Comput. Methods Appl. Mech. Engrg. 199 (2010) 1352–1370

Contents lists available at ScienceDirect

Comput. Methods Appl. Mech. Engrg.

journal homepage: www.elsevier.com/locate/cma

Collaborative software infrastructure for adaptive multiple model simulation

Fabien Delalondre *, Cameron Smith, Mark S. Shephard

Scientific Computation Research Center, Rensselaer Polytechnic Institute, 110, 8th Street, Troy, NY 12180-3590, United States

ARTICLE INFO

Article history:
Received 9 February 2009
Received in revised form 12 January 2010
Accepted 15 January 2010
Available online 2 February 2010

Keywords:
Multiscale
Multifidelity
Multimodel
Adaptivity
Software Infrastructure

ABSTRACT

This paper presents a software infrastructure being developed to support the implementation of adaptive multiple model simulations. The paper first describes an abstraction of single and multiple model simulations into the individual operational components with a focus on the relationships and transformations that relate them. Building on that abstraction, consideration is then given to how adaptively controlled multiple model simulations can be constructed using existing simulation components interacting through functional interfaces. This includes addressing how experts would provide the infrastructure with the needed components and define the relations and transformations needed to interact with other components, and for users to define the simulations they wish to be executed. Next, a discussion of the software environment used to implement the multiple model simulation infrastructure is given. Finally, there is discussion of the implementation, using this infrastructure, of two multiscale and one multiple fidelity model simulation applications.

1. Introduction

The ability to translate recent advances in understanding the interactions of phenomena across the atomic, molecular, microscopic, and macroscopic scales into new products and industries requires a transformation in the methodologies of modeling, simulation, and design used by scientists and engineers. Historically, most problems have been addressed through the application of models that consider a single physical model acting on a single scale. Increasingly, the key questions that need to be addressed require consideration of multiple interacting models and scales to represent the phenomena of interest. In response to this need there are ongoing efforts in the development of methods to link multiple models over multiple scales to support engineering simulations. Although the development of these methods represents an area on ongoing research, efforts to this point clearly demonstrate that the ability to address the wide range of scientific and engineering problems of interest requires developing models and scale linking methods that can be efficiently combined to meet the specific needs of the applications being considered. Considering the thousands of person-years that has gone into the development, verification and validation of single scale simulation software, an approach that allows that software to be effectively combined with appropriate scale linking components is the only practical approach for the development and wide spread application of multiple model simulation. This paper describes a software infrastructure being developed to support the various classes of area experts that must contribute to the development and application of multiple model simulations. A key feature of the infrastructure being developed is the ability to support a full set of adaptive model, model linking and discretization control techniques.

Efforts to date have been focused primarily on combinations of methods to demonstrate the capability of a specific scale linking technology. Among the most mature of these is the quasicontinuum method [26,27] that starts from lattice level atomic modeling in which molecular mechanics potentials are applied to the atoms in critical regions. In noncritical regions groups of atoms are approximated by representative atoms with specific assumptions on the variation of fields. Adaptive versions of these procedures have been developed. Other scale linking procedures consider operators to link continuum PDE fields to discrete atomic fields. In several cases the scale spanning operators consider the discretized PDE form (e.g., element mesh) when constructing these operators (i.e. see [7,8,55]). Others define the operators at the equation level. The heterogeneous multiscale methods [17] defines compression operators to relate discrete to continuum and reconstruction operators to relate continuum to discrete scales. The equation-free multiscale method [26] link statistically averaged fine scale realizations to the coarse scale. The concurrent domain decomposition methods employ a blending process between the two models that is carried into weak forms appropriate for solution by generalized discretization methods [19,33].

Errors introduced in the representation of physical phenomena depend on the mathematical model selected, the physical scale on which the mathematical model is applied, the domain of analysis, the boundary/initial conditions and methods of equation

* Corresponding author. Tel.: +1 518 276 6795; fax: +1 518 276 4886.
E-mail address: delalf@scorec.rpi.edu (F. Delalondre).
URL: http://www.scorec.rpi.edu (F. Delalondre).

0045-7825/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.cma.2010.01.011

discretization. Methods to control the errors associated with equation discretization for PDEs are known for several equation classes [2,3]. Extensions to these methods have made it possible to estimate errors in quantities of interest [36]. Recently, these methods have been further extended and applied to atomistic/continuum multiscale methods [40,42]. Another approach to the adaptive control of models, including those acting at different scales, is the use of model breakdown error indicators [8,33].

A large number of software supporting multiphysics and more recently multiscale simulation has been developed in the past few years. Most of the analysis software can be considered as frameworks that use either one or partially two levels of computational abstraction. Commercial analysis packages such as Abaqus [1] are those frameworks that are using only one level of abstraction. They can usually be extended through the implementation of sets of user-defined routines. The fact that these sets are rather small and static prevents from easily extending such software to support complex simulation that includes a large number of adaptive methods and scales. Open source software using one level of abstraction such as PHASTA [64] or LAMMPS [44,45] provides more control due to the possibility for the user to access the source code. However they do require the user to provide extensions using internal software mechanisms and data structures. This forces the user to comply with the concepts, standards and data structures defined by the framework. Another strateg y to extend such kind of software relies on the use of a code coupler strategy such as MpCCI [24]. If this strategy seems to be attractive, it requires defining a coupler component between the considered software that acts as a data converter. This can introduce computational inefficiency that can result in a dramatic loss of scalability when executing simulation on massively parallel machines. Software using a higher level of abstraction tends to separate the needed sets of information such that one defined set can be treated as standalone package. This allows more easily extending the considered software and supporting the use of some level of adaptivity by taking advantage of component interoperability. Software packages such as SIERRA [54], Cactus [12], Salome [47], COOLFluid [14] and COMSOL Multiphysics [13] are frameworks that use higher levels of abstraction and a better separation of the set of information being involved. Using such an approach, complex applications can be generated by plugging components that respect interfaces defined by the framework. Execution of applications is realized by following rules dictated by the considered framework. If this approach allows creating a complex application and supporting the use of some level of adaptive control through component interoperability, the fact that the user must follow a rather large set of rules dictated by the framework prevents from supporting the flexibility needed to support the easy integration of other existing components.

Based on that analysis, it is believed that there is a lack of generalized structures and tools for the effective implementation of adaptive multiscale and other multiple model methods. To provide the desired flexibility, component-based software along with a higher level of abstraction that supports high component interoperability and model linking needs to be developed. An infrastructure that efficiently supports adaptive multiple model simulation must consequently address:

- General interfaces between the various functional and information components of a simulation with specific care to allow the integration of existing simulation components.
- General methods to transfer information between the models and scales over interacting portions of the space/time domain including the operators required.
- Supporting of various classes of experts to work together to define and execute multiple model simulations.
- Supporting adaptive control of models, model linking procedures and discretizations.

This paper is organized as follows: The next section gives an abstraction of multiple model simulations into functional components that can be adaptively controlled through functional interfaces. It also provides a description of the design of an infrastructure that will support area experts in the construction and execution of multiple model simulations. Section 3 discusses the implementation aspects of the infrastructure to meet the goals defined in Section 2. Section 4 discusses how the infrastructure is being used to support the implementation of three different adaptive multiple model applications that include (i) hierarchic multiscale analysis of biotissues, (ii) concurrent multiscale analysis of mechanical failure processes, and (iii) multiple fidelity modeling of airframe structures with specific consideration of the design of structural connections.

2. Infrastructure design

The persons executing simulations are focused on having simulation answer performance questions. To satisfy the defined questions, the software infrastructure is required to fulfill the following requirements:

1. Efficiently build the needed multiple model simulations.
2. Support adaptivity to control the simulation such that it provides the best available result satisfying the user goal. Consequently, a complete adaptive procedure requires supporting:
   (a) The identification and estimation of all used levels of approximation.
   (b) The ability to use available alternatives to reduce levels of approximation in a flexible way.
   (c) The construction of hierarchy of alternatives to guide the reduction of levels of approximation.
3. Support various expert interactions.
4. Integrate existing components.

The following sub-sections detail abstractions and structures that satisfy these requirements.

2.1. Adaptive multiple model simulation abstraction

A key problem with current simulation software is its rigid implementation in terms of a specific fixed model and no separation of simulation transformation processes. Even with the level of effort that has gone into single model simulation software, the vast majority of it does not interact with a general abstraction of the simulation. This is a key reason that even though adaptive procedures have been known for years, they are still not in common use. Consequently, the first step in the design of a software infrastructure to support adaptive multiple model simulations is to carefully abstract the hierarchy of information needed, operations executed and transformations required to go from a physical problem description, through the application of mathematical models, to the construction of the computation models used to solve the mathematical models (requirement (2.a)).

The process of defining and generalizing adaptive multiple model simulations must begin with consideration of a single model simulation. Fig. 1 shows a general abstraction of an adaptive single physical model simulation into a set of interacting components. It is divided into four hierarchic levels of abstraction that include a conceptual model, mathematical model, computational model and computational system. The information defined within each level is placed into one of the three groups:

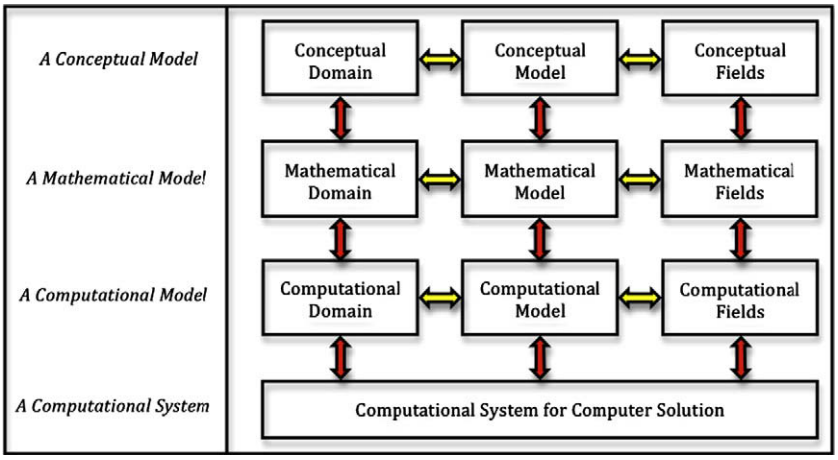

Fig. 1. Physics simulation abstraction.

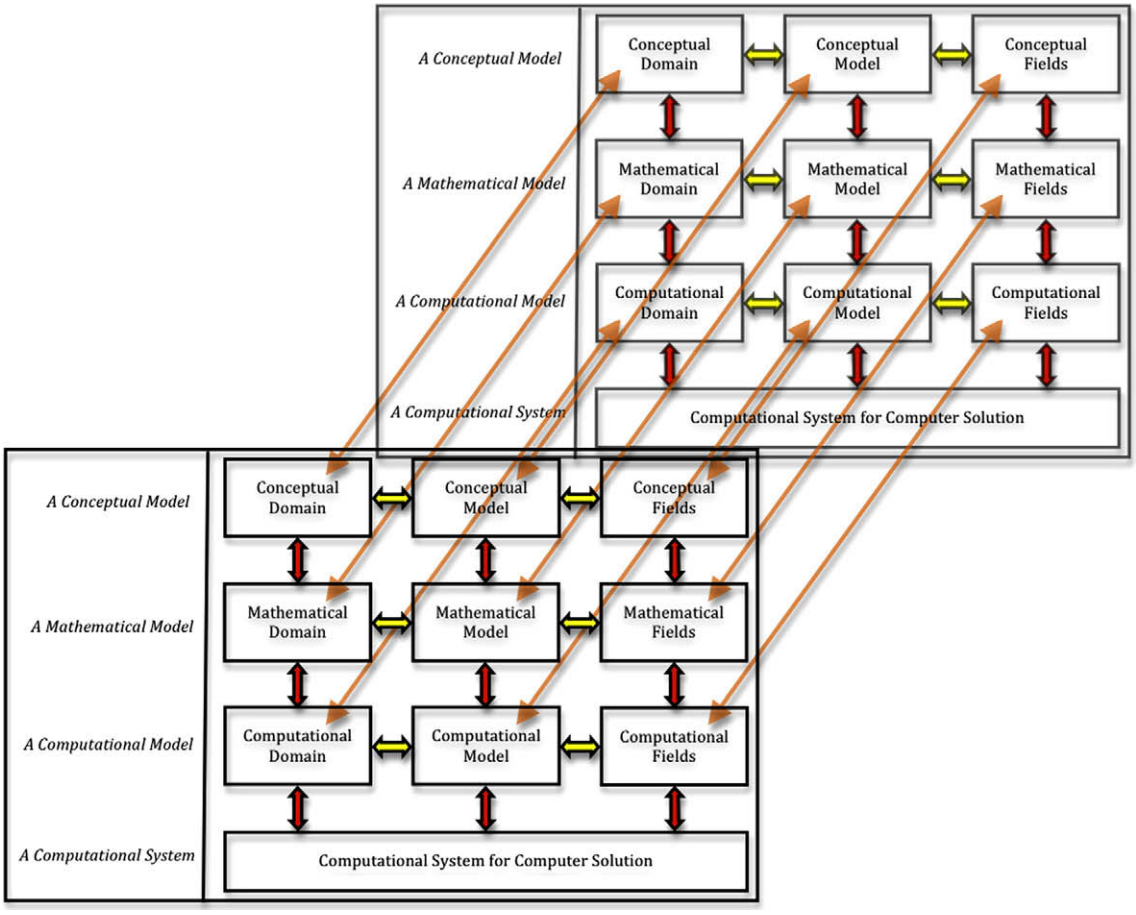

Fig. 2. Abstraction of linked physics simulations.

the domain, model and fields. Those groups are related to each other (horizontal arrows in Fig. 1), to form the specified level of abstraction. The transformations required to traverse levels are represented by the vertical arrows.

The key addition in the abstraction of multiple model simulation is the structure and functions to perform both relations and transformations of information across models (oblique arrows in Fig. 2).

The top level of each model consists in a generalized statement of the problem that defines its conceptual representation. Each of the lower levels represents the result of the transformation of the information from the previous level until a computational system appropriate for solution on a digital computer is constructed. Each transformation, represented by both vertical (Fig. 1) and oblique arrows (Fig. 2), changes the form of the information into the one needed for the next step. The operations involved in the

transformations can be complex and involve the introduction of various levels of approximation. The effective identification and adaptive control of these approximations are central to an effective application of simulation.

The transformations between components in linked physics simulations is performed by the model linking functions that supports independently defined rules for the component interactions. As an example of the issues addressed by the model linking, consider linking atomistic and continuum models in which case the field transformations must deal with technical issues such as relating quantities on different scales (e.g., atomistic forces and continuum stresses), defining quantities not defined on the fine scale (e.g., temperature), filtering unneeded high frequency components as moving up in scale, and accounting for statistical variation as traversing scales. Since the computational fields are typically constructed through a discretization process, the transformation functions for the computational fields must also account for that process.

An outline of the three information groups follows:

2.1.1. Domain

The domain corresponds to the spatial and temporal region used in the definition and execution of a physics simulation. The corresponding hierarchic abstraction is described as follows:

- *Conceptual domain:* Represents the space/time domain on which a physical phenomenon is to be studied. At this level of abstraction, the explicit representation of the domain needs not yet to be specified.
- *Mathematical domain:* Is a mathematically complete definition of the conceptual domain. It is a function of the type of mathematical model being used for the domain. For example, For example, a continuum domain definition is needed in the case of PDEs while a set of atomic positions is needed in molecular dynamics.
- *Computational domain:* Corresponds to the discretization of the mathematical domain and thus consists of a collection of discrete entities. In the case of a continuum domain its geometric accuracy depends upon the precision with which the boundaries of the domain are represented.

2.1.2. Model

The model represents the physical phenomena of interest that take place over the defined space/time domain. The corresponding hierarchic abstraction is defined as follows:

- *Conceptual model:* A general statement that formalizes the physical problem to be solved. It indicates the overall physical principles that govern the considered problem.
- *Mathematical model:* An explicit mathematical description of the physics of interest as given at the conceptual level. It becomes functional when associated with both mathematical domain and fields.
- *Computational model:* The transformation of the mathematical model into the computational model is focused on providing the framework for the construction of the computational system, which is ultimately sets of algebraic equations. It consists of discretizing the equations that have been specified within the mathematical model when those models are infinite dimensional.

In the case of discrete systems like molecular dynamics the potentials are applied to discrete entities directly yielding algebraic relationships. Since the unmodified application of general potentials typically can lead to unacceptable large systems, there are often approximations made at this level. These approximations can be as simple as the use of a cut-off function to limit the distance of interactions considered, or may be a more complex set of coarse graining operations. Maintaining a knowledge of the approximations made here is needed so that adaptive procedures to improve those approximations can be applied when needed. In the case of continuum mathematical models the transformation of the mathematical model to the computational model typically employs a formal discretization process. The most commonly applied discretization methods are based on a double discretization process in which the domain is decomposed into a geometric grid or mesh and the equation parameter s, the computational tensor fields, are approximated over the entities of the mesh by piecewise distribution functions times yet to be determined multipliers, referred to as degrees of freedom (DOF). The result of this process is a set of mesh entity level algebraic systems that are assembled into the computational system that is then solved to determine the values of the DOF. The processes of executing the computational model defines specific relationships between entities in the computational domain and the computational tensor fields. Three common methods that employ different combinations of interactions between the mesh entities and the distributions that define the computational tensor fields are finite difference, finite volume and finite element methods [52].

2.1.3. Fields

To complete the description of the physical problem to be solved, fields need to be specified. The three hierarchic levels of abstraction are defined as follows:

- *Conceptual fields:* Define the physical parameters that are used by the conceptual model to describe the physical phenomena that takes place over the considered conceptual domain.
- *Mathematical fields:* The set of variables that are used within the mathematical model to mathematically represent the physics of interest over a portion of the defined mathematical domain. They are tensor quantities that can be a function of the mathematical domain as well as other mathematical fields. Mathematical fields are defined by their distribution over mathematical domain entities. For example in the case of solving PDEs over continuum domains, the distribution of the given input tensors are effectively related to topological entities of regions, faces, edges and vertices [38] (e.g., constitutive matrix for a region, a distributed traction over a surface, etc.).
- *Computational fields:* The representation of the mathematical fields that corresponds to the discretized distribution of the defined fields over the discretized entities of the considered portion of the computational domain. For example, in the case of a continuum PDE solved using finite elements, the computational fields are tensors discretized into a set of shape functions time degrees of freedom. A single tensor field can be used by a number of different analysis routines that interact, and the field may be associated with multiple computational models using different distributions.

The presented abstraction supports part of the requirements by introducing a flexible organization of information needed throughout the execution of multiple model simulation where all levels of approximation are clearly identified (requirement (2.a)). In the following, an infrastructure that supports such abstraction and completes the satisfaction of the requirements is detailed.

2.2. Simulation interaction and control

The development and deployment of a complete application requires various interactions with users through a hierarchy of inter-

faces (requirement (3)). The knowledge required to successfully incorporate new functionality (requirement (4)) is much different from that needed to initiate and adaptively execute a simulation (requirements (1) and (2)). Thus the processes involved require different interactions with users with different domains of expertise. The three classes of users with different forms of expertise are:

- Application experts who are focused on studying the physical problem of interest. Application experts are responsible for being able to answer specific performance questions of interest and using the results provided to advance their application. Their interaction with the infrastructure is limited to asking performance questions, providing the problem physical description and receiving the results.
- Modeling experts provide the physical/numerical models that are introduced within the infrastructure to support simulations. Their interaction with the infrastructure consists in defining the abilities of available components and using them to implement physical/numerical models.
- Computational experts provide components to support the development of physical/numerical models. Their interaction with the infrastructure consists in implementing the functionality of components by incorporating simulation software into the infrastructure.

Fig. 3 presents the overall structure that satisfies the defined requirements and supports user interactions by making use of the defined set of abstractions and associated transformation processes. The construction of simulations is supported by the definition of simulation components that represent one or a group of several of the components defined within the physics simulation abstraction.

The overall structure consists of two main parts which are the simulation initialization (left hand side in Fig. 3) and the adaptive simulation execution (right hand side in Fig. 3).

Before an actual simulation can be executed the modeling and computational experts must have provided the system a broad enough set of simulation components and information on how those components interact so that the goals of the simulation to be requested by an application expert can be evaluated (requirement (4)). Assuming a sufficient set of components the application expert begins by indicating the simulation goals in terms of a set of performance parameters to be evaluated and level of accuracy desired for the parameters. They must also provide the physical information needed for the conceptual definition of the simulation. This includes providing the space/time domain of interest, indicating the physics of interest, and providing the parameters needed to evaluate the conceptual fields.

The mapping of simulation goals onto the simulation components library allows the creation of a subset of components that are combined to create a hierarchy of physics simulations that can be applied to answer the requested performance question (requirements (1) and (2.c)). Fig. 4 depicts the hierarchy built from a subset of simulation components. At the highest level the interacting physics simulations abstracts the component interactions defined in Fig. 2. At the intermediate level is the abstraction of a physics simulation defined in Fig. 1 into ten components. At the lowest level is the abstraction of each component into the subset of components capable of fulfilling its definition. The defined hierarchy is organized in the sense of the amount of physics included and/or degree of accuracy possible to satisfy simulation goals.

The flexible definition of this hierarchy is supported by the use of interoperable interfaces between components that supports requirement (2.b). An interoperable component is a procedure that interacts through a functional interface to obtain the needed model, domain and/or field information and performs transformation operations on that information as needed at that point in a multiple model simulation. Such functional interfaces, which are being used in some single model simulation procedures [7,50], provide an effective mechanism to couple alternative software packages to support multiple model simulation. By defining the interfaces based on an appropriate level of information abstraction, completeness and generality, it becomes easy to evaluate the ability of alternative single model simulation components and model linking processes to properly perform the required operations within the context of a multiple model simulation process.

Given the simulation goals and hierarchy of physics simulations, the adaptive simulation execution step is responsible for applying the hierarchy that satisfies the defined performance question in the most effective manner (requirements (1) and (2)). The adaptive simulation execution step is monitored by the simulation state controller that interacts with the built hierarchy of physics simulations, the active simulation state, the adaptive controller and, as needed, the simulation state history. The active simulation state, which is represented in Fig. 4 by the filled squares, is defined as the physics simulations that have been selected to form the active state of the simulation. The adaptive processes that support this selection ensure the satisfaction of the defined simulation goals throughout the execution of the simulation. The goal of the adaptive processes is to control the errors introduced within the used transformations such that it provides the best available result satisfying the user goal. This control is achieved by the adaptive controller which is responsible for making the active simulation state evolve, whenever it is required, by realizing the following adaptive sequence:

1. The active simulation state is evaluated using the physics simulation evaluator (Fig. 5) that is in charge of quantifying the different levels of errors made throughout the execution of the adaptive multimodel simulation.

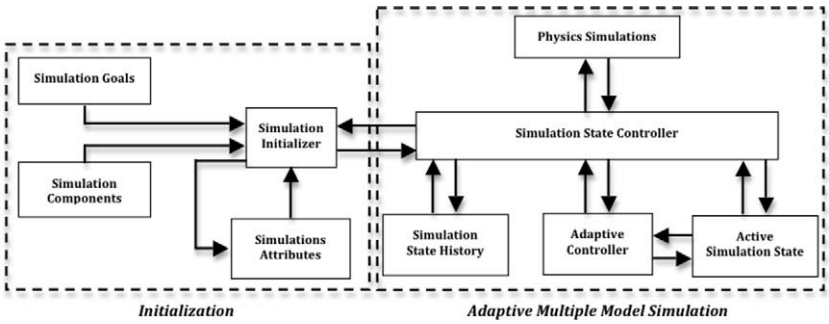

Fig. 3. Overall concept.

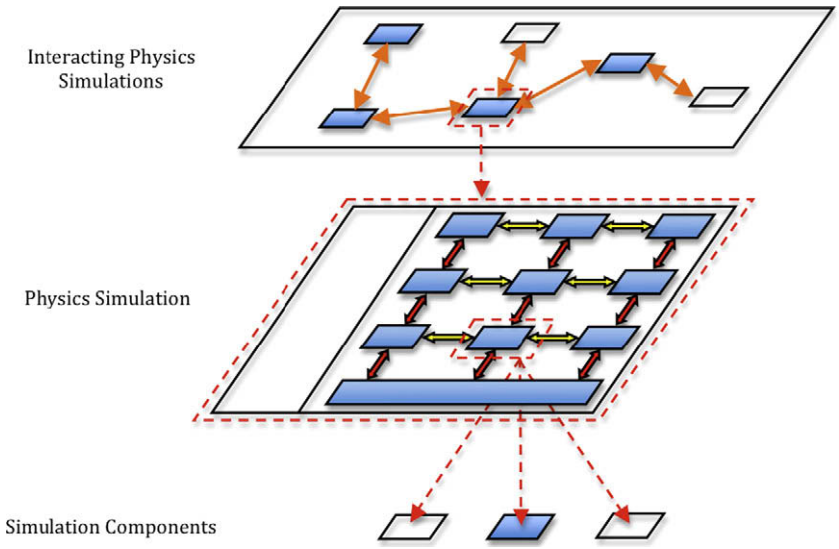

Fig. 4. Graph-like structure representing the abstraction of a multiple model simulation and the built hierarchy. Active physics simulations and components are filled squares.

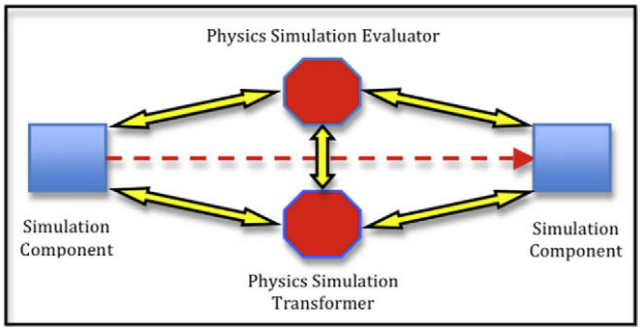

Fig. 5. Adaptive transformation.

2. If the active simulation state does not yet satisfy the simulation goals, the models, transformations, and or discretizations that need adaptation are determined by using comparison with bounds defined in the physics simulation evaluator.
3. The required model changes, transformation changes and discretization modifications are determined and executed using the physics simulation transformer (Fig. 5).
4. The next active simulation state is built whereas the previous state is transferred to the simulation state history.

This adaptive sequence is repeated until either the simulation goals are reached or the available hierarchy of physics simulations is considered as insufficient to reach the defined goals.

The present paper is focused on providing the fundamentals to support adaptivity via component transformation. Ongoing efforts are now focusing on providing full sets of methods for guiding adaptivity in a multi level/model approximation framework. Preliminarily developments to date demonstrate the need for:

- A multi level approximation estimation strategy that will support error evaluation for each defined transformation within a defined physics simulation (Fig. 1). The error defined within each transformation can be a combination of different classes of error.
- A multimodel approximation estimation strategy that will support error evaluations when coupling multiple models (Fig. 2) is needed. Key to this is determining appropriate error accumulation when coupling models.

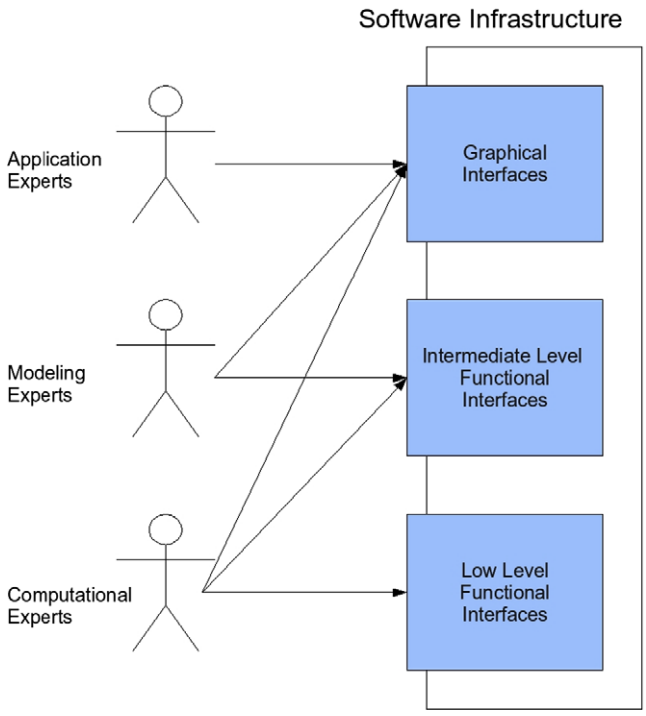

Fig. 6. Expert user interactions with hierarchy of interfaces.

3. Implementation

Realization of the software infrastructure capable of adaptive multiple model simulations requires functionality to interface independent simulation software (requirement (4)) and to maintain their respective performance levels (requirement (1)) throughout the adaptive simulation process (requirement (2)) while supporting expert user interactions (requirement (3)).

The interfacing technologies accomplish this via a hierarchy of interfaces (Fig. 6). At the highest level graphical user interfaces allow application experts to design simulation goals and modeling/computational experts to define relational information. At intermediate levels object oriented, component-based and logic pro-

gramming functional interfaces are leveraged to define associations between software components. Lastly, at the lowest level high performance functional interfaces support interfacing computationally or communication intensive routines.

3.1. Adaptive multiple model simulation support

The software infrastructure for supporting adaptive multiple model simulations is built upon the abstractions of Section 2.1. Functional interfaces of the ten components are used to implement the information passing and transformation routines with minimal overhead. Implementation of the functional interfaces is in the programming language C [25] because of its ability to efficiently interface with several other languages common to performance oriented computing such as FORTRAN [32] and C++ [61]. An object oriented data structure can define the organization of the ten components as depicted in Fig. 1. The Python programming languages [29,31,46] object oriented paradigm allows run time the storage of references to the software libraries implementing the ten components. The Python ctypes [21] package allows Python functionality to be ex tended with C language libraries and thus permits access to the component functional interfaces. The definition of such a hierarchy of functional interfaces supports the efficient resolution of computationally intensive tasks (requirement (1)) while providing the flexibility to adapt and possibly swap any defined component at run time without having a predefined execution sequence (requirement (2.b)).

A data tunnel is used to support the layered implementation of the interfaces. It is an association between components permitting access to functional interfaces such that information passing and transformation procedures can be implemented without adding significant computational overhead. Establishment of a data tunnel between the computational domain and the computational model is demonstrated in Fig. 7 for information passing.

The detailed implementation helps in satisfying part of the defined requirements by:

- Providing a flexible framework that supports the information organization detailed in Section 2.1 (requirement (2.a)) as well as an extensive use of adaptive processes throughout the simulation (requirement (2.b)).
- Providing levels of functional interfaces to support:
  - Computational and modeling expert interactions (requirement (3)).
  - Introduction of existing simulation components (requirement (4)).

In the following section, the implementation of an infrastructure that completes the fulfillment of the requirements by supporting infrastructure interaction and control is detailed.

3.2. Simulation interaction and control support

The first step required for the application expert to be able to use the infrastructure consists for both modeling and computational experts in introducing computational capabilities (requirement (4)). Incorporation of software to the infrastructure described in Section 2.2 by computational experts requires expressing the relational and performance parameters of the component(s) with respect to existing components. Application of such data is most intuitive through a graphical user interface that provides a visualization of the relational structure. As such a graphical user interface defines the highest level of the hierarchy of interfaces (requirement (3)).

An attribute definition language [38,55] implements a template of the information levels and groups such that any simulation component can be described and related to existing components. A graphical user interface was implemented to instantiate component template instances such that a graph of instances is built (Fig. 8) and stored in a database.

Interface functionality allows defining multiple performance parameters, properties, and multiple component interactions, component connections (Fig. 9). Existing simulation software, such as Abaqus [1], tend to encompass several components of a physics simulation. Likewise the incorporation of software components into the infrastructure requires instantiating them as groups of components; the section headed member components of Fig. 9 supports this. A physics simulation is defined by specifying its performance parameters and by selecting composing components.

Once computational capabilities have been introduced into the infrastructure, the simulation initialization as defined in Section 2.2 can start with the creation of simulation goals. The specification of goals is through a graphical user interface. This interface implements the functionalities and interactions of the simulation initializer of Fig. 3. The typical user of this interface, an application expert as described in Section 2.2, has knowledge of the problem domain and simulation goals in the form of a set of performance parameters to be evaluated and level of accuracy desired for the parameters. Knowledge of the physics models, numerical methods and subsequent software implementations is not a concern for such users. The procedures abstract this level of detail through the goal input form and a feedback mechanism that relays the results of the goal mapping process onto simulation components. The goal input form accepts user knowledge of simulation goals and transforms it into that which can drive the feedback mechanism. Implementation of the feedback mechanism is with an inference engine. An inference engine provides a convenient mechanism to maintain an evolving set of relationships between the various components and transformations in a manner that allows evaluation of the validity of the utilization of components or transformation based on simulation goals. Implementation of the inference engine is in the programming language Prolog [65,66]. A functional interface

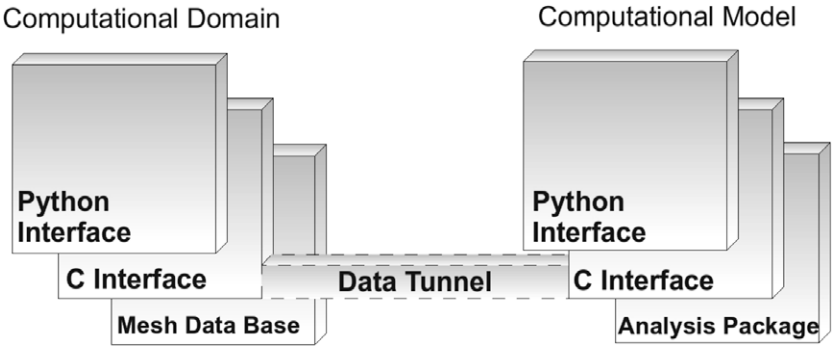

Fig. 7. Data tunnel concept supporting transformation and information passing.

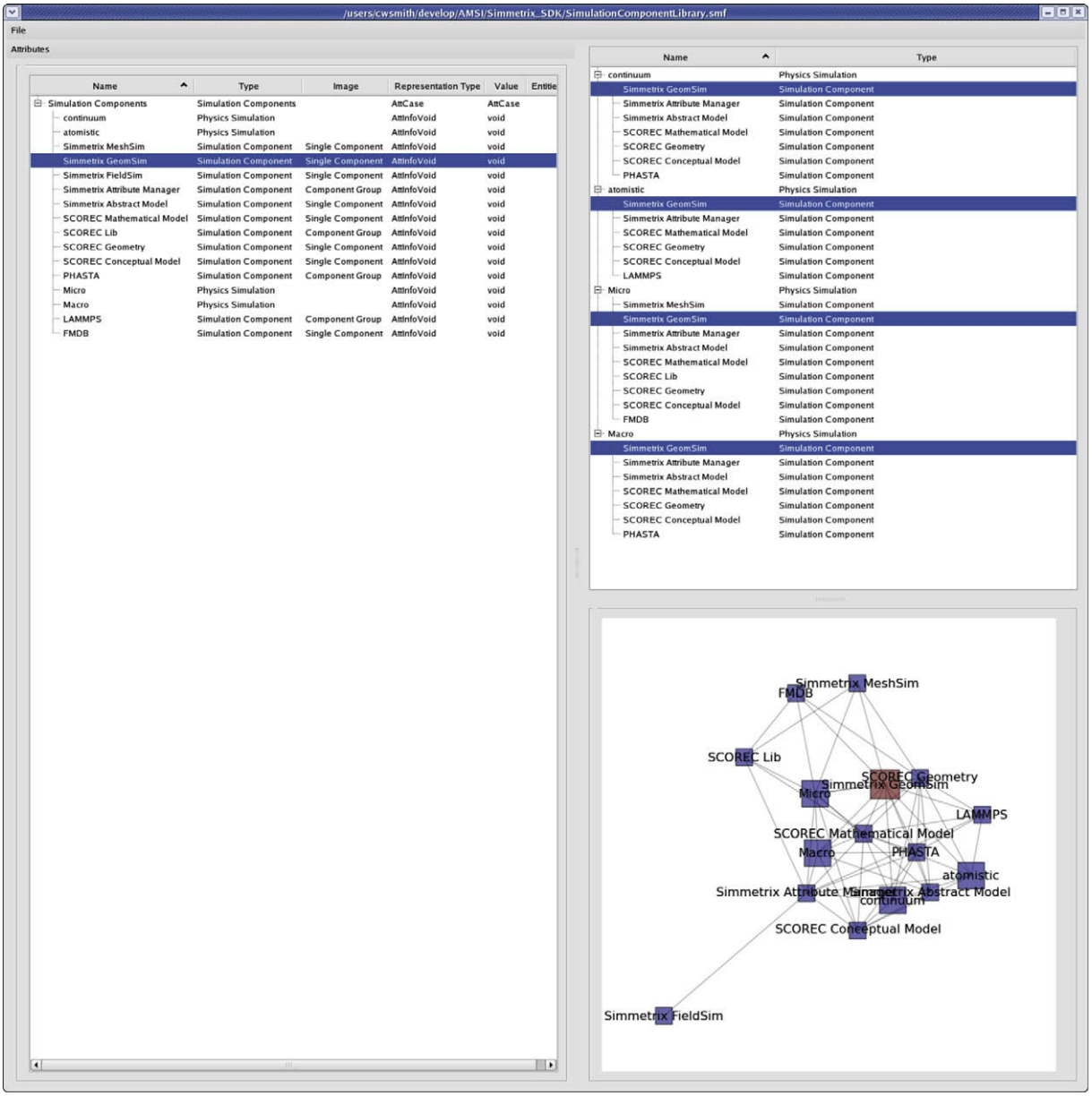

Fig. 8. Graphical user interface for abstraction definition.

to the inference engine provides functionality for definition of goals, executing the inferencing process, and interacting with the results of the process to support requirements (1) and (2).

Execution of the goal mapping process is through a graphical interface. The user goals are first transformed into the standard form [23] for the inference engine. The inference engine then builds a list [10,16,60] of physics simulations sorted by those which have performance properties that best match the performance goals (requirements (1) and (2.c)). Next physics simulations which connect to the best matching physics simulation of the list are collected and sorted. Lastly simulation components which compose the physics simulations are found and sorted. A graph [9,18,63], such as depicted in Fig. 4, can define the organization of the physics simulations and their composing simulation components. The graph is built as a set of connected physics simulation nodes. Each node has subgraphs for each component that contains the possible composing components. The best matching physics simulation (simulation component) to the performance goals is noted within the current graph (subgraph). The set of best matching simulation components composing a physics simulation define the Python organizational structure for a physics simulation. Fig. 10 illustrates a possible graph in which the shaded nodes are those best matching the performance goals and the current state of the simulation.

For each physics simulation the user launches a graphical user interface capable of defining simulation attributes at the highest possible level of a physics simulation. Fig. 11 depicts a graphical user interface which provides this functionality by loading modeling and computational expert defined simulation. This attribute information is stored and associated with the physics simulation in the graph. Once all physics simulations have been defined the user can then execute the simulation process from the graphical interface.

The experts define procedures supporting execution of the adaptive simulations (requirement (3)). For each simulation component a setup procedure is implemented with the Python ctypes

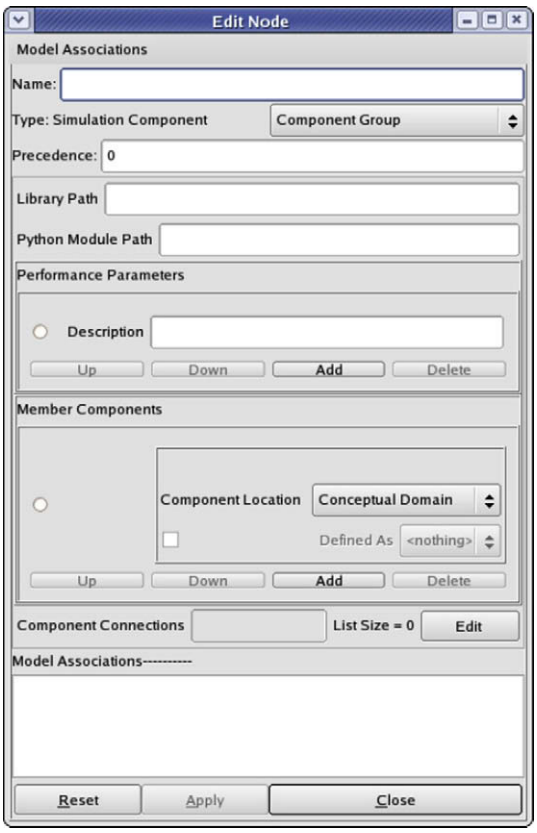

Fig. 9. Template for abstraction composition via groups.

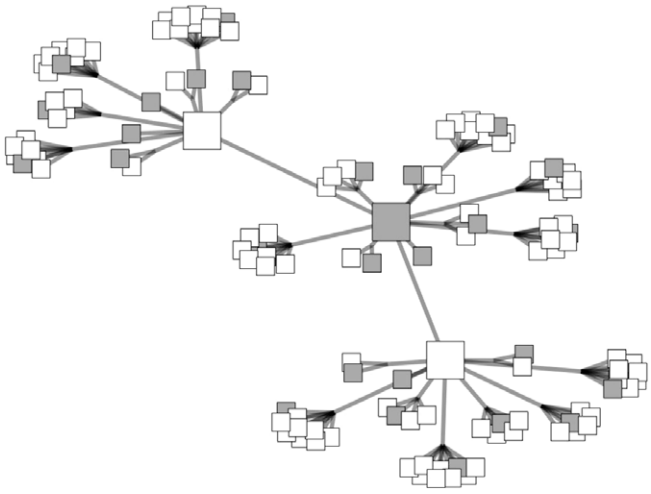

Fig. 10. Graph of a multiple model simulation.

functions to load their implementation libraries (requirement 2.b)), and the C interface functions to satisfy their relational dependencies (requirement (1)). For each physics simulation a solve procedure defines the execution sequence and implements the functionalities and interactions of the simulation state controller of Fig. 3. This procedure first deploys the setup functions of the components that define the physics simulation of interest. The solve routine then implements the execution sequence with the functions of the component interfaces at, and connected to, those at the computational level.

The adaptive simulation sequence discussed in Section 2.2 begins by calling the solve procedure of the physics simulation which best matched the performance goals set by the user. The graph containing this procedure and data implements the physics simulations and active simulation state of Fig. 3. The active simulation state is maintained by setting a nodes state to active when its solve or setup procedure is invoked. A node is deactivated when the Python deactivate function for each physics simulation (component) is called.

Evolving the simulation state (requirements (1) and (2.a,b)) requires adaptive control of models, transformations and discretizations which can range from mathematically specified error control techniques to simple rules (e.g., has a material limit been reached). The adaptive control mechanisms must leverage the functional interfaces and relationships defined to make the decisions based on the simulation result information available.

High-level control of these processes is through the inference engine that supports the ability to control execution of the a posteriori error measurements and correction indications procedures that are key to performing the specific model and discretization adaptations to be executed. Maintenance of computational efficiency is achieved through calls to component functional interfaces that support the execution of computationally intensive adaptive routines. Both functional interfaces to the inference engine and the simulation component implement the adaptive controller of Fig. 3. The adaptive sequence described in Section 2.2 and implemented within the solve routine of a physics simulation is described as follows:

1. Complex error evaluation methods that define part of the physics simulation evaluator functionalities are supported through calls to component functional interfaces. Examples of methods requiring such type of support include ZZ superconvergent patch recovery [67] and goal oriented strategy methods [40–43].
2. Simple evaluation operations that define the rest of the physics simulation evaluator functionalities are supported through calls to the inference engine functional interface. Examples of such operations include comparing error estimation values to defined error bounds to determine whether a specific limit has been reached [10,16,60].
3. Both complex calculation of transformation information and transformation realization that define part of the physics simulation transformer functionalities are supported through calls to component functional interfaces. Examples of methods requiring such type of support include the concurrent multiscale [36], the equation free [26] and hierarchical modeling methods [39].
4. High-level and computationally non intensive adaptive methods that define the rest of the physics simulation transformer functionalities are supported through calls to the inference engine functional interface. Examples include the selection of components through the mapping of performance goals to the evolving graph of possible simulation components and physics simulations.
5. Construction of the next simulation state and transfer of the previous simulation state to the simulation state history is supported through calls to the inference engine.

Infrastructure implementation satisfies requirements by:

- Efficiently building adaptive multimodel simulation that answers the asked performance (requirements (1) and (2)) question using:
  - Mechanisms to support various error estimation methods to quantify all levels of approximations (requirement (2.a)).

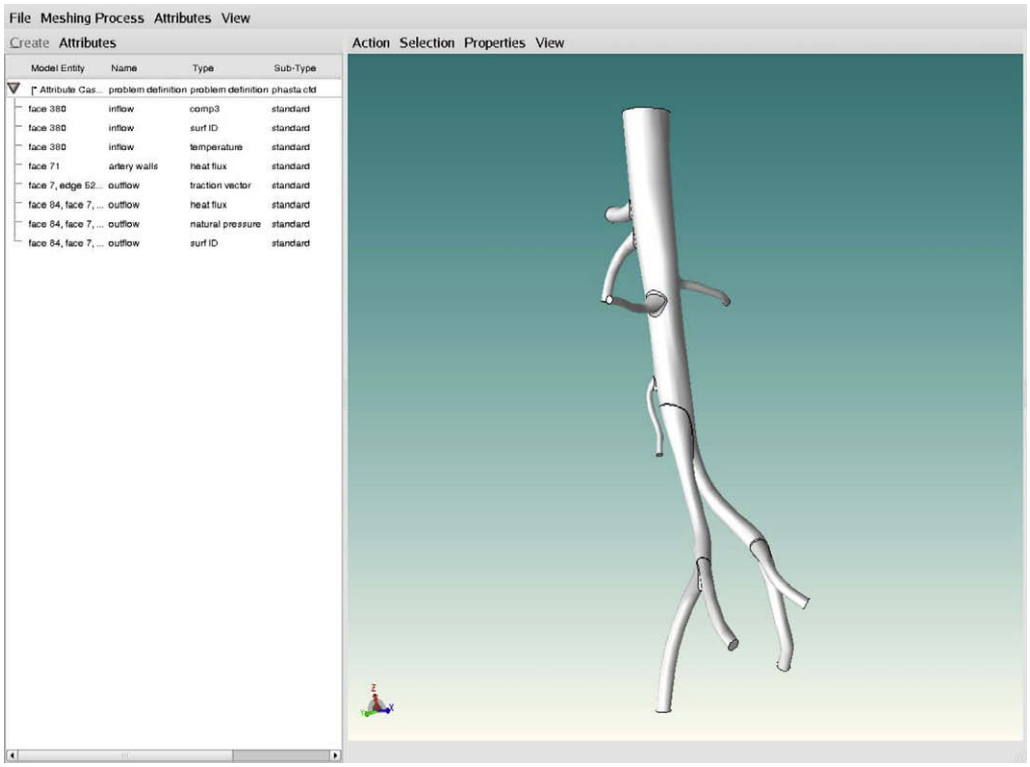

Fig. 11. Graphical user interface for attribute definition of a computational fluid dynamics problem.

- A hierarchy of functional interfaces that provides flexibility while maintaining efficiency (requirement (2.b)).
- The automatic construction of hierarchies of models as well as mechanisms to maintain and evolve model relationship at run time (requirement (2.c)).

- Interfacing various expert interactions (requirement (3)) and providing mechanisms to easily integrate existing simulation components using a hierarchy of interfaces (requirement (4)).

3.3. Simulation components integrated into the infrastructure to date

A number of powerful simulation components have been introduced within the library (requirement (4)). Multiple implementations are available for a number of specific components. Using component interoperability allows selecting the most appropriate implementation of a desired functionality during simulation execution.

- Conceptual domain component and transformation from conceptual to mathematical domain (Fig. 1).
  - *Abstract model:* Structure containing abstractions for domains which are independent of a specific CAD model of the problem domain. The Abstract Model [56] provides functionality to define the instantiation process, thus containing the important linkage information to lower level models. The Abstract Model can deal with multiple hierarchal decompositions and multiple viewpoints [11,22,35], with specific functionalities to support a full range of simulation viewpoints [51].

- Transformation from conceptual to mathematical domain and mathematical domain (Fig. 1).
  - *GeomSim:* Provides capabilities to load and query models from a wide variety of sources. Its unified topological model allows all geometry sources to be queried using a standard set of functions [5,6,58]. It also supports specific sets of geometric modifications.
  - *SCOREC model:* Is an open source library implementing geometry interface interrogation functionalities [5].

- Computational domain component and transformation from mathematical to computational domain (Fig. 1).
  - *Flexible Mesh Data Base (FMDB):* Mesh management library that is able to provide a variety of services for mesh users [48,49,53].
  - *MeshSim:* Component software for automatically generating unstructured meshes directly from CAD models [5,6,59].

- Conceptual and Mathematical fields components, and transformation from mathematical to computational fields (Fig. 1).
  - *Attribute manager:* Supports the specification of information, past that of the domain definition, needed to qualify an engineering analysis. The information managed by this system includes various order tensors needed to specify the analysis attributes of material properties, loads, and boundary conditions as well as additional data constructs used by the analysis such as strings, and references to either other attributes or model entities [38,58].

- Computational fields component, and transformation from mathematical to computational fields (Fig. 1).
  - *FieldSim:* Component software for storing, querying and manipulating solution information on a mesh. It provides standard ways to query simulation data from a wide variety of sources [5,57].

- Computational and mathematical model and fields components, transformation from mathematical model and fields to computational model and fields, construction of computational system (Fig. 1).
  - *SCOREC Library:* Collection of tools that support the definition and resolution on a defined domain of a wide variety of problems using finite element method [5].

- Computational domain, model and fields components, and the construction and resolution of computational system for molecular statics and dynamics (Fig. 1).
  - *Large-scale atomic/molecular massively parallel simulation (LAMMPS):* Molecular statics and dynamics potentials for soft materials (biomolecules, polymers) and solid state materials (metals, semiconductors) and coarsegrained or mesoscopic systems [44,45].

- Mathematical domain, computational domain, model and fields components, and the construction and resolution of computational system for solid continuum mechanics (Fig. 1).
  - *Abaqus:* Software suite delivers solutions for non-linear solid mechanics problems [1].

Using those components the following physics simulations have been introduced by the modeling expert:

- Continuum mechanics:
  - Reduced dimension linear continuum mechanics analysis using shell theory that supports elastic behavior of the material [1].
  - Full dimension continuum solid mechanics analysis using non-linear finite element method that includes large deformation. Both microscale and macroscale constitutive relationships are supported.
  - Full dimension continuum solid mechanics analysis using finite element method that supports both linear and non-linear elastic behaviors.
  - Full dimension continuum solid mechanics analysis using non-linear finite element method that supports linear elastic behavior and non-linearities due to contact.

- Discrete analysis using non-linear finite element method where a representative volume element (RVE) is filled with fibers to determine pointwise force/deformation relations. The fibers are represented by axial members having a non-linear behavior [30].
- Atomistic model using concurrent multiscale analysis with an overlap region [20,36,37] and molecular statics with embedded atom method potential [37].

4. Applications

Three applications requiring the use of multiple interacting models are outlined. Each application, namely, a hierarchic multiscale, concurrent multiscale and multifidelity modeling application, requires the use of multiple models and the support of the interaction between the models. The aim of this section is to demonstrate the flexibility of the multimodel infrastructure to support adaptive multimodel simulations. Each application is organized as follows: the physical problem of interest is first outlined. Then the interaction of the three levels of users to support the definition and execution of the simulation is presented. The sequence realizing the adaptive resolution of the multiple model simulation is then discussed considering the adaptive processes required to answer the performance question in the best available way.

4.1. Hierarchic multiscale

4.1.1. Problem description

The guiding principle of functional tissue engineering is that the engineered tissue must achieve biocompatibility and functionality. Underlying the challenge of functional tissue engineering is the fact that accurate prediction of mechanical properties of the engineered tissue requires careful consideration of the state of material's microstructure.

In the present case, the application expert is studying the mechanical behavior of an artery made of collagen tissue (Fig. 12). The aim is to determine the influence of microscopic fiber orientation on macroscopic displacements and stress when the artery is subjected to macroscopic traction boundary conditions.

4.1.2. Simulation goal mapping

The application expert defines the following simulation goals:

- Macroscopic displacements.
- Macroscopic stress state.
- Microscopic fiber orientation.

The physics simulations that result from the application of the simulation goals on the simulation components library are described as follows:

- Full dimension continuum solid mechanics analysis using non-linear finite element method written for a displacement formulation that includes large deformation solved using a standard updated lagrangian method to represent the macro scale. The force/deformations relationship at material point is supported by a compressible Neo-Hookean hyperelastic macroscopic constitutive relationship ((1), [68]) that can be calibrated based on microscopic numerical tests [4].This physics simulation will be referred to as macroscale state 1. It is supported by abstract moel, attribute manager, SCOREC library, Abaqus, SCOREC model and FMDB components.

$$\sigma_{ij} = \frac{\mu}{J}(g_{ij} - \delta_{ij}) + \lambda(J-1)\delta_{ij}, \tag{1}$$

  where σ is the Cauchy stress tensor, μ and λ are the Lamé parameters, g_{ij} is the left Cauchy strain tensor and δ_{ij} is the Kronecker delta function.
- Full dimension continuum solid mechanics analysis using non-linear finite element method written for a displacement formulation that includes large deformation solved using a standard updated Lagrangian method to represent the macro scale. The force/deformations relationship at material point is supported by a microscale discrete analysis solution. This physics simula-

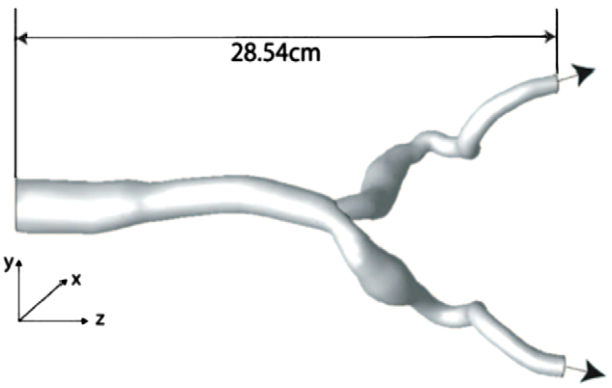

Fig. 12. Artery subjected to traction.

tion will be referred to as macroscale state 2. It is supported by abstract model, attribute manager, SCOREC library, FMDB and SCOREC model components.

- Discrete analysis using non-linear finite element method where a representative volume element (RVE) is filled with fibers to determine pointwise force/deformation relations to represent the micro scale. The fibers are represented by axial members having a non-linear behavior described by a phenomenological equation of the form [62],

$$F_f = \frac{E_f A_f \left(e^{B\varepsilon_f} - 1\right)}{B}, \tag{2}$$

$$\varepsilon_f = 0.5\left(\lambda_f^2 - 1\right), \tag{3}$$

where $F_f, E_f, A_f, \epsilon_f$ and λ_f respectively represent the force exerted on the fiber, fiber linear elastic modulus, fiber cross-sectional area, fiber's Green strain, fiber stretch ratio. The B parameter is defined as a constant. This physics simulation will be referred to as microscale. It is supported by abstract model, attribute manager and SCOREC model, FMDB and SCOREC library components.

Interactions between the selected physics simulations are realized as follows:

- *Between macroscale states 1 and 2:* The model is transformed to support the change in the constitutive relationship definition. However as this change can create a discontinuity in the force/displacement curve, a stabilization parameter is added during the transformation as follows:
  - The displacement u_2 computed prior to the transformation of macroscale state 2 into macroscale state 1 is used to obtain the stress state $\sigma_1(u_2)$ calculated with the macro constitutive relationship.
  - The difference of stress states is computed and used to determine the value of a stabilizing parameter σ_{stab} that ensures the elimination of the discontinuity,

$$\sigma_{stab} = \sigma_2(u_2) - \sigma_1(u_2). \tag{4}$$

- Between macroscale state 2 and microscale:
  - Interaction from macroscale state 2 to microscale: macroscopic displacements at Gauss point are transformed to provide the micro displacement Dirichlet boundary conditions needed by the discrete analysis. The macroscopic deformation gradient F_M is computed and provided to the microscale model. It is then transformed to set Dirichlet boundary conditions on the RVE using the following equation [28]:

$$x_m = F_M \cdot X_M, \tag{5}$$

where x_m is the microscale position vector whereas F_M and X_M are the macroscale deformation gradient and position vector.

- Interaction from microscale to macroscale state 2: volume averaging techniques are used to transform stresses and their derivatives from RVE boundaries to macro scale integration points. The position and force of fiber points crossing the RVE boundaries are used to compute the macroscopic averaged quantities on the deformed configuration [15] using the following equations,

$$S_{ij} = \frac{1}{V} \sum_{\text{boundary crosslinks}} x_i f_j, \tag{6}$$

$$S_{ij,j} = \frac{1}{V} \oint \left(s_{ij} - S_{ij}\right) u_{k,i} \eta_k \, dS, \tag{7}$$

where S_{ij} and s_{ij} are the macro and micro stresses, x_i the i-component of the position of the boundary cross-link, f_j is the force developed on the boundary cross-link in the j-direction, u is the displacement of the RVE boundary and η_k is the unit normal vector.

The defined physics simulations are organized using a two level hierarchy (Fig. 13). The use of adaptive methods allows automatically selecting material integration points that do not require the explicit computation of the microstructure (i.e. the group formed by macroscale state 2 and microscale physics simulations) to satisfy the defined simulation goals. The following adaptive methods have been selected:

- *Physics simulation evaluator:* The relative difference of macroscopic displacements between two consecutive load steps is evaluated,

$$Eval = \left|\frac{\Delta(u)}{u}\right|. \tag{8}$$

- *Physics simulation transformer:* Regions where the evaluated relative difference of macroscopic displacements is less than 5% are selected for not computing a new microscopic solution. In those regions, macroscale state 2 components are transformed into the ones that support macroscale state 1.

Those adaptive methods are supported by the use of both SCOREC library component and AMSI inference engine.

4.1.3. Simulation attributes

Using the attribute GUI, a tensile load is applied on the right side of the arterial bifurcation along the z-direction as shown in Fig. 12 (the left hand side remaining fixed) to increase the length from 28.54 cm to 31.39 cm (10 % of elongation) in 20 incremental steps 0.5% of elongation by step). The values of the material parameters used for the microscale constitutive relationship are defined such that $\frac{E_f A_f}{B} = 8 \times 10^{-7}$ N. The B constant is set to 1.2. The volume fraction is less 5% with 362 fibers in the RVE.

4.1.4. Simulation results

Figs. 14–16 depict macroscopic displacements and stress state along with the corresponding microstructure as parts of the results requested by the application expert [30].

To allow the study of the influence of fibers orientation on macroscopic results as detailed in [62], the infrastructure has executed the following sequence: Prior to the actual first step of the simulation execution, macrosopic quantities are initialized by executing the physics simulation representing the macroscale that supports the use of macroscopic stress/displacement constitutive relationship (macroscale state 1). The first step of the simulation execution

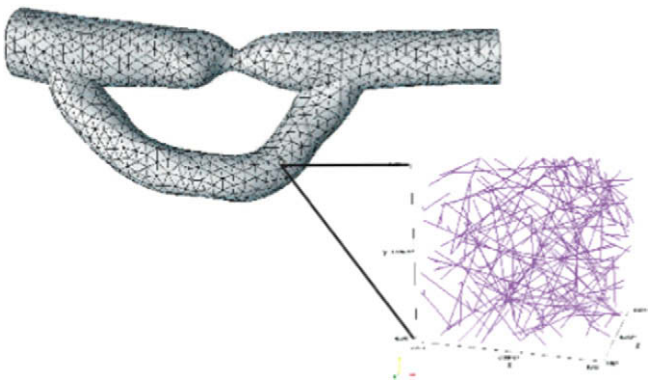

Fig. 13. Domains representing both macro and micro scales.

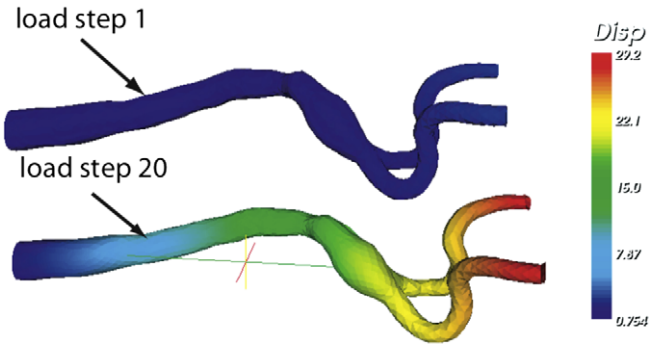

Fig. 14. Macrosopic displacements contour at the initial and final load steps.

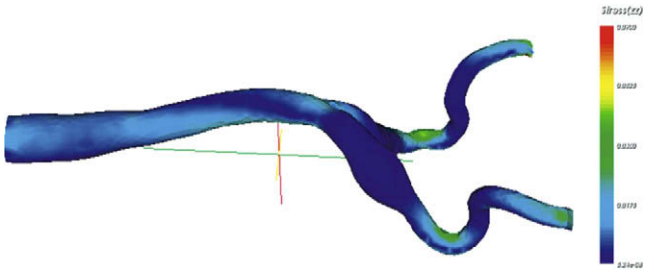

Fig. 16. Macrosopic Szz contour at the final load step.

begins with the transformation of macroscale 1 into the interacting group of physics simulations that is made of both macroscale 2 and microscale. The required macroscale quantities are transferred into macroscale 2, and then transformed and transferred to microscale. Within microscale, mechanical equilibrium is determined and fiber orientations are updated. Microscopic quantities are transformed and transferred back to macroscale state 2. After each load step, the selected adaptive methods are executed and the needed transformations occur as described if it is required.

4.2. Concurrent multiscale

4.2.1. Problem description

The macroscopic behavior of materials is inherently governed by the physics that take place at finer scales. To improve material design, it is necessary to study processes taking place at the nanoscale and their relationship to phenomena that appear at the macroscale.

In the present case, the application expert is studying the influence of the formation and propagation of dislocations on macroscopic deformation in a porous material subjected to hydrostatic loading-unloading.

4.2.2. Simulation goal mapping

The following simulation goals are specified by the application expert:

- Strain of the material at the macroscopic scale.
- Energy of the atoms that composed the material at the nanoscale [37].

The physics simulations that result from the application of the simulation goals on the simulation components library are described as follows:

- Full dimension continuum solid mechanics analysis using finite element method that supports both linear and non-linear elastic behaviors (9), (10) to represent the macro scale. When the current state of the physics simulation representing the macro scale requires the use of components that support linear or non-linear behavior it will respectively be referred to as macroscale states 1 and 2.

$$\sigma_{ij} = F_{iK}F_{jL}C2_{KLMN}E_{MN}, \tag{9}$$

$$\sigma_{ij} = F_{iK}F_{jL}\left(C2_{KLMN} + \frac{1}{2}C3_{KLMNOP}E_{OP}\right)E_{MN}, \tag{10}$$

  where $C1$, $C2$ and $C3$ are elastic constants, F is the deformation gradient and E the Green–Lagrange strain. Big Roman subscripts I, J,... denote components of a tensor in the reference or undeformed configuration and small Roman subscripts i, j,... denote components of a tensor in the current or deformed configuration. Macroscale states 1 and 2 are supported by SCOREC library, FMDB, SCOREC model and attribute manager, MeshSim, GeomSim and abstract model components.
- Atomistic model using concurrent multiscale analysis (11)–(13) with an overlap region [20,36,37] and molecular statics with embedded atom method potential [37] to represent the nano scale (14)–(16). The equilibrium equation in the interface Ω^I is defined as follows,

$$\left(\Theta^C(x)\sigma_{ij}\right)_{,j} + \Theta^C(x)b_i + \sum_{\alpha}^{n^I}\left[\left(\sum_{\beta}^{neig^{\alpha}}\left(\Theta^A_{\alpha\beta}f_{i\alpha\beta}\right) + \Theta^A_{\alpha}b_{i\alpha}\right)\delta(x - x_{\alpha})\right], \tag{11}$$

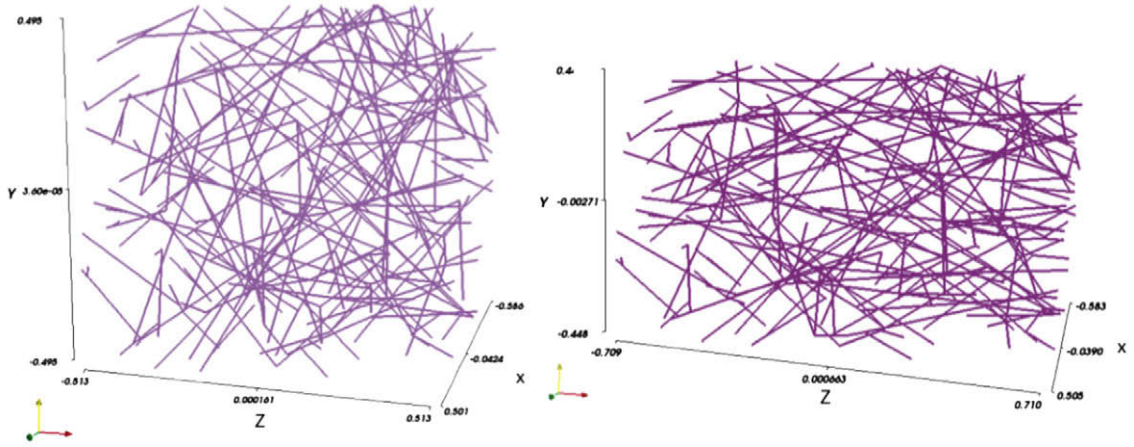

Fig. 15. Fiber meshes at the initial (left) and final (right) load steps.

where

$$\Theta^A_\alpha = 1 - \Theta^C(x_\alpha), \tag{12}$$

$$\Theta^A_{\alpha\beta} = 1 - \frac{1}{2}\left(\Theta^C(x_\alpha) + \Theta^C(x_\beta)\right), \tag{13}$$

and n^I is the number of atoms in the interphase, Θ^A_α is the value of the atomistic blending function for the atom position x_α, Θ^C is the continuum blending function for a continuum material point x, σ and b are the Cauchy stress tensor and the body force per unit volume for the continuum, $f_{i\alpha}$ and $b_{i\alpha}$ respectively are the sum of the internal forces and the body force acting on atom α. The continuum blending function $\Theta^C(x_\alpha)$ is evaluated based on the proximity of the point $x_\alpha \in \Theta^I$ to the boundaries Γ^{CI} and Γ^{AI}. For instance $\Theta^C(x) = 1$ on Γ^{CI} and $\Theta^C(x) = 0$ on Γ^{AI}. For the EAM inter-atomic potential, the total energy Φ of a system of n atoms is obtained as the sum of energies of individual atoms Φ_α defined as,

$$\Phi = \sum_\alpha^n \Phi_\alpha, \tag{14}$$

$$\Phi_\alpha = E(\rho_\alpha) + \frac{1}{2}\sum_{\beta,\beta\neq\alpha}^{neig^\alpha} V(r_{\alpha\beta}), \tag{15}$$

$$\rho_\alpha = \sum_{\beta,\beta\neq\alpha}^{neig^\alpha} \Psi(r_{\alpha\beta}), \tag{16}$$

where ρ_α is the total electron density at atom α, $E(\rho_\alpha)$ is the embedding energy function. $r_{\alpha\beta} = |x_\alpha - x_\beta|$ is the distance between atoms α and β. $V(r_{\alpha\beta})$ is the pair potential term and $\Psi(r_{\alpha\beta})$ is the electron density function, which has a cutoff distance in terms of r as defined by the inter-atomic potential. Thus the summation in Eqs. (15) and (16) is over the atoms in a neighborhood of the atom α denoted by $neig^\alpha$. Greek letters $\alpha, \beta, \ldots$ denote atoms and are used as subscripts to represent the quantities related to atoms. Also there is no summation convention on the Greek subscripts. This physics simulation will be referred to as nanoscale state. It is supported by LAMMPS, SCOREC model and GeomSim, abstract model and attribute manager components.

Interactions between the selected physics simulations are realized as follows:

- *Between macroscale states 1 and 2:* The model is transformed to support the change in the elastic constitutive relationship definition.
- *Between macroscale state 2 and nanoscale:*
  - *From macroscale state 2 to nanoscale in the overlap region [20,36,37]:* The atomistic model is a function of the continuum model. In the overlap region, continuum and atomistic fields are blended using the blending functions described in Eq. (11). Equilibrium of both continuum and atomistic fields is solved using the defined weak integral form. Least square integral forms are applied along with Lagrange multipliers to constrain atomistic displacements (17).

$$\int_{\Omega_I} \lambda_i(x) \sum_\alpha^{n^I} \left(u^C_i(x_\alpha) - u^A_{i\alpha}\right)\delta(x - x_\alpha)d\Omega = 0, \tag{17}$$

    where λ_i is the Lagrange multiplier constraining the i-direction, u^C_i is the i-component of the continuum displacement, $u^A_{i\alpha}$ is the i-component of the displacement and x_α the position of atom α.
  - From nanoscale to macroscale state 2 in the overlap region [20,36,37]: The continuum model is a function of the atomistic model. In the overlap region, the continuum displacements are interpolated using the defined continuum shape functions at atom positions to set the Dirichlet boundary conditions at the atomistic scale.

The defined physics simulations are organized using a three level hierarchy. The following adaptive methods allow automatically selecting the proper model from the defined hierarchy:

- From linear to non-linear elasticity models (i.e. from macroscale state 1 to macroscale state 2):
  - *Physics simulation evaluator:* The relative error in the energy between the linear model and non-linear model is evaluated using the following equation,

$$Eval_{NL-L} = \left|\frac{\epsilon^{NL}_{\Omega^e} - \epsilon^{L}_{\Omega^e}}{\epsilon^{NL}_{\Omega^e}}\right|, \tag{18}$$

    where $\epsilon^{NL}_{\Omega^e}$ and $\epsilon^{L}_{\Omega^e}$ are respectively defined as follows,

$$\epsilon^{L}_{\Omega^e} = \int_{\Omega^e}\left(\frac{1}{2!}C2_{KLMN}E_{KL}E_{MN}\right)d\Omega, \tag{19}$$

$$\epsilon^{NL}_{\Omega^e} = \int_{\Omega^e}\left(\frac{1}{2!}C2_{KLMN}E_{KL}E_{MN} + \frac{1}{3!}C3_{KLMNOP}E_{KL}E_{MN}E_{OP}\right)d\Omega. \tag{20}$$

  - *Physics simulation transformer:* Regions where the defined tolerance is exceeded are selected for being resolved using the non-linear elasticity model (macroscale state 2).

- From non-linear elasticity to atomistic model (i.e from macroscale state 2 to nanoscale):
  - *Physics simulation evaluator:* The stress gradient dislocation nucleation criteria as defined in [20,34,36,37] is evaluated,

$$Eval_{NL-A} = |m_i curl(\sigma)_{ij} l_j|, \tag{21}$$

    where m_i is the direction of the Burger's vector and l_j the dislocation line in the direction j.
  - *Physics simulation transformer:* Regions where the defined tolerance is exceeded are selected for representation as an atomistic region solved using molecular statics model (nanoscale) with a concurrent formulation at the continuum/atomistic interface.

These adaptive methods are supported by the SCOREC library and LAMMPS components as well as the AMSI inference engine.

4.2.3. Simulation attributes

The problem domain is a 622.56 Å^3 block with four spherical nano-voids of 32.256 Å in diameter. An hydrostatic tensile loading-unloading is applied quasi-statically through Dirichlet boundary conditions in increments of 0.4665 Å on each face of the cube. The material parameter used for both the continuum and atomistic models can be found in [20,36,37].

4.2.4. Simulation results

Figs. 17 and 18 present the distribution of energetic atoms for the last step of macroscopic strain as parts of the results requested by the application expert.

To study the influence of dislocations and their propagation on the macroscale deformation in a porous material, the infrastructure has executed the following sequence: The simulation execution begins with the execution of macroscale state 1 that realizes equilibrium at the macroscale on the entire domain. The adaptive methods are executed and transformation of macroscale state 1 into macroscale state 2 is achieved in selected regions. The computation continues with the execution of both macroscale states 1 and 2 that again realize the equilibrium at the macro scale. Adaptive methods are executed and transformation of macroscale state 1 into macroscale state 2 and from macroscale state 2 to nanoscale are achieved in selected region. Equilibrium on the entire domain is finally obtained via the execution of the selected physics simulations.

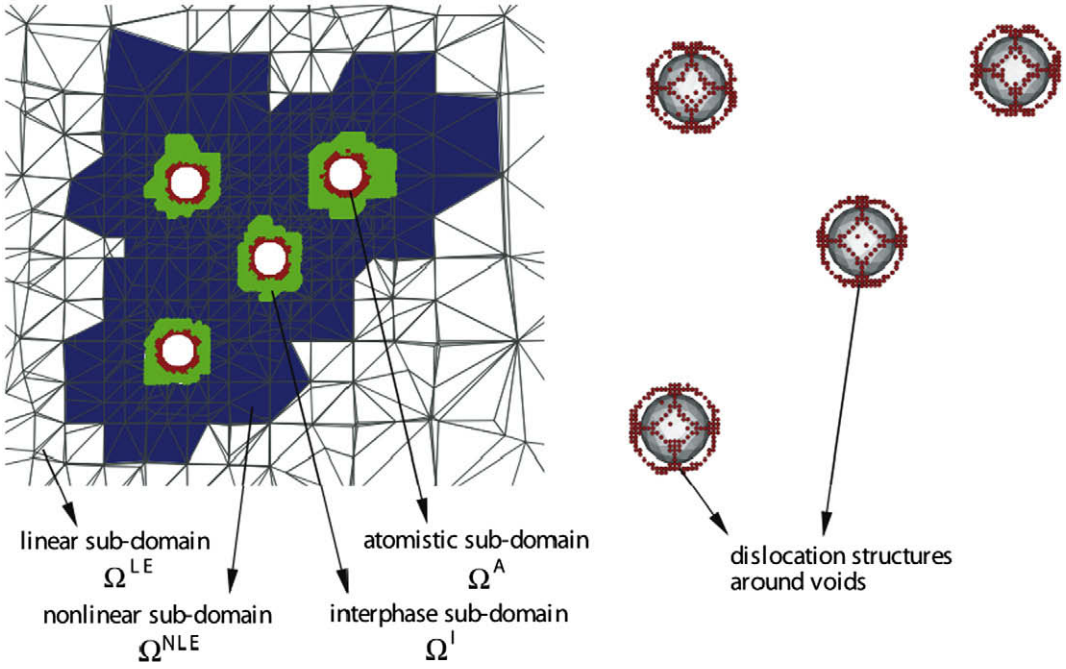

Fig. 17. The concurrent model and the dislocation structures around voids-load step 20 (trace of strain = 0.0945).

Fig. 18. Advanced dislocation structure around voids-load step 28 (trace of strain = 0.126).

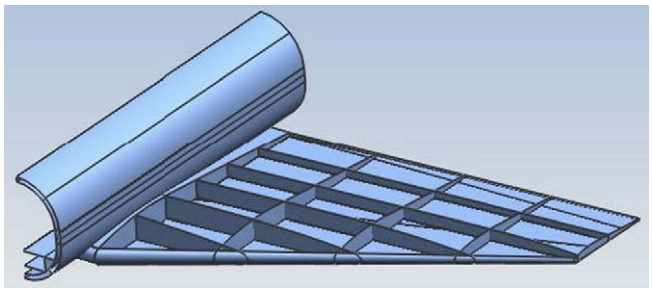

Fig. 19. Aircraft wing structure using shell elements.

4.3. Multifidelity modeling simulation

4.3.1. Problem description

Accurately determining the behavior of joints in aircraft design is especially important since the efficient design of joints directly impacts the ability to produce lightweight structures, and inaccurate results could lead to catastrophic failure.

In the present case, the application expert is studying the behavior of the joints (Fig. 20, second row) that compose the structure of an aircraft wing (Fig. 19) subjected to aerodynamic lift that can most likely experience fracture.

4.3.2. Simulation goal mapping

The following simulation goals are specified by the application expert:

- Von Mises macroscopic stress greater than the design limit.
- Joint design parameters.

The physics simulations that result from the mapping of the defined simulation goals on the simulation components library are described as follows:

- Reduced dimension linear continuum mechanics analysis using shell theory that supports elastic behavior of the material to model the behavior of the wing structure that does not include structural joint details. This physics simulation will be referred to as shell physics simulation. It is supported by Abaqus, SCOREC model, FMDB and abstract model, attribute manager, GeomSim, MeshSim components.
- Full dimension continuum solid mechanics analysis using nonlinear finite element method that supports linear elastic behavior and non-linearities due to contact solved using a standard penalty method to model an isolated portion of the structure including either a basic or a detailed 3D representation of a joint (Fig. 20, second row). The detailed representation of the joint allows for the full geometry of the connectors and full set of contact models that include a standard linear Coulomb friction

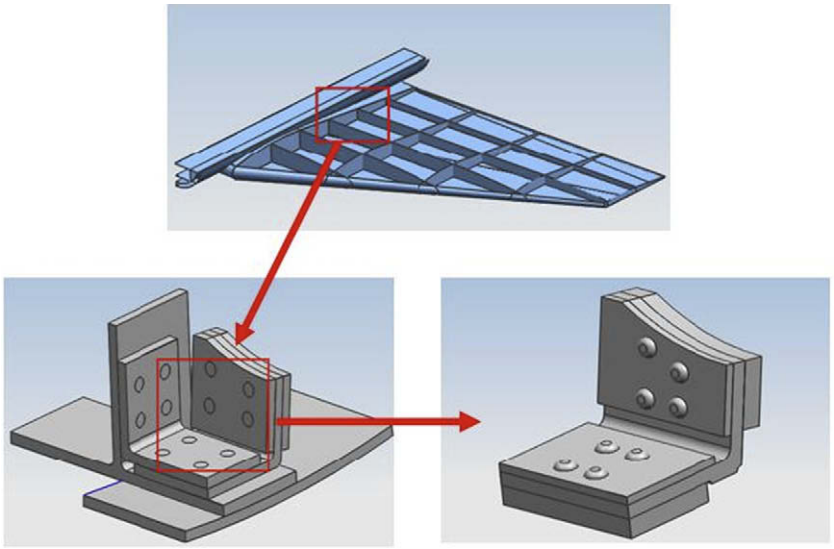

Fig. 20. Hierarchy of models for the studied multifidelity modeling simulation.

model. Both basic and detailed 3D physics simulation are supported by Abaqus, SCOREC model, FMDB and abstract model, attribute manager, GeomSim, MeshSim components.

Interactions between the selected physics simulation are realized as follows:

- Between shell and 3D physics simulations: The mathematical representation of the selected portion of the domain is transformed from a middle surface to fully 3D. The geometric information defining the joint (Fig. 20) is also included. Displacements and rotations of the shell physics simulation are transformed to 3D displacements using standard shell/3D-continuum relations and finite elements interpolation techniques to provide the boundary conditions of the 3D physics simulations.
- *Between basic and detailed 3D physics simulations:* The mathematical representation of the domain is transformed to take into account the full details of the joint including the connectors. Displacements of the basic 3D physics simulation are transformed using finite elements interpolation techniques to provide the boundary conditions of the detailed 3D physics simulation.

The three defined physics simulations are organized using a three level hierarchy (Fig. 20) where the data are exchanged through a one way coupling scheme using a top (simple model)-down (complex model) approach. This method of coupling, although clearly not optimal, is seen as the first step towards the development of a more general multifidelity modeling simulation framework that could involve a two-way coupling scheme as often used in multiscale simulations [20,36,37]. As the present paper is focused on describing both design and implementation of an infrastructure that supports the development of such a multifidelity framework and not on its development itself, details dedicated to that specific topic is postponed to an up coming paper. The following adaptive methods allow automatically selecting the proper model from the defined hierarchy:

- From shell analysis to basic 3D representation of joint:
  - *Physics simulation evaluator:* Determine load, from shell stresses, through key joints. Apply company design rules to identify joints requiring more evaluation and design based on total load transferred to the joint.

$$Eval_{Sh-3Dbasic} = |\sigma^{VM}_{FE} - \sigma^{VM}_{DesignRule}| \geqslant \epsilon. \quad (22)$$

With σ^{VM}_{FE} being the Von Mises stress computed using finite elements method and $\sigma^{VM}_{DesignRule}$ the company provided limit.
  - *Physics simulation transformer:* Isolate the local region and bring in the joints 3D geometry for representation. Interpolate the required kinematic boundary conditions.

- From basic 3D representation of joint to detailed 3D representation of joint:
  - *Physics simulation evaluator:* (a) Evaluate 3D stress states from basic 3D simulation and indicate the need for a detailed 3D simulation for joints requiring more detailed evaluation to determine design parameters. (b) Control the mesh discretization error of the detailed 3D representation using a standard SPR projection based error indicator [67].
  - *Physics simulation transformer:* (a) Replace the simplified contact representation of the connectors with a detailed model of the connector and include frictional contact. (b) Convert the element level error indicators into a new mesh size field that is used by a general mesh modification procedure to adapt the mesh to match the new mesh size field.

Those adaptive methods are supported by the use of abstract model, GeomSim, SCOREC model and Abaqus simulation components as well as AMSI inference engine.

4.3.3. Simulation attributes

The wing is subjected to a vertical and uniform pressure of 200 MPa representing the aerodynamic lift (vertical arrows in Fig. 21). Symmetric boundary conditions have been applied on the left hand side of the fuselage. The material used for the plates is an Aluminium having elastic constants equal to $E = 71.7$ GPa and $\nu = 0.33$. Rivets are made of Steel with $E = 209$ GPa and $\nu = 0.29$. The friction coefficient value used within the linear Coulomb friction model of the detailed 3D representation is equal to 0.8.

4.3.4. Simulation results

Figs. 22 and 23 present the distribution of Von Mises stress obtained by the shell and detailed 3D physics simulation as parts of the results requested by the application expert.

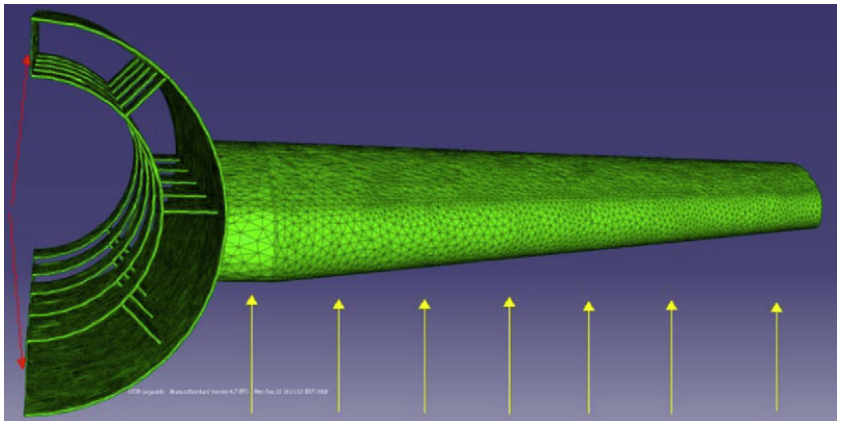

Fig. 21. Attributes defined on the aircraft wing structure.

To allow the study of the joint behavior in the structure of a wing aircraft subjected to aerodynamic lift, the infrastructure has executed the following sequence: The simulation begins with the execution of shell physics simulation that models the entire wing structure using shell analysis (Fig. 22). Adaptive methods are executed and transformation from shell to the two 3D physics simulations in the region where Von Mises stress is greater than the design limit (squared in Fig. 22) is realized. The computation continues with the execution of the 3D physics simulation. It allows selecting and studying the region(s) of the structure that include the joint(s) where the Von Mises stress is greater than the specified design limit (Fig. 23). Adaptive mesh refinement is finally used to improve the solution obtained on the detailed 3D representation.

5. Closing remarks

This paper has presented a collaborative infrastructure designed to address what has been identified as the key points to support the development of various adaptive multiscale/multimodel simulation tools:

- The use of a set of generalized interfaces that support the easy incorporation of new methods and software components while maintaining computational efficiency.
- The support of information transfer between the models and scales over interacting portions of the space/time domain.
- The interactions with the three classes of experts, the application, modeling and computational experts, involved in the definition and execution of multimodel simulations.
- The introduction of adaptive model, discretization and model linking error control.

The infrastructure employs a set of generalized interfaces that support the easy incorporation of new methods and software components while maintaining computational efficiency and supporting the inclusion of adaptive model and discretization error control. The infrastructure has been designed to support the interactions with the three classes of experts, the application, modeling and computational experts, involved in the definition and execution of multimodel simulations.

The implementation of the infrastructure takes advantage of a number of interoperable simulation structures and components

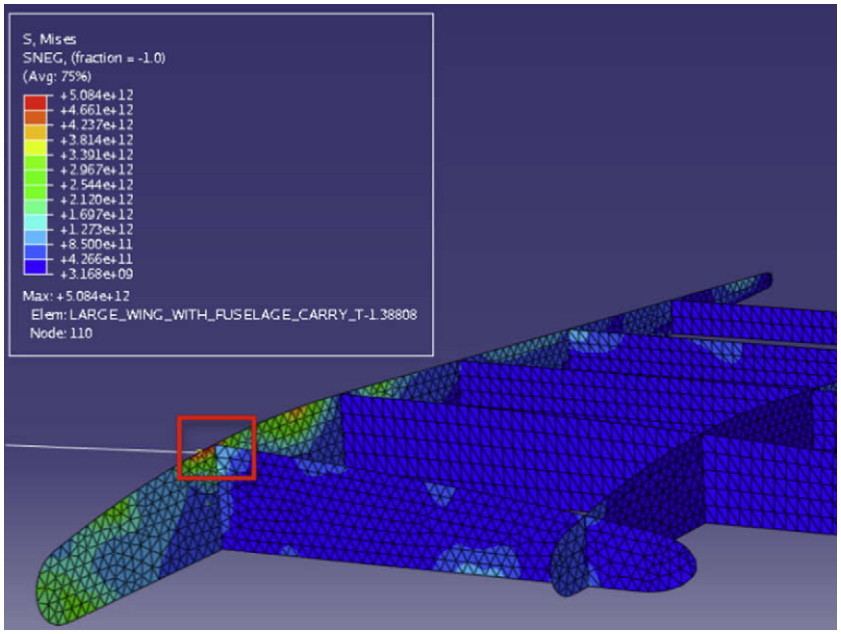

Fig. 22. Von Mises stress distribution in the overall wing structure using shell analysis.

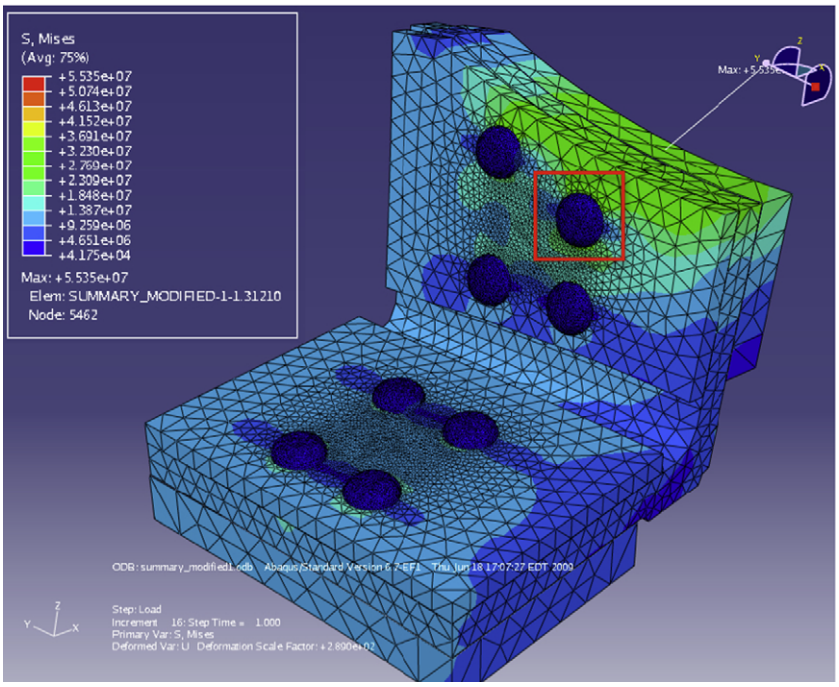

Fig. 23. Von Mises stress distribution in a region selected for detailed 3D analysis.

that have been under development by multiple groups for a number of years. Although some aspects of the infrastructure are not yet fully developed, the component-based approach being used and the availability of a number of key structures and components has allowed the effective implementation of the three multimodel applications described in Section 4.

This paper is seen as the first step toward efficient multiple model simulation where extensive use of adaptive methods is being supported. Ongoing and future research efforts are focused on improving the infrastructure and developing both multiple model error estimation methods to guide the adaptive procedure and an adaptive coupled multiple model framework that ensures the reliable resolution of multifidelity modeling simulation as described in Section 4.

Acknowledgements

The work presented in this paper has been supported by the Army Research Office (ARO) through Grant Number W911NF0510411 entitled "Adaptive Multiscale Analysis of Deposition Processes" and through grants from Simmetrix Inc. as parts of joint STTR projects supported by the Army Research Laboratories (through ARO) and the Navy (NAVAIR).

References

[1] Dassault Systémes, Abaqus documentation. <http://www.simulia.com/support/documentation.html>.
[2] M. Ainsworth, J. Oden, A Posteriori Error Estimation in Finite Element Analysis, John Wiley & Sons, Inc., 2000.
[3] I. Babuska, T. Strouboulis, The Reliability of the FE Method, Oxford Press, 2001.
[4] P.T. Bauman, J.T. Oden, S. Prudhomme, Adaptive multiscale modeling of polymeric materials with Arlequin coupling and Goals algorithms, Comp. Meth. Appl. Mech. Engrg. 198 (5–8) (2009) 799–818.
[5] M.W. Beall, An Object-oriented Framework for the Reliable Automated Solution of Problems in Mathematical Physics, Ph.D. Thesis, Rensselaer Polytechnic Institute, 1999.
[6] M.W. Beall, J. Walsh, M.S. Shephard, Accessing cad geometry for mesh generation, in: Proceedings of the 12th International Meshing Roundtable, Sandia National Laboratories, 2003, pp. 2003–3030.
[7] M. Beall, J. Walsh, M. Shephard, A comparison of techniques for geometry access related to mesh generation, Engrg. Comput. 20 (2004) 210–221.
[8] T. Belytschko, S. Xiao, Coupling methods for continuum model with molecular model, Int. J. Multiscale Comput. Engrg. 1 (1) (2003) 115–126.
[9] B. Bollobas, Modern Graph Theory, Springer-Verlag, New York, NY, 1998.
[10] I. Bratko, Prolog Programming for Artificial Intelligence, first ed., Addison-Wesley, Reading, Massachusetts, 1986.
[11] W.F. Bronsvoort, F.W. Jansen, Feature modelling and conversion – key concepts for concurrent engineering, Comput. Ind. 21(1) 61–86.
[12] Cactus Home Page. <http://www.cactuscode.org/>.
[13] Comsol Multiphysics Home Page. <http://www.comsol.com/>.
[14] P. De Ceuninck, T. Quintino, S. Vandewalle, H. Deconinck, Object-oriented framework for multi-method parallel PDE software, in: Proceedings of European Conference on Object-Oriented Computing (ECOOP), Darmstadt, Germany, 2003.
[15] P.L. Chandran, V. H Barocas, Deterministic material-based averaging theory model of collagen gel mechanics, J. Biomech. 129 (2) (2007) 137–147.
[16] W.F. Clocksin, C.S. Mellish, Programming in PROLOG, Springer-Verlag, New York, 1981.
[17] D. Datta, C. Picu, M. Shephard, Composite grid atomistic continuum method: an adaptive approach to bridge continuum with atomistic analysis, Int. J. Multiscale Comput. Engrg. 2 (2004) 401–420.
[18] R. Diestel, Graph Theory, Springer-Verlag, Berlin Heidelberg, 2006.
[19] W. E, B. Engquist, Z. Huang, Heterogeneous multiscale method: a general methodology for multiscale modeling, Phys. Rev. B 67 (092101) (2003) 1–4.
[20] J. Fish, M. Nuggehally, M. Shephard, C. Picu, S. Badia, M. Parks, M. Gunzburger, Concurrent atc coupling based on a blend of the continuum stress and the atomistic force, Comput. Methods Appl. Mech. Engrg. 196 (2007) 4548–4560.
[21] Thomas Heller, The Ctypes Package. <http://python.net/crew/theller/ctypes/>.
[22] C.M. Hoffmann, R. Joan-Arinyo, Distributed maintenance of multiple product views, Comput. Aided Des. 32 (7) (2000) 421–431. URL <http://dx.doi.org/10.1016/S0010-4485(00)00023-3>.
[23] International Standards for Business, Government and Society, ISO/IEC 13211-1:1995 Prolog–Part 1: General Core. <http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=21413>.
[24] W. Joppich, M. Krschner, MpCCI – a tool for the simulation of coupled applications, Concurrency Comput. Pract. Experience 18 (2) (2006) 183–192.
[25] Brian W. Kernighan, Dennis M. Ritchie, The C Programming Language: ANSI C Version, Prentice Hall, 1988.
[26] I. Kevrekidis, C. Gear, J. Hyman, P. Kevrekidis, O. Runborg, C. Theodoropoulos, Equation-free coarse–grained multiscale computation: enabling microscopic simulators to perform system-level tasks, Commun. Math. Sci. 1 (4) (2003) 715–762.
[27] J. Knap, M. Ortiz, An analysis of the quasicontinuum method, J. Mech. Phys. Solids 49 (2001) 1899–1923.
[28] V.G. Kouznetsova, Computational Homogenization for the Multi-scale Analysis of Multi-phase Materials, Ph.D. Thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, 2002.
[29] H.P. Langtangen, Python Scripting for Computational Science, Springer-Verlag Series in Computational Science and Engineering, 2004.

[30] X.-J. Luo, T. Stylianopoulos, V. Barocas, M. Shephard, Multiscale computation for bioartificial soft tissues with complex geometries, Engrg. Comput. 25 (2008) 87–95.
[31] A. Martelli, Python in a Nutshell, O Reilly Media Inc., 2003.
[32] Michael Metcalf, John Reid, Malcolm Cohen, Fortran 95/2003 Explained (Numerical Mathematics and Scientific Computation), Oxford University Press, USA, 2004.
[33] R. Miller, E. Tadmor, The quasicontinuum method: overview, applications and current directions, J. Comput. Aided Mater. Des. 9 (3) (2002) 203–239.
[34] R. Miller, A. Acharya, A stress–gradient based criterion for dislocation nucleation in crystals, J. Mech. Phys. Solids 52 (2004) 1507–1525.
[35] A. Noort, G.F.M. Hoek, W.F. Bronsvoort, Integrating part and assembly modelling, Comput. Aided Des. 34 (12) (2002) 899–912. URL <http://dx.doi.org/10.1016/S0010-4485(01)00145-2>.
[36] M. Nuggehally, M. Shephard, C. Picu, J. Fish, Adaptive model selection procedure for concurrent multiscale problems, Int. J. Multiscale Comput. Engrg. 5 (5) (2007) 369–386.
[37] M. Nuggehally, Concurrent Atomistic to Continuum Coupling and Adaptive Model Selection for Multiscale Problems, Ph.D. Thesis, Rensselaer Polytechnic Institute, 2007.
[38] R. O'Bara, M. Beall, M. Shephard, Attribute management system for engineering analysis, Engrg. Comput. 18 (2002) 339–351.
[39] J.T. Oden, K. Vemaganti, N. Moës, Hierarchical modeling of heterogeneous solids, Comput. Methods Appl. Mech. Engrg. 172 (1–4) (1999) 3–25.
[40] J. Oden, S. Prudhomme, Goal-oriented error estimation and adaptivity for the finite element method, Comp. Meth. Appl. Mech. Engrg. 41 (5–6) (2001) 735–756.
[41] J.T. Oden, S. Prudhomme, Estimation of modeling error in computational mechanics, J. Comput. Phys. 182 (2002) 496–515.
[42] J.T. Oden, S. Prudhomme, P. Bauman, On the extension of goal-oriented error estimation and hierarchical modeling to discrete lattice models, Comput. Methods Appl. Mech. Engrg. 194 (2005) 3668–3688.
[43] J.T. Oden, K.S. Vemaganti, Estimation of Local Modeling Error and Goal-oriented Adaptive Modeling of Heterogeneous Materials; Part I: Error Estimates and Adaptive Algorithms, Technical Report. URL <http://citeseer.ist.psu.edu/426058.html>; <http://www.ticam.utexas.edu/reports/2000/0001.ps.gz>.
[44] S.J. Plimpton, Fast parallel algorithms for short-range molecular dynamics, J. Comp. Phys. 117 (1995) 1–19.
[45] S. Plimpton, R. Pollock, M. Stevens, Particle-mesh ewald and rRESPA for parallel molecular dynamics simulations, in: PPSC, SIAM, 1997.
[46] Python Software Foundation, Python Programming Language-Official Website. <http://www.python.org/>.
[47] Salome Home Page. <http://www.salome-platform.org/home/presentation/overview/>.
[48] E.S. Seol, FMDB: flexible Distributed Mesh Database for Parallel Automated Adaptive Analysis, Ph.D. Thesis, Rensselaer Polytechnic Institute, 2005.
[49] E. Seol, M. Shephard, Efficient distributed mesh data structure for parallel automated adaptive analysis, Engrg. Comput. 22 (3–4) (2006) 197–213.
[50] M. Shephard, M. Nuggehally, B. FranzDale, C. Picu, J. Fish, Component Software for Multiscale Simulation, Rensselaer Polytechnic Institute.
[51] M. Shephard, M. Beall, R. O'Bara, B. Webster, Toward simulation-based design, Finite Elem. Anal. Des. 40 (12) (2004) 1575–1598.
[52] M. Shephard, E. Seol, B. FrantzDale, Toward a multi-model hierarchy to support multiscale simulation, in: P. Fishwick (Ed.), CRC Handbook of Dynamic System Modeling, Chapman and Hall, Boca Raton, 2007, pp. 12.1–12.18.
[53] M. Shephard, E.S. Seol, Flexible distributed mesh data structure for parallel adaptive analysis, Adv. Comput. Infrastruct. Parallel Distrib. Adapt. Appl. (2007) 1–38.
[54] J.R. Stewart, H.C. Edwards, A framework approach for developing parallel adaptive multiphysics applications, Finite Elem. Anal. Des. 40 (12) (2004) 1599–1617.
[55] Simmetrix Home Page. <http://www.simmetrix.com/>.
[56] Simmetrix, Inc., Abstract Model. <http://www.simmetrix.com/products/SimulationModelingSuite/GeomSimAbstract/GeomSimAbstract.html>.
[57] Simmetrix, Inc. Fieldsim. <http://www.simmetrix.com/products/Simulation-ModelingSuite/FieldSim/FieldSim.html>.
[58] Simmetrix, Inc. Geomsim. <http://www.simmetrix.com/products/Simulation-ModelingSuite/GeomSim/GeomSim.html>.
[59] Simmetrix, Inc. Meshsim. <http://www.simmetrix.com/products/Simulation-ModelingSuite/MeshSim/MeshSim.html>.
[60] L. Sterling, E. Shapiro, The Art of Prolog, MIT Press, The Art of Prolog, MIT Press, Cambridge, Massachusetts, 1986.
[61] B. Stroustrup et al., The C++ Programming Language, Addison-Wesley, Reading, MA, 1997.
[62] T. Stylianopoulos, C.A. Bashur, A.S. Goldstein, S.A. Guicher, V.H. Barocas, Computational predictions of the tensile properties of electrospun fiber meshes: effect of fiber diameter and fiber orientation, J. Mech. Behavior Biomed. Mater. 1 (2008) 326–335.
[63] D.B. West, Introduction to Graph Theory, second ed., Prentice Hall, 2001.
[64] C.H. Whiting, Stabilized Finite Element Methods for Fluid Dynamics Using a Hierarchic Basis, Ph.D. Thesis, Rensselaer Polytechnic University, 1999.
[65] J. Wielemaker, An overview of the SWI-Prolog programming environment, in: F. Mesnard, A. Serebenik (Eds.), Proceedings of the 13th International Workshop on Logic Programming Environments, Katholieke Universiteit Leuven, Heverlee, Belgium, 2003, pp. 1–16, cW 371.
[66] J. Wielemaker, SWI-Prolog 5.6.69 Reference Manual, Human-Computer Studies, University of Amsterdam, 2008.
[67] O.C. Zienkiewicz, J.Z. Zhu, The superconvergent patch recovery and a posteriori error estimates. part 1: the recovery technique, Int. J. Numer. Methods Engrg. 33 (1992) 1331–1364.
[68] H. Zhong, M.P. Wachowiak, T.M. Peters, A real time finite element based tissue simulation method incorporating nonlinear elastic behavior, Comput. Methods Biomech. Biomed. Engrg. 8 (3) (2005) 177–189.