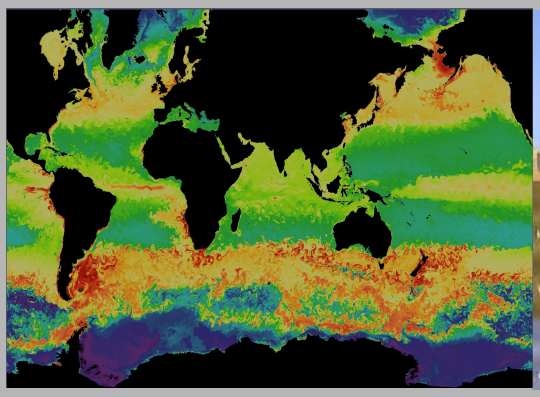
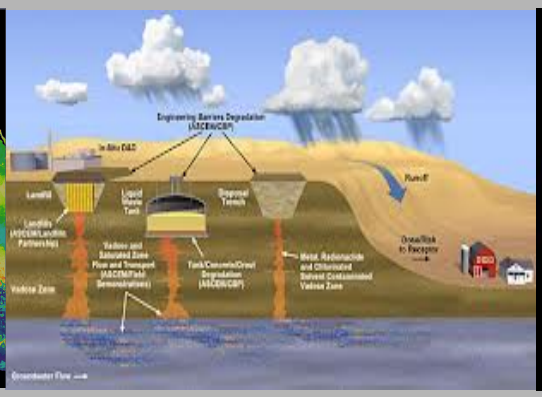
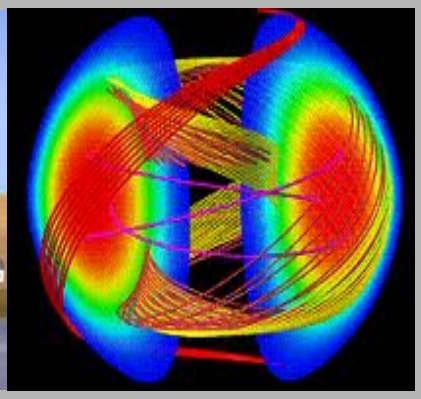

# Extreme-Scale Scientific Application Software Productivity:

## Harnessing the Full Capability of Extreme-Scale Computing

September 9, 2013

Hans Johansen (LBNL), David E. Bernholdt (ORNL), Bill Collins (LBNL),
Michael Heroux (SNL), Robert Jacob (ANL), Phil Jones (LANL),
Lois Curfman McInnes (ANL), J. David Moulton (LANL),
Thomas Ndousse-Fetter (DOE/ASCR), Douglass Post (DOD), William Tang (PPPL)

**EXECUTIVE SUMMARY** ..... **3**

**1 INTRODUCTION** ..... **4**

**2 SCIENTIFIC APPLICATION SOFTWARE CHALLENGES AT EXTREME SCALE** ..... **5**

2.1 COMMERCIAL INNOVATION IN SOFTWARE PRODUCTIVITY ..... 6

2.2 TOWARD SOFTWARE ENGINEERING FOR COMPUTATIONAL SCIENCE ..... 7

**3 CASE STUDIES: SOFTWARE PRODUCTIVITY IN APPLICATIONS** ..... **8**

**4 PATH FORWARD FOR COMPUTATIONAL SCIENCE SOFTWARE** ..... **10**

4.1 EARLY EXPERIENCES FROM NUMERICAL LIBRARIES AND PROTOTYPE APPLICATIONS ..... 10

4.2 IMPLICATIONS FOR THE FUTURE ..... 11

**5 ADDRESSING APPLICATION SOFTWARE PRODUCTIVITY CHALLENGES IN THE OFFICE OF SCIENCE: RECOMMENDATIONS** ..... **11**

**APPENDIX A: APPLICATION SOFTWARE PRODUCTIVITY CASE STUDIES** ..... **13**

A.1 DOE CLIMATE MODELING SOFTWARE INFRASTRUCTURE ..... 13

A.2 DOE ENVIRONMENTAL MANAGEMENT SOFTWARE DEVELOPMENT ..... 15

A.3 APPLICATION SOFTWARE PRODUCTIVITY FOR DOE FUSION ENERGY SCIENCES ..... 17

**REFERENCES:** ..... **19**

## Executive Summary

Computational science is approaching a crisis. Mission-critical research on scientific grand challenge problems has become increasingly dependent on high-performance computing, and yet the *productivity of extreme-scale scientific application development* has lagged. The reason for this growing gap is clear: unique and dramatic architectural changes in extreme-scale scientific computing platforms (such as those in DOE's leadership computing facilities) require more sophisticated algorithmic and computer science techniques, while complex scientific application teams targeting these platforms lack the software development tools, libraries, and methodology needed to be most productive. Within DOE, there are compelling examples of this impending crisis and how it is starting to be addressed, but a broader study is needed to generate a strategic vision for research on software productivity for extreme-scale science, including how to exploit the unique features of DOE mathematical software and codes simulating physical systems.

This whitepaper presents the background and arguments for investing in this critical gap, with the goal of improving DOE's scientific and computing productivity. By investing in research to better support collaborative development of long-lived computational software products, mission-focused scientific research will be more effective and productive.

# 1 Introduction

Extreme-scale computing platforms are increasingly becoming the *de facto* computational systems for grand challenge problems in the DOE scientific community. With these systems, a diverse collection of multiscale and multiphysics applications has evolved to answer many grand challenge questions. Mission-critical efforts in climate prediction, environmental management, fusion energy, and materials science, among others, are poised for breakthroughs with distributed, multidisciplinary teams, each specializing in particular aspects of modeling and simulation, using DOE's extreme-scale platforms and leadership computing facilities (LCFs). The extraordinary capabilities of these systems enable scientists to resolve more scales, sample larger ensembles, and support the coupling of multiple physical phenomena not possible on lesser computing platforms [7,8,18].

Nevertheless, despite the unprecedented potential of these systems, studies consistently point to the need for *better scientific application software development processes* as a key enabler for fully exploiting these platforms for scientific advancement [3,6,7,9,12,13,14,15,17,18,19]. Frequent failure modes include the following:

1. Increasing lag between extreme-scale hardware and algorithmic innovations and their effective use in applications, leading to ***poorly scaling codes and loss of LCF productivity***.
2. ***Lack of agile yet rigorous software engineering practices for high-performance computational science***, thus preventing distributed teams from fully exploiting extreme-scale platform capabilities.
3. Failure to consider the entire lifecycle of large scientific software efforts, leading to ***fragile, complex scientific applications that have become increasingly difficult to enhance***.

These barriers to software productivity, which are hampering progress in critical scientific application development, hinder DOE's ability to achieve its mission goals through computational science. The challenges are perhaps most visible in areas where DOE investment has focused on the development of one or a handful of large code-bases that are shared across entire research communities. In such cases, the **large scientific software systems should be thought of as long-term software investments** that provide key infrastructure for a broad range of research aligned with DOE's mission. By analogy to experimental user facilities, such resources require professional planning, design, staffing, and upgrades to maximize their useful life and ability to serve their user base.

The confluence of new manycore and accelerator-based architectures with millions of cores, radical new programming models, and a new generation of multiphysics applications sets the stage for an inevitable crisis in application software that is likely to present a significant impediment to realizing the promise of extreme-scale scientific computing applied to DOE's grand science challenges. We recommend

that DOE ***invests in research on scientific software productivity and capability*** in order to remove these barriers and create robust software development infrastructure and techniques.

## 2 Scientific Application Software Challenges at Extreme Scale

As computing architectures, programming paradigms, and computational models become more complicated, software development has become a major obstacle for large-scale computational science projects. Key challenges include the following:

- **Inherent complexity of today's sophisticated modeling and simulation codes.** As capabilities of large-scale computers have increased, our computational models have grown similarly—transitioning from qualitative to quantitative, with increasing ability to accurately simulate physical phenomena and to predict experimental observations. The goals and methods of computational science software have long posed a challenge for traditional software development methodologies, created in very different contexts. The size, complexity, and pace of change of modern computational science applications exacerbate this challenge.
- **Extensive parallelism, billion-way concurrency, and uncertainty in extreme-scale multicore systems**. Emerging from a lengthy period of relative stability in computer architectures, we now face a period of significant flux. New and emerging computing platforms are characterized by myriad novel architectural features, including millions of cores, simultaneous multithreading, vectorization, core heterogeneity, unconventional memory hierarchies, and new programming models that complicate software development. Additionally, many domain science communities such as climate, environmental science, fusion, and materials science have large legacy code bases for conventional computing systems that will face additional challenges of porting and refactoring to work efficiently on extreme-scale systems.
- **Complex coupling of multiscale and multiphysics component algorithms.** Enabled by rapid advances in computational power, and driven by the need for higher levels of physical fidelity, multiscale and multiphysics simulations play an increasingly important role. Applications need common interfaces to facilitate interoperability and must include well-defined code coupling tools and methodologies to integrate subcomponents into complete applications, while supporting the necessary explorations of the mathematics of the couplings.
- **Development of stable library interfaces.** As intermediate software layers between lower-level programming environments and higher-level application codes, libraries provide key functionalities to applications and insulate users from many architectural complexities. While some approaches to enhance application-level productivity will also address challenges arising in library development, these intermediate layers of the software stack face unique difficulties, requiring the ability to incorporate changes and new features over time yet provide stable user interfaces . Additional productivity challenges arise for lower-level tools, including profilers and runtime application support.

## 2.1 Commercial Innovation in Software Productivity

Software engineering, defined as the application of a systematic, disciplined, repeatable, and testable approach to the design, development, operation, and maintenance of software, was spurred by the software crisis a few decades ago [5]. Since that time, the broader software industry has made significant advances in understanding and addressing the challenges of the development of complex, large-scale software systems. Examples of iterative software development methodologies include (1) *unified processes*, which emphasize iterations for different roles during a software project and show the importance of frequent testing and change management, and (2) *agile development processes*, which emphasize customer-driven product iterations, with open visibility into development effort, continuous testing, and immediate feedback.

Among the best practices identified are the following:

- **Component or modular software** – a software development construct that enables a large code to be decomposed into manageable modules, packages, or containers to facilitate software interchangeability, reusability, and testing. Examples include aspects of object-oriented design, the open source software ecosystem, and component-based architectures.
- **Software productivity metrics** – a set of metrics (reusable, interoperable, testable, portable, scalable, and easily maintainable) that can greatly improve software productivity. Examples include integrated development environments (IDEs), which have code test coverage metrics in test-driven development, and continuous integration/performance profiling tools. Also included are software design and architecture analysis, ranging from automated static analysis and modeling tools, available in the open source Eclipse framework, to maturity models, such as SEI's Capability Maturity Model Integration (CMMI) and Architectural Tradeoff Analysis Method (ATAM).
- **Software methodology and architecture** – frameworks for planning, management, and control of the software development process. These are primarily iterative software development methodologies that aim to deliver software incrementally and remove risks that contribute to poor quality.
- **Collaborative lifecycle management** – automated tools or computer-aided methodologies to assist the management of people, process, information, governance, maintenance and related tools that drive the software life cycle. Examples include portals tied directly to software product management, such as SourceForge and Atlassian's BitBucket and JIRA, and tools tied to software configuration management through fully integrated IDEs, such as Microsoft VisualStudio and IBM Rational suite, and the open source Eclipse framework.

The lessons learned and best practices observed in successful application of software engineering in other areas of software development provide a good foundation for addressing the emerging challenges of extreme-scale scientific

application software. However, we must also recognize the distinctive nature and needs of computational science software.

## 2.2 Toward Software Engineering for Computational Science

Several factors explain why the field of computational science has lagged in adopting many of the software development innovations that have proved successful in other software areas. One factor is that few computational scientists receive any training in modern software engineering practices. Many eschew software engineering based on experience with practices prevalent much earlier in the field's development, which, in contrast to today's techniques, tended to be rigid and bureaucratic and were not well aligned with typical computational science software development and usage. Further, today's scientific culture, including most funding agencies, tend to view software as the means to an end (new scientific discovery) rather than a scientific instrument that must be carefully engineered, maintained, and extended in order to enable novel science.

Other useful software development concepts that have not been addressed in traditional software engineering but are critical for the computational science community include the following:

- **Software development infrastructure** – a holistic community-based, end-to-end management of services, common standards, workflows, and testing standards to support multiteam software development effort, especially for extreme-scale computing.
- **Software management workflows** – management of collaborative software development and processes to support reproducibility, provenance, and maintainability in large simulation projects.
- **Verification and validation** – application software support for validating models and physical parameterizations, including novel sampling techniques to enhance software testing, verification, and model validation.
- **Agile research software processes** – support for coevolution of software life cycle processes (modeling, design, coding, testing, and executing complex software system) and scientific research goals. See [4,11] for a schematic comparison of validation-centric software projects and phased projects.
- **Group dynamics and management** – understanding and managing the human aspects of software productivity and life cycle in a large-scale multi-disciplinary community code development effort.
- **Legacy code refactoring** – transitioning legacy code and numerical libraries to new programming paradigms and computational science software methodologies for efficient performance on new multicore and hybrid computing platforms.
- **Multiphysics/multiscale component coupling** – interfaces with appropriate levels of encapsulation and granularity that provide stable resources for applications yet enable performance enhancements and exploration of new algorithmic approaches.

Other system management concepts that need further considerations include software component provenance, software integration workflows and metadata, and scalable software system architecture. These concepts have been well-studied and evaluated in commercial software productivity, in frameworks such as the SEI CMMI, and the Predictive Capability Maturity Model (PCMM) [16] for scientific and engineering applications. Wider adoption of these ideas has the potential to greatly improve the productivity and quality of large-scale computational science software projects.

# 3 Case Studies: Software Productivity in Applications

Large-scale multiphysics, multiscale predictive simulations are a critical component of DOE science applications. These simulation codes are developed by large, diverse, and distributed groups of domain scientists, computer scientists, and applied mathematicians. Developers create full applications by connecting components that represent important processes at a variety of spatial and temporal scales, as well as solver capabilities, operators, and other utility functions.

In the appendix we discuss three specific application productivity case studies:

A.1 DOE Climate Modeling Software Infrastructure
A.2 DOE Environmental Management Software Development
A.3 Application Software Productivity for DOE Fusion Energy Sciences

These three applications have been selected as representative examples of a much broader DOE science and modeling portfolio.

The resulting codes in these projects are themselves complex pieces of software that are used to inform decisions and must be trusted by end users. Many of these application codes have developed this trust through a long history of continuous development over decades, but such legacy code presents additional challenges, as described below. Application code complexity and high impact argue for increased attention to software development, software productivity, and software quality. The immediate need for code readiness on new architectures provides an additional impetus for rapid adoption of new processes and techniques in software development. A strong focus on software productivity during this transition will minimize the disruption while providing mechanisms for ensuring software quality during refactoring. A number of challenges impact software productivity from the applications perspective.

**Collaborative distributed development.** As described in the appendices, all of the application codes (an even a broader census is included in [14]) are built by using contributions from large groups of scientists (hundreds in the climate example) across DOE laboratories and academic sites. Effectively harnessing the contributions from a large and distributed team of developers requires

- modern software development processes (e.g., agile, test-driven development);
- distributed development tools, repositories, and processes; and
- interdisciplinary interactions with agreements on how components couple together.

**Legacy code.** In both the climate and fusion applications, expertise and code have been built up over several decades. During previous transitions in computing architecture, these groups adapted algorithms to the new architectures. Now, they must do so again. Refactoring code to take advantage of new hardware and new algorithms while not adversely impacting scientific progress and built-up trust in existing codes requires

- adapting to hardware changes and using advanced algorithms,
- testing to preserve functionality and previous results, and
- communicating with an existing user base that is invested in current approaches.

**Lifecycle.** Large application codes have a lifetime measured in decades. Even relatively new codes, such as those being developed in ASCEM [1], must be designed with this assumption. Designing for a future long life requires

- developing code that can hide architectural changes using abstractions or other means and
- designing flexible code that can more easily incorporate new capabilities and new algorithms.

**Testing, quality assurance, verification and validation.** The three example codes are used to inform energy choices (climate, fusion), cleanup decisions (EM), and the design of large-scale experiments (fusion). Ensuring code correctness and trust in simulation results is critical and requires

- test-driven software development;
- comprehensive testing for multiscale, multiphysics codes using a hierarchy of unit, integration, regression tests; and
- a graded approach for research code or less critical/sensitive components, and for processes that are less well understood.

**Recruiting and training.** All the example applications incorporate large teams of scientists with a wide range of scientific and coding expertise but often with little training in software engineering. It is unreasonable to expect a few professional software engineers to provide this expertise, and often the research-funding paradigm will not support such an investment. These diverse development teams need a consistent view of software development, including

- training of the interdisciplinary team to ensure common practices and a commitment to testing,

- buy-in by all developers so that process and testing continue to be part of the development cycle over time, and
- decentralizing of quality assurance so that all are responsible, rather than a few gatekeepers or software engineers.

A recent report [14] incorporates perspectives of applied mathematicians, computer scientists, and domain scientist from a much broader range of multiphysics applications. While the focus of the report is on algorithmic and software challenges for multiphysics coupling, it emphasizes the universal nature of the issues listed above and strengthens the argument for an increased focus on extreme-scale application software productivity.

# 4 Path Forward for Computational Science Software

The goals we need to accomplish are very challenging, but the seeds of what we need are already planted by a number of DOE projects. We already have sophisticated tool sets and processes, as well as software projects that deal with multiple, modular components and have high standards for software engineering. In addition, we already have efforts to port applications to modern node architectures, with emerging insight into strategies for how new application frameworks should be designed.

The first steps to accomplishing our goals must be to identify, collect, and disseminate our existing knowledge and experiences. Once we have a clear picture of the current state, we can then identify gaps and proceed to fill them.

## 4.1 Early Experiences from Numerical Libraries and Prototype Applications

Although few applications have embarked on the substantial refactoring required to realize scalability on emerging manycore and accelerator platforms, a substantial amount of work has gone into preparing numerical libraries for these platforms. Furthermore, some prototype applications—starting points for future versions of our large-scale applications—have been fundamentally redesigned in numerous ways in order to explore the emerging design space. From these efforts we make the following observations that are likely to be broadly applicable:

1. **The performance-critical parts of existing applications will be displaced or modified:** To scale on emerging platforms, we must exploit vectorization for every important loop set. To have sufficient thread-level parallelism, we must refactor computational functions to have no side effects. These facts affect every key data structure and function in existing applications. With careful planning and design, we can make the transformation cost low per line of code, but the sheer number of lines will make the cost unavoidably high. Promising strategies are already underway in numerical libraries, where separation of the control logic of the algorithms from the computational kernels of the solvers has proven crucial to allow injecting

new hardware-specific computational kernels without having to rewrite the entire solver software stack.

2. **New algorithms will be essential for key computations:** While solvers and other algorithms with global data dependence are often the most computationally intensive parts of high-performance simulations, these algorithms do not always scale well. Recursions, collective operations, and data-driven parallelism pose a significant challenge for current approaches, and new extreme-scale algorithms must be developed.
3. **We are well positioned to exploit the unique features of mathematical software and codes simulating physical systems for extreme-scale scientific software productivity research:** The community is already beginning to address software organization and design challenges through numerical libraries and software tools, which are an essential foundation of many DOE scientific applications. We can further leverage the mathematical nature of DOE codes to facilitate software development and testing.

### 4.2 Implications for the Future

Transformation and expansion of existing applications to fully utilize emerging platforms will require a very large software design and refactoring effort. *Of particular importance is a focus on addressing the unique requirements of large-scale multiphysics and multiscale applications.* Although the extreme-scale computing community has already made strides in understanding some of the challenges we face, the complexity of our algorithms and software brings with it challenges that appear only when combining physics and scales.

We must both leverage the experience and progress made so far, and also make new investments. In particular, we must focus on new algorithms research and development, multiphysics/multiscale prototypes for rapid design space exploration, and thorough planning for the reorganization of existing functionality. Only with substantial and focused planning and investment will we be able to successfully migrate our existing application base and supporting high-performance software infrastructure to next-generation computing systems.

## 5 Addressing Application Software Productivity Challenges in the Office of Science: Recommendations

Addressing these challenges in extreme-scale application software productivity is essential in order to fully exploit emerging architectures for scientific discovery. Moreover, because software is the practical means through which computational science collaboration occurs, this work is a prime opportunity for synergistic collaboration across the Office of Science, centered on partnerships between ASCR and its sister programs in DOE/SC, as well as activities in concert with NNSA and other DOE offices. The following are some of the most important recommended activities:

- Convene a workshop to identify the short and long-term research challenges of extreme-scale scientific application software productivity.
- Include application software productivity as an integral part of the Exascale Computing Initiative (ECI) discussions.
- Develop a strategic plan to address long-term extreme-scale computational science software productivity challenges.

Additional priorities include:

- Address software implications of extreme-scale systems productivity.
- Develop and disseminate educational materials on emerging lessons learned from early manycore application development efforts and best practices in scientific software engineering.
- Address HPC legacy code refactoring to take advantage of new architectures and programming models.
- Determine strategies for the extreme-scale computational science community to leverage and influence software best practices.

Success in these activities will require coordinated investment from other agencies such as NSF, DOD, and NASA.

# Appendix A: Application Software Productivity Case Studies

We expand on the exteme-scale software productivity challenges introduced in Section 3 from the perspective of three mission-critical DOE applications: climate modeling, environmental management, and magnetic fusion energy.

## A.1 DOE Climate Modeling Software Infrastructure

### Climate Modeling and Simulation Software Challenges

A climate model is the canonical multiphysics and multiscale application. Representing all the internal variability and other features of the climate system requires several individual models of the atmosphere, ocean, ice, and land coupled to one another through exchanges of heat, mass, and momentum. Each of these models is itself a multiphysics application. For example, the atmosphere includes components describing chemistry, geophysical fluid dynamics, radiative transfer, and other processes. The climate system is multiscale in both time and space.

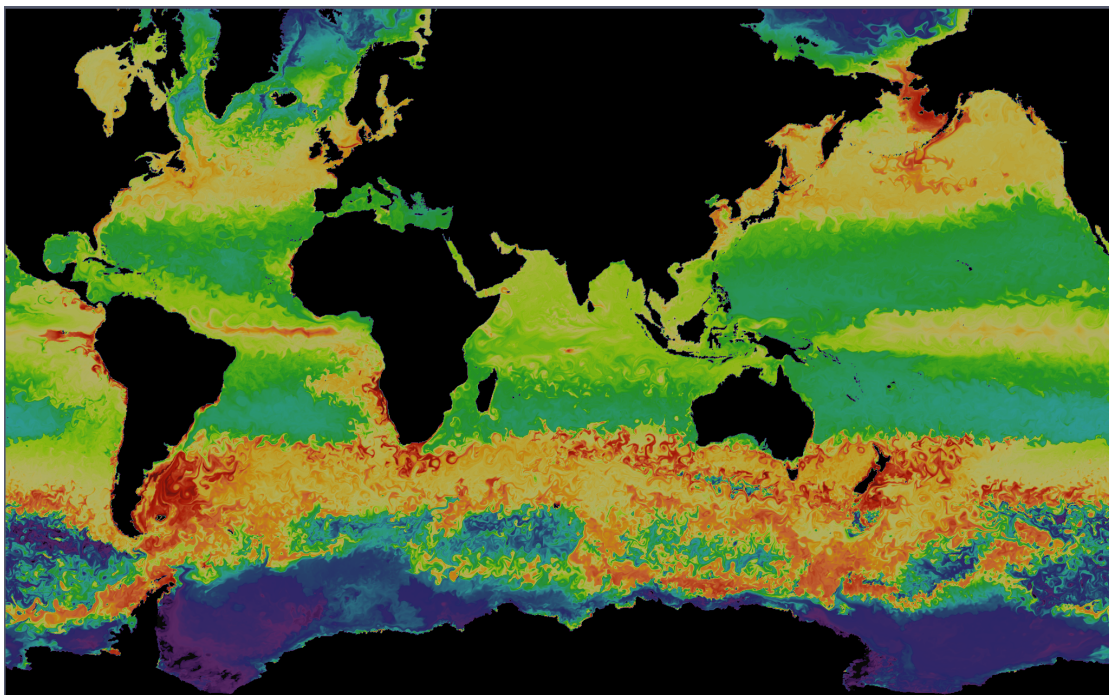

*Figure A.1: Chlorophyll concentration from an eddy-resolving ocean simulation that includes a biogeochemical model to explore ocean carbon uptake and acidification.*

Length scales range from the micrometer scales of cloud microphysics to the planetary scales of major circulation features. New variable resolution approaches are being introduced to capture part of this range, but some processes will always be parameterized. Time scales from seconds to minutes must be resolved while performing decadal and centennial simulations for longer-term variability and analysis of climate change. Results from climate simulations are used by a broad

range of users in order to identify strategies for adapting to and mitigating the effects of climate change. In order to provide these users with better estimates of regional/local changes and climate extremes, ensembles of simulations and high-resolution simulations are needed. Current high-resolution (~10 km grid spacing) configurations require over 100M core-hours on leadership-class computing resources for a small (5-member) ensemble of only 30 simulated years each.

The development of coupled climate models involves hundreds of climate scientists, applied mathematicians, and computer scientists working on all aspects of the model in order to include appropriate physical models, create the most accurate and efficient algorithms, and implement these on the available computer architectures. DOE researchers currently collaborate on the Community Earth System Model (CESM) that includes over 300 developers working on 1.2 million lines of code mostly written in Fortran. The long history of climate modeling dating back to the 1960s has required implementing the models on new computing architectures while also preserving the fidelity of current simulations. In fact, BER is initiating a new project that will start with the CESM code base but develop a version that can better utilize DOE leadership computing and is more targeted to climate problems of interest to DOE. This transition beyond petascale to exascale systems will require substantial refactoring of the CESM, exploration of new algorithmic choices, and a significant investment in software engineering to ensure a more efficient transition.

The software productivity and engineering challenges include the following:

- Refactoring legacy code and introducing new algorithms to exploit the capabilities of new computing architectures
- Introduction of more comprehensive V&V and UQ frameworks, particularly in an environment where data is often sparse and first-principle solutions are not available
- Training in and adoption of proven software methodologies that can be customized for large, distributed climate model development teams
- Evolution of coupling methodologies for new variable-resolution component models
- Development of a communitywide high-performance climate modeling software infrastructure as recommended by a recent NRC report [6]. This infrastructure should feature shared abstractions that enable performance portability, adaptability to rapid changes in underlying architecture, and ease of programming for nonexpert developers.

## A.2 DOE Environmental Management Software Development

The Office of Environmental Management (EM) oversees the remediation and closure of DOE sites storing legacy waste from the development of nuclear weapons and related technologies. Although significant cleanup progress has been made, several large and technically complex sites remain, with total life cycle cost estimates between $272 and $327 billion. The groundwater and soil contamination alone includes more than 40 million cubic meters of contaminated soil and debris and 1.7 trillion gallons of contaminated groundwater at 17 sites in 11 states [20]. At these complex sites the conservative simplifications and abstractions used in the past to estimate the fate and transport of contaminants can lead to overly conservative and costly remediation and closure scenarios. For the complex cleanup problems that remain, the highly uncertain and multiscale nature of the subsurface hydrology and geochemical transport must be treated mechanistically, requiring advanced modeling and data analysis techniques that leverage modern computational resources in order to inform a comprehensive approach to risk and performance assessment. To address this urgent need for transformational solutions, EM initiated the Advanced Simulation Capability for Environmental Management (ASCEM) program in 2010 [1]. The expectation of DOE-EM and the earth science community is that programs such as ASCEM can move beyond existing regulatory codes, significantly reducing reliance on conservative abstractions and simplifications through the use of advanced and emerging computational methods.

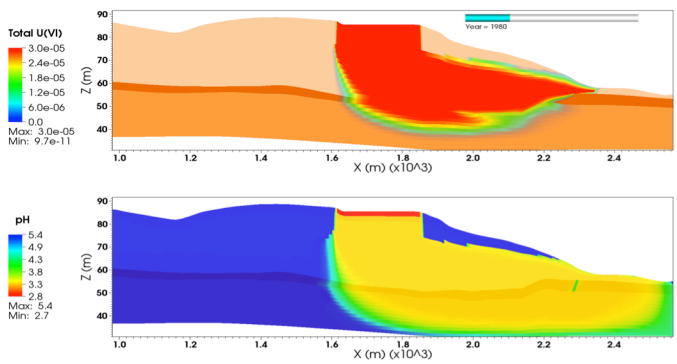


*Figure A.2: Simulations of complex geochemistry and hydrology at the Savannah River F-Area seepage basins capture the retardation of the Uranium plume, prior to any remedial action. Sorption of Uranium depends on the pH, and hence, the Uranium plume (top) lags behind the acidic front of the waste water (bottom). Manipulation of pH is a key ingredient in controlling Uranium migration in the future [10]. ASCEM tools under development will support simulations and analysis of the effects of ongoing and future remedial actions leading to better projections of the time and cost required to meet remedial goals.*

### Software Challenges of Subsurface Process Simulation

Subsurface flow and transport are governed by a particularly challenging suite of coupled multiscale processes (including surface and subsurface flows, bio-geochemical reactions, and thermal-mechanical deformations) occurring in highly heterogeneous subsurface environments with external forcing. Exacerbating the

complexity is the often-limited information about the heterogeneous distribution of subsurface flow and transport properties and reaction networks. Consequently, risk and performance assessment uses a graded and iterative approach to underpin scientifically defensible decisions and strategies. This approach first establishes simplified models and then iteratively enhances geometric and process level complexity to identify and characterize the key processes and assumptions that are needed to efficiently reach a defensible decision. ASCEM must design its approach and tools to handle this workflow, not only to be flexible and extensible, but also to leverage an increasingly powerful and diverse set of computational resources. To achieve this goal the ASCEM program has identified common themes found in multiscale and multiphysics applications, and has assembled an interdisciplinary team to leverage advances from various DOE SC programs, ASC, and earth science. This effort is in its infancy and faces many of the challenges outlined previously in Section 3. At the heart of these challenges for ASCEM is the need to develop a new approach to the lifecycle of software tools in the earth sciences. In particular, development and collaboration tools continue to advance, enabling much more agile approaches that can make use of hierarchal and automated testing. Furthermore, languages and design methodologies continue to advance, improving our ability to reduce long-term maintenance and refactoring costs. These advances could enable a much more efficient approach to development and maintenance of regulatory codes, whose strict quality assurance requirements have significantly hindered adoption of new algorithms and architectures. Key software engineering challenges include the following:

- Developing a new modular hierarchical design of integrated tools that not only addresses life cycle needs for flexibility and extensibility, but also provides a suitable framework to refactor or encapsulate existing robust legacy tools
- Developing a comprehensive hierarchical approach to testing that aligns development practices with the efficient migration of code from the research branch, through stable community code releases, to fully qualified regulatory releases (NQA-1 [2])
- Supporting flexible nonlinear workflows that integrate tools, data, and simulations, which are distributed across a range of platforms, in order to inform the graded and iterative modeling approach and subsequent cleanup and monitoring
- Training an interdisciplinary team to have a common understanding of both the application modeling and simulation needs, as well as modern agile software design, development, and testing practices
- Meeting the demand for high-fidelity multiscale process-level simulations through new algorithms and/or implementations on emerging architectures, in order to address discretization, solver, statistical sampling, and visualization needs.

A focused effort to address these complex software engineering challenges is needed in order to realize the potential benefits of emerging computational power.

## A.3 Application Software Productivity for DOE Fusion Energy Sciences

### Fusion Energy Modeling and Simulation Software Challenges

The goal of modeling and simulations in fusion energy research is to harness the rapidly increasing power of extreme-scale computing platforms in order to simulate the complex dynamics governing key magnetic confinement properties of fusion-grade plasmas—especially ITER, a $20-billion international experimental device under construction in France and involving the partnership of 7 governments representing over half the world's population. Fundamentally, the fusion of light nuclides forms the basis of energy release in the universe, which can potentially be harnessed and used as a clean and sustainable supply of energy on Earth. In order to build the scientific foundations needed to develop fusion energy, a key requirement is the timely development of reliable predictive simulation capabilities for magnetically confined fusion plasmas. Fusion energy science (FES) simulation is well recognized as a grand challenge multiphysics, multiscale science application. A large international FES community of computational plasma physicists, applied mathematicians, and computer scientists has been actively engaged over the years in the development of increasingly realistic physics-based predictive modeling capabilities.

Advanced computing is expected to be vital for accelerating the needed progress in FES research in the 21st century. The imperative is to translate the combination of the rapid advances in supercomputing power from the petascale to the exascale range and beyond, together with the emergence of effective new algorithms and computational methodologies to help enable corresponding increases in the physics fidelity and the performance of the scientific codes used to model complex physical systems. The magnetic fusion energy research community has made excellent progress in developing advanced codes for which computer runtime and problem size scale very well with the number of processors on massively parallel supercomputers. A good example is the effective usage of the full power of modern leadership class computational platforms at the petascale and beyond to produce nonlinear particle-in-cell simulations that have accelerated progress in understanding the nature of plasma turbulence. For example, Figure A.3 shows results from a petascale-level production simulation (24M CPU hours, engaging 100K cores of the OLCF Jaguar system for 240 hours) that was carried out by the XGC-1 code, which integrates plasma dynamics in the complex edge with the core region of the DIII-D Tokamak plasma. More recently, excellent demonstrations in fusion codes have been presented that illustrate the algorithmic progress in dealing with low-memory-per-core, extreme-scale computing challenges for the current top 4 supercomputers worldwide—just prior to the recent introduction of China's Tianhe-2 computer system.

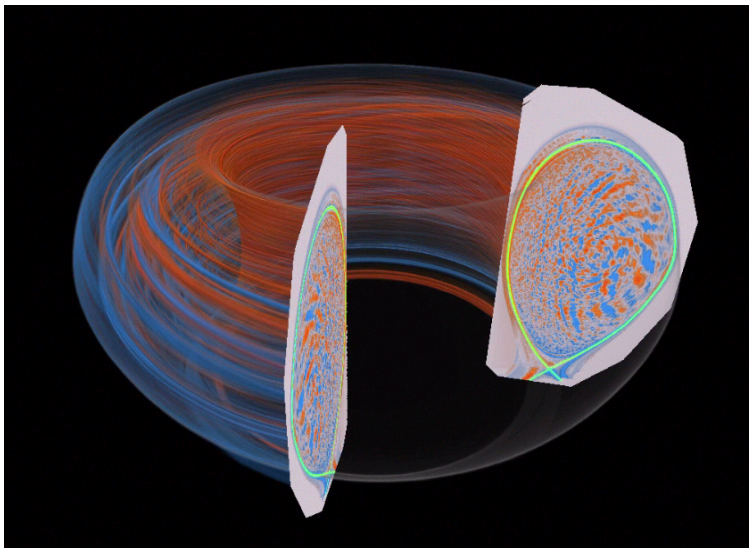

*Figure A.3: Flux driven turbulence (fluctuating electrostatic potential), filling the whole plasma volume in diverted DIII-D geometry.*

Addressing formidable questions from a whole-system perspective in magnetic fusion energy research will require development of integrated predictive simulation capabilities with unprecedented physics fidelity. This will require proper cross-validation of laboratory experiments with a suite of advanced codes in regimes relevant for producing practical fusion energy. Adoption of proper software engineering principles for engaging verification, validation, and uncertainty quantification (VVUQ) will demand systematic testing and improved community development practices. Associated software engineering challenges for FES include the following:

- Modernization of legacy fluid (MHD) codes through novel programming approaches to demonstrate scalability on new multicore/manycore computing architectures
- More effective V&V and UQ frameworks and workflows to improve the physics fidelity and reliability of the integrated FES code, with the goal of providing "open-source" solutions to FES applications scientists
- Evolution of coupling/integration methodologies to enable possible adoption of variable-resolution component models, and more tightly integrated component coupling to address the challenge of reducing communications on the path to extreme-scale systems
- Exploration of modern machine learning methodologies to address "big data" challenges associated with avoiding macroscopic disruptive events in fusion-grade plasmas
- Development of a communitywide high-performance FES modeling software infrastructure with shared abstractions that enable portability and adaptability to rapid changes in underlying architecture, and ease of programming for nonexpert developers
- Recruiting and training new software engineers and computational scientists to complement existing personnel and fill needed capabilities

## References


[1] Advanced Simulation Capability for Environmental Management (ASCEM) program, URL: http://ascemdoe.org.

[2] ASME NQA-1-2004 (including addenda 2005 and 2007), *Quality Assurance Requirements for Nuclear Facility Applications*, American Society of Mechanical Engineers, 2007.

[3] *ASCR Programming Challenges for Exascale Computing*, Office of Science, U.S. Department of Energy, 2011. URL: http://science.energy.gov/~/media/ascr/pdf/program-documents/docs/ProgrammingChallengesWorkshopReport.pdf.

[4] R. Bartlett, M. Heroux, J. Willenbring, TriBITS Lifecycle Model Version 1.0: *A Lean/Agile Software Lifecycle Model for Research-based Computational Science and Engineering and Applied Mathematical Software*, SANDIA REPORT SAND2012-0561, February 2012.

[5] L. Bass, P. Clements, and R. Kazman, **Software Architecture in Practice**, 3rd Edition. Software Engineering Institute series in Software Engineering. Addison-Wesley, 2012. ISBN: 9780321815736.

[6] C. Bretherton (Chair), *A National Strategy for Advancing Climate Modeling*, National Research Council of the National Academies, 2012.

[7] D. Brown and P. Messina (Chairs), *Scientific Grand Challenges: Crosscutting Technologies for Computing at the Exascale*, 2010. URL: http://science.energy.gov/%7E/media/ascr/pdf/program-documents/docs/Crosscutting_grand_challenges.pdf.

[8] J. Dongarra, P. Beckman, et al. *The International Exascale Software Project Roadmap*, **IJHPCA**, 25:3–60, 2011.

[9] J. Dongarra, R. Graybill, W. Harrod, R. Lucas, E. Lusk, P. Luszczek, J. McMahon, A. Snavely, J. Vetter, K. Yelick, S. Alam, R. Campbell, L. Carrington, T.-Y. Chen, O. Khalili, J. Meredith, M. Tikir, *DARPA's HPCS Program: History, Models, Tools, Languages*, 2008.

[10] M. Freshley, S. Hubbard, I. Gorton, J. D. Moulton C. I. Steefel, V. Freedman, H. Wainwright, et al. *Phase II Demonstration*. Technical Report ASCEM-SITE-2012-01, Office of Environmental Management, United States Department of Energy, Washington, DC, 2012.

[11] M. Heroux, R. Bartlett, J. Willenbring. *Software Engineering Principles: The TriBITS Lifecycle Model,* DOE Climate Summit. Berkeley, CA, December 2012. URL: http://www.sandia.gov/~maherou/docs/HerouxTribitsOverview.pdf.

[12] J. Kepner, Guest Editor, *High Productivity Computing Systems and the Path Towards Usable Petascale Computing, Part A: User Productivity Challenges*,

**Cyberinfrastructure Technology Watch (CTWatch) Quarterly**, http://www.ctwatch.org, Volume 2, Number 4A, November 2006.

[13] J. Kepner, Guest Editor, *High Productivity Computing Systems and the Path Towards Usable Petascale Computing, Part B: System Productivity Technologies*, **Cyberinfrastructure Technology Watch (CTWatch) Quarterly**, http://www.ctwatch.org, Volume 2, Number 4B, November 2006.

[14] D. E. Keyes, L. C. McInnes, C. Woodward, W. Gropp, E. Myra, M. Pernice, J. Bell, J. Brown, A. Clo, J. Connors, E. Constantinescu, D. Estep, K. Evans, C. Farhat, A. Hakim, G. Hammond, G. Hansen, J. Hill, T. Isaac, X. Jiao, K. Jordan, D. Kaushik, E. Kaxiras, A. Koniges, K. Lee, A. Lott, Q. Lu, J. Magerlein, R. Maxwell, M. McCourt, M. Mehl, R. Pawlowski, A. P. Randles, D. Reynolds, B. Rivière, U. Rüde, T. Scheibe, J. Shadid, B. Sheehan, M. Shephard, A. Siegel, B. Smith, X. Tang, C. Wilson, and B. Wohlmuth. *Multiphysics simulations: Challenges and opportunities*, **IJHPCA**, 27(1):4–83, Feb 2013, special issue. doi: 10.1177/1094342012468181.

[15] P. Kogge (Editor), *Exascale Computing Study: Technology Challenges in Achieving Exascale Systems*, Sept 2008.

[16] W. L. Oberkampf, M. Pilch, and T. G. Trucano, *Predictive Capability Maturity Model for Computational Modeling and Simulation*, SANDIA REPORT SAND2007-5948, October 2007, available at: https://cfwebprod.sandia.gov/cfdocs/CCIM/docs/Oberkampf-Pilch-Trucano-SAND2007-5948.pdf.

[17] E. Przybylowicz (Chair), *Advice on the Department of Energy's Cleanup and Technology Roadmap: Gaps and Bridges*, National Research Council of the National Academies, 2009.

[18] R. Rosner (Chair). *The Opportunities and Challenges of Exascale Computing. Office of Science*, U.S. Department of Energy, 2010. URL: http://science.energy.gov/~/media/ascr/ascac/pdf/reports/Exascale_subcommittee_report.pdf.

[19] S. Sachs (Editor). *Tools for Exascale Computing: Challenges and Strategies, Report of the 2011 ASCR Exascale Tools Workshop*, Office of Science, U.S. Department of Energy, 2011. URL: http://science.energy.gov/~/media/ascr/pdf/research/cs/Exascale%20Workshop/Exascale_Tools_Workshop_Report.pdf.

[20] *Science and Technology to Reduce the Life Cycle Cost of Closure – Investing in Our Future: Technology Innovation and Development for Footprint Reduction*, Office of Environmental Management, U.S. Department of Energy, Washington, D.C., 2010.