M. (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.

Shephard, M., M. Beall, R. O'Bara, and B. Webster (2004). Toward simulation-based design. *Finite Elements in Analysis and Design* 40, 1575–1598.

Shephard, M., J. Flaherty, K. Jansen, X. Li, X.-J. Luo, N. Chevaugeon, J.-F. Remacle, M. Beall, and R. O'Bara (2005). Adaptive mesh generation for curved domains. *Journal for Applied Numerical Mathematics 50*(2–3), 251–271.

Shephard, M. and M. Georges (1992). Reliability of automatic 3-d mesh generation. *Computational Methods in Applied Mechanics and Engineering* 101, 443–462.

Simmetrix Inc. (2006). http://www.simmetrix.com/.

Spatial Inc. (2006). 3d acis modeler. http://www.spatial.com/components/acis.

TSTT Software (2006). http://tstt-scidac.org/software/software.html.

Wan, J., S. Kocak, and M. Shephard (2005). Automated adaptive 3-d forming simulation process. *Engineering with Computers 21*(1), 47–75.

Weiler, K. (1998). The radial-edge structure: A topological representation for non-manifold geometric boundary representations. *Geometric Modeling for CAD Applications*, 3–36.

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 where the left image shows the problem set-up with geometry of the two dies and initial workpiece and the right two images show the initial (top) and final (bottom) meshes.

Figure 6. Adaptive molecular/continuum multiscale simulation.