

Designing a Component-Based Architecture for the Modeling and Simulation of Nuclear Fuels and Reactors

[Extended Abstract]

Jay J. Billings
Wael R. Elwasif
Lee M. Hively
David E. Bernholdt
Oak Ridge National
Laboratory
P.O. Box 2008
Oak Ridge, TN 37831
{billingsjj,elwasifwr,hivelylm,
bernholdtde}@ornl.gov

John M. Hetrick III
IBM
8401 Greensboro Drive Suite
120
McLean, VA 22102
hetrick@us.ibm.com

Tim Bohn
IBM
5744 Nutwood Circle
Simi Valley, CA 93063
tbohn@us.ibm.com

ABSTRACT

Concerns over the environment and energy security have recently prompted renewed interest in the U. S. in nuclear energy. Recognizing this, the U. S. Dept. of Energy has launched an initiative to revamp and modernize the role that modeling and simulation plays in the development and operation of nuclear facilities. This Nuclear Energy Advanced Modeling and Simulation (NEAMS) program represents a major investment in the development of new software, with one or more large multi-scale multi-physics capabilities in each of four technical areas associated with the nuclear fuel cycle, as well as additional supporting developments. In conjunction with this, we are designing a software architecture, computational environment, and component framework to integrate the NEAMS technical capabilities and make them more accessible to users. In this report of work very much in progress, we lay out the “problem” we are addressing, describe the model-driven system design approach we are using, and compare them with several large-scale technical software initiatives from the past. We discuss how component technology may be uniquely positioned to address the software integration challenges of the NEAMS program, outline the capabilities planned for the NEAMS computational environment and framework, and describe some initial prototyping activities.

Categories and Subject Descriptors

D.2.11 [Software Engineering]: Software Architectures; I.6.3 [Simulation and Modeling]: Applications; J.2 [Computer Applications]: Physical Sciences And Engineering—Engineering

1. INTRODUCTION

Nuclear power accounts for approximately 20% of U. S. electricity and offers a safe, reliable, and environmentally friendly alternative to the use of fossil fuels. This, along with increasing global demand and security concerns associated with fossil fuels, has resulted in a resurgence of both public and private interest in nuclear energy in the U. S. Applications for 26 new reactors have been submitted to the U. S. Nuclear Regulatory Commission (NRC) since 2007 alone. This interest, however, comes after a lengthy period in which the U. S. developed no new reactors. As a consequence, the level of modeling and simulation used in the nuclear energy industry had largely stagnated. In light of the recent increased interest, the U. S. Department of Energy created the Nuclear Energy Advanced Modeling and Simulation (NEAMS) program to bring to the nuclear energy industry the high-end modeling and simulation (M&S) capabilities that are now the state of the art in other areas of science and engineering. The goal is to use high-end M&S to accelerate development and deployment of new technologies while while increasing safety and efficiency and lowering costs.

The NEAMS program is organized around four technical areas of the nuclear fuel cycle: fuels, reactors, separations and safeguards, and wastefoms. Additional cross-cutting activities support these areas: fundamental methods and models (FMM), validation and verification and uncertainty quantification, enabling computational technologies, and capabilities transfer. Each of the technical areas, as well as FMM involve the development of one or more large-scale modeling package, each at a scale comparable to that of a large modern computational science project, such as those supported by the Dept. of Energy’s Scientific Discovery through Advanced Computing (SciDAC) initiative [2]. As part of the NEAMS vision, an overarching software architecture and framework is also being developed, with the goal of facilitating software integration across the program and deployment to a broad spectrum of users ranging from national laboratory and academic researchers to nuclear technology vendors, powerplant operators, and regulators.

This report describes the approach being taken to develop this system (Sec. 2, and offers a snapshot of the progress to date, (roughly 9 months into the process) on the design of the NEAMS Computational Environment (Sec. 3) and the NEAMS Framework (Sec. 4). The report closes with a discussion of related work (Sec. 5) and a summary (Sec. 6).

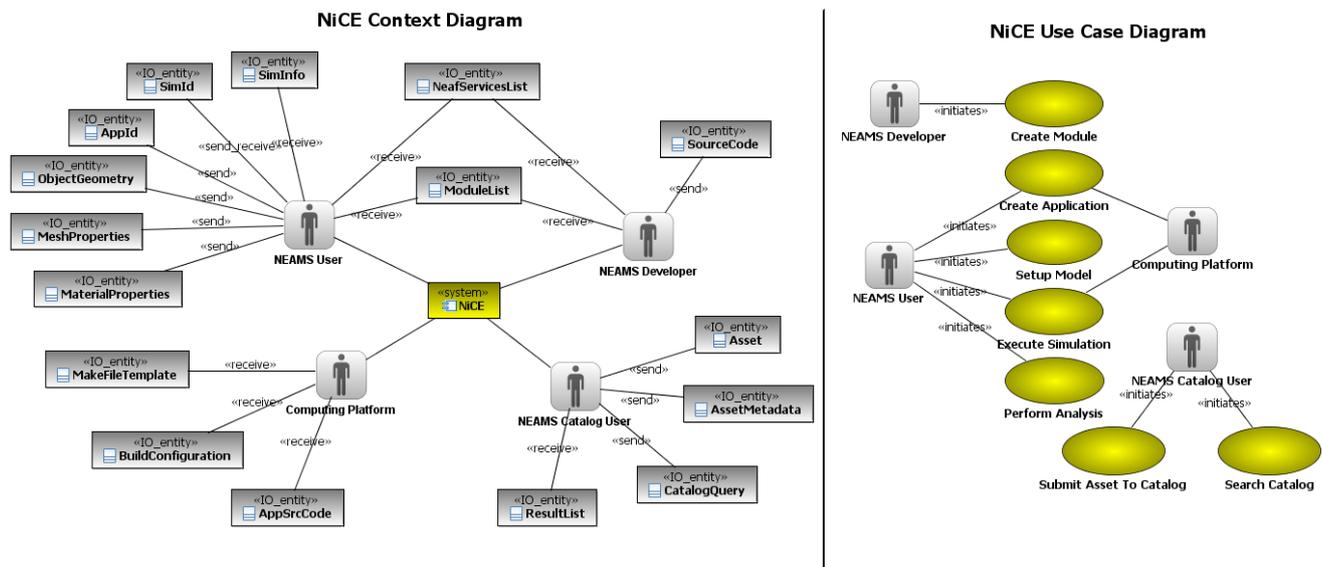


Figure 1: A UML context diagram for NiCE showing the actors and information passed to and from the system (left), while a UML use case diagram showing how the actors use the system (right).

2. DESIGN PROCESS

Our design process follows a use case driven, model based approach. The method is specifically designed to apply development best practices to large and/or complex problems. The approach is prescriptive and provides a rigorous approach to software and systems development. While aspects of this process will be familiar to many who have experience developing large software systems, our experience with many previous computational science and computer science software development projects is that the design process is often rather ad hoc, and not very systematic. Given the scale of the NEAMS effort, our desire from the start was to pursue a more systematic, rigorous, and carefully documented approach in order to develop a design that is both scalable and sustainable in the long term (10+ years).

The design process structures the work into a flow that provides for consistency from problem identification, to problem analysis and specification, to solution design, and finally solution implementation. The method takes a top down approach while considering bottom up constraints. There is a flow to the method from outside to inside recognizing discrete levels of abstraction.

The system is initially presented as single “black box.” The work initially focuses on understanding and specifying information passing in and out of the system. Once the black box flow of information is understood, the internal workings of the system can be explored. This internal view is called the “white box.”

2.1 Black Box

Bounding and scoping are represented by communicating the context of the system (Fig. 1). Context isolates the system as a single black box and describes who or what interacts with the system (actors). Not only are a comprehensive set of actors identified, but also what is exchanged between the actors and the system under development.

Use cases describe the value that actors expect to get out of the system. In other words, the use cases identify what the actors want the system to do for them, or help them do. This identifies the behavior that the system must provide to be successful. At a glance the identified use cases and their description provide a resume of

what the system will and will not do. Note that this does not indicate the level of complexity, nor should it.

The specification of the black box, i.e. the creation and passage of information to the system, is a description of the NEAMS Integrated Computational Environment (NiCE).

2.2 White Box

After the black box, the design process focuses on how the system will work. Scenarios derived from the use cases continue to drive the process. These scenarios are explored using concrete examples from the NEAMS fuels and reactors teams. There is an m to n relationship between the scenarios and the examples applied to the scenarios. Each example contributes to fleshing out new aspects of the scenarios. Thus, the solution grows and solidifies based on the actual problems the system is being built to solve. This also provides a rigorous and controlled approach to designing the system.

The set of components that comprise the system are identified. Interactions between instances of the components are specified for each scenario and for each example. This yields the relationships between the components and determines what the responsibilities are for each component, which is fundamental for driving implementation.

Beyond simply identifying components and their interaction is effort to recognize common situations that could be and should be addressed in standard ways. For instance recognizing that application modules can be dealt with in a uniform manner by a driver, or having a standard way for modules written in different languages to interoperate. These standard solutions are often called architectural or design mechanisms and provide the foundations for extensible architectures yielding robust implementations. These architectural mechanisms, often themselves manifested as frameworks, are core aspects of the NEAMS Framework.

In the following sections, we outline some of the major considerations and design features that have emerged from our work so far, as well as describing additional aspects of the design process.

3. NEAMS INTEGRATED COMPUTATIONAL ENVIRONMENT

The NEAMS Integrated Computational Environment (NiCE) is intended to allow flexible development and composition of a wide range of simulations and applications. Anticipated NEAMS applications include integrated performance and safety codes (IPSCs) for reactors, including fuel performance, core, and balance of plant modeling; as well as fundamental methods and models (FMMs), smaller length scale material modeling work, and atomistic to continuum multi-scale simulation, which provide understanding and improve properties and models for the integrated codes. The environment must also be flexible and extensible in order to accommodate new simulation capabilities that were not necessarily planned when the environment was initially specified and constructed.

NiCE will support both the development of new software modules and the incorporation of existing software into a common environment to promote interoperability. The Computational Environment will support the composition of simulation applications from software modules external to the Environment and utilities internal to the Environment, setup of models, execution of simulations on those models, and analysis of the results. In addition, the Computational Environment will support the NEAMS program's needs for verification, validation, and uncertainty quantification of both individual modules, and composite applications. NiCE will provide Enabling Computational Technologies (ECTs) in the context of a NEAMS Architecture and Framework. ECTs are capabilities of widespread utility, which facilitate and simplify the development and integration of software modules (though developers are free to use alternatives of their own if desired). Types of utilities likely to be of interest to NEAMS include, for example, unstructured meshes, solvers, data format readers and translators, visualization and analysis tools, and data management systems.

Use cases (right side of Fig. 1) have played a central role in “understanding” the NEAMS Computational Environment. Although their names, and even their descriptions may sound generic to virtually any simulation application, when developed carefully and in detail, following best practices [6] provide a wealth of detail unique to the target system and its users. Briefly, some of those key use cases in our design work are:

Creating Modules and Applications. NiCE provides a way for users and developers to work with modules and makes it possible to create or edit them in a user-friendly way. In this case, “modules” are components that provide some unit of functionality.

NiCE also provides tools for creating or editing applications. Applications are a specific set of modules, combined with any needed “infrastructure” modules from the framework, and with all connections between components specified.

Creating Models: Geometries, Meshes, and Materials. Creating input information, i.e. models of physical systems, can be a difficult process. NiCE makes this task easier for users by providing a comprehensive set of geometry, meshing, and materials tools. Additionally, NiCE will address the issue of coupling these three types of information together.

Executing Simulations. A fundamental aspect of any computational environment is the ability to execute simulations. Simulations represent the combination of an application with a specific model. NiCE provides a platform-independent execution scheme for compiling and running simulations on local or remote machines.

Performing Analyses. Many users will inevitably desire to perform some sort of analysis on the data produced by their simulations. NiCE provides a set of built-in analysis operations to help users with this task. NiCE also provides the ability to extend the

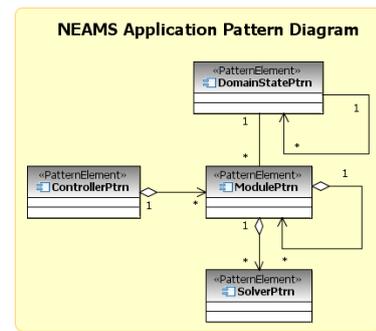


Figure 2: A UML class diagram depicting a pattern for creating applications.

analysis functionality by facilitating the creation of new analysis operations.

4. NEAMS FRAMEWORK

The NEAMS Framework is a specification, middleware, and associated tools that allow the realization of NEAMS applications in a flexible way through composition of independently-written modules and utilities in a common environment. Furthermore, the NEAMS Framework will also classify and generally describe the types of modules anticipated as part of the NEAMS program and, where possible, will define interfaces representing the preferred way to access the functionality of such capabilities.

4.1 Language Interoperability and Platform Portability

Problems with language interoperability are common, especially in the scientific community. The design of the NEAMS Framework is language agnostic at a high level and drives down to language specifics. The goal of this approach is to let users ignore language issues; they can create modules in the language of their choice and worry only about their research.

The NEAMS Framework takes a similar approach to platform portability. The diverse nature of computing platforms across the nuclear energy community requires a system that is nimble enough to deal many different hardware platforms and operating systems. This includes everything from laptops to petascale supercomputers, and beyond. Considering these platform differences at the highest level of the design process will insure that every user and developer can perform meaningful research, regardless of their computing resources.

4.2 Modules and Services

The NEAMS Framework makes no distinction between modules and services at this point. While it may be the case that certain “infrastructure” modules are created and delivered as part of the framework, there is no difference between these modules and those created by other developers. In fact, the infrastructure modules could be replaced with other modules so long as the functionality remained.

This view of modules is in contrast to many other large-scale scientific frameworks. However, this approach will allow the functionality of the NEAMS Framework to change in tandem to the research efforts and insure that the NEAMS Framework is enabling rather than constraining to developers.

Fig. 2 illustrates a pattern for creating applications from modules. In this pattern, the user controls a set of modules with a driver or “controller” component and the modules store information in

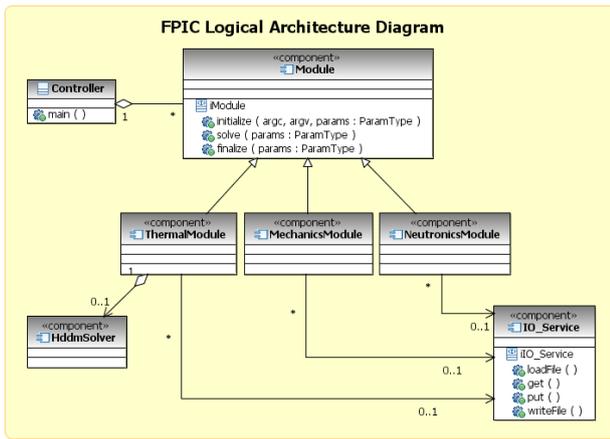


Figure 3: A UML class diagram depicting the logical architecture for the NEAMS Fuels Performance Integrated Code prototype.

a “domain state” that is common between them, (note the one to many relationship in the figure). Finally, the relationship of modules to other modules and math solvers is shown.

4.3 Software Prototypes

Part of the white box design effort of the NEAMS Framework has centered on exploring physical models provided by the NEAMS fuels and reactor teams. These examples were explored with prototype implementations of the NEAMS Framework. The prototypes combine various elements of the ADVENTURE suite, [1], as depicted in Fig. 3.

Two prototype implementations exist, one in C and Python and another created with the Common Component Architecture (CCA) [5]. The C/Python implementation provides immediate access to a large number of different packages for testing while the CCA implementation helps in the exploration of language interoperability issues, amongst other things.

5. COMPARISON WITH OTHER FRAMEWORKS

Many other software frameworks exist in computational science and engineering, though few at the scale envisioned for NEAMS. Several among them have been influential in our thinking, and are being carefully evaluated in the context of the NEAMS design.

The Common Component Architecture (CCA) is a component architecture designed for high-performance computational science and engineering, [5]. The CCA specification is maintained by the CCA Forum, an open community organization, based largely in the computer and computational science research community.

SALOME 5 is a generic platform for pre- and post-processing for numerical simulation, [4]. Salome provides a link between CAD modeling and simulation software. There are also components available or being integrated into Salome for solid mechanics, neutronics, and thermal-hydraulics, among others. It is developed by a group in Électricité de France (EDF) R&D.

SWIM Integrated Plasma Simulator (SWIM IPS) is a component framework aimed at “whole-device modeling” of fusion reactors, [7]. The design of SWIM IPS is conceptually based on the CCA, but the implementation is more narrowly tailored to the project’s needs.

Simulation-based High-efficiency Advanced Reactor Prototyp-

ing (SHARP) is a suite of codes for fast reactor simulations. SHARP is designed around a mesh and targets BG/P and Cray supercomputers, [8].

SCALE (Standardized Computer Analyses for Licensing Evaluation) is a modular code system that was developed by Oak Ridge National Laboratory for the NRC, [3]. System development has been directed at problem-dependent cross-section processing and analysis of criticality safety, shielding, depletion/decay, and reactor physics problems. SCALE has been widely used for evaluation of nuclear fuel facility and package designs.

6. SUMMARY

We have presented preliminary design work on the NEAMS Integrated Computational Environment and the NEAMS Framework. We have also described our design process and provided a list of other frameworks which are relevant to this work.

7. ACKNOWLEDGMENTS

We are particularly grateful to Kevin Clarno, Scott Mosher, Sreekanth Pannala, Srdjan Simunovic, Tim Tautges, and John Turner for many helpful discussions during the course of this work.

This work has been supported by the U. S. Department of Energy, Offices of Nuclear Energy and by the ORNL Postmasters Research Participation Program which is sponsored by ORNL and administered jointly by ORNL and by the Oak Ridge Institute for Science and Education (ORISE). ORNL is managed by UT-Battelle, LLC for the U. S. Department of Energy under Contract No. DE-AC05-00OR22725. ORISE is managed by Oak Ridge Associated Universities for the U. S. Department of Energy under Contract No. DE-AC05-00OR22750.

8. REFERENCES

- [1] The ADVENTURE project. <http://adventure.sys.t.u-tokyo.ac.jp/>.
- [2] SciDAC - DOE’s Scientific Discovery through Advanced Computing. <http://www.scidac.gov>.
- [3] SCALE: A Modular Code System for Performing Standardized Computer Analyses for Licensing Evaluation. *ORNL/TM-2005/39*, I-III, November 2006.
- [4] The SALOME Platform, Summer 2009. <http://www.salome-platform.org>.
- [5] B. A. Allan, R. Armstrong, D. E. Bernholdt, F. Bertrand, K. Chiu, T. L. Dahlgren, K. Damevski, W. R. Elwasif, T. G. W. Epperly, M. Govindaraju, D. S. Katz, J. A. Kohl, M. Krishnan, G. Kurfert, J. W. Larson, S. Lefantzi, M. J. Lewis, A. D. Malony, L. C. McInnes, J. Nieplocha, B. Norris, S. G. Parker, J. Ray, S. Shende, T. L. Windus, and S. Zhou. A component architecture for high-performance scientific computing. *Intl. J. High-Perf. Computing Appl.*, 20(2):163–202, Summer 2006.
- [6] K. Bittner and I. Spence. *Use Case Modeling*. Addison-Wesley, Boston, 2003.
- [7] W. R. Elwasif, D. E. Bernholdt, L. A. Berry, and D. B. Batchelor. Component framework for coupled integrated fusion plasma simulation. In *HPC-GECO/CompFrame 2007, 21–22 October, Montreal, Quebec, Canada*, 21–22 October 2007.
- [8] A. Siegal et al. Software Design of SHARP. *Joint International Topical Meeting on Mathematics & Computational and Supercomputing in Nuclear Applications*, 2007.