National Science Foundation
Advisory Committee for CyberInfrastructure
Task Force on Software for Science and Engineering

Final Report, March 2011

Task Force Co-Chairs from the ACCI

David Keyes, KAUST and Columbia
Valerie Taylor, TAMU

ACCI Members

Tony Hey, Microsoft
Stuart Feldman, Google

Community Panelists

Gabrielle Allen, LSU
Phillip Colella, LBNL
Peter Cummings, Vanderbilt
Frederica Darema, AFOSR
Jack Dongarra, UT
Thom Dunning, UIUC
Mark Ellisman, UCSD
Ian Foster, ANL and UofC
William Gropp, UIUC
Chris Johnson, Utah
Chandrika Kamath, LLNL
Ravi Madduri, ANL
Michael Mascagni, FSU
Steve Parker, NVIDIA
Padma Raghavan, PennState
Anne Trefethen, Oxford
Scott Valcourt, UNH

Principal NSF Liaison

Abani Patra, SUNY Buffalo

NSF Liaisons

Fahmida Choudhury, SBE, Clark Cooper, ENG, Peter McCartney, BIO, Manish Parashar, OCI, Tom Russell, OIA Barry Schneider, OCI, Jen Schopf, OCI, Nigel Sharp, MPS

Preface

The Software for Science and Engineering (SSE) Task Force commenced in June 2009 with a charge that consisted of the following three elements:

- *Identify specific needs and opportunities across the spectrum of scientific software infrastructure*. Characterize the specific needs and analyze technical gaps and opportunities for NSF to meet those needs through individual and systemic approaches.
- *Design responsive approaches.* Develop initiatives and programs led (or co-led) by NSF to grow, develop, and sustain the software infrastructure needed to support NSF's mission of transformative research and innovation leading to scientific leadership and technological competitiveness.
- *Address issues of institutional barriers*. Anticipate, analyze and address both institutional and exogenous barriers to NSF's promotion of such an infrastructure.

The SSE Task Force members participated in bi-weekly telecons to address the given charge. The telecons often included additional distinguished members of the scientific community beyond the task force membership engaged in software issues, as well as personnel from federal agencies outside of NSF who manage software programs. It was quickly acknowledged that a number of reports loosely and tightly related to SSE existed and should be leveraged. By September 2009, the task formed had formed three subcommittees focused on the following topics: (1) compute-intensive science, (2) data-intensive science, and (3) software evolution.

Throughout the process of preparing this report, it was deemed important to seek input from the broader community. This was accomplished through a combination of birds-of-a-feather sessions at conferences, participation in relevant workshops, and individual conversations at conferences and our respective institutions. In particular, we participated in the series of International Exascale Software Workshops co-sponsored by DOE and NSF in 2009, a birds-of-a-feather session at SC 2009, and informal discussions at SC 2010. Further, presentations were given at DDDAS Workshop in August 2010 and the CI-EPSCOR Workshop in October 2010. In addition, some of the members of the SSE Task Force were members of five other NSF ACCI task forces working simultaneously on other reports in this series, thereby allowing for cross-referencing of materials among the different task forces. Approximately two years from inception, we are pleased to deliver the final version of the SSE task force report.

Table of Contents

Executive Summary

Software is a critical and pervasive component of the cyberinfrastructure for science and engineering. It is the software that binds together the hardware, networks, data, and users such that new knowledge and discovery result from cyberinfrastructure. Furthermore, software must evolve to meet rapid changes in hardware architectures as well as any new functionality that is demanded and results when communities advance in various disciplines. According to many federal reports, software is *a*, if not *the*, "grand challenge" of cyberinfrastructure. Yet, software is historically among the least coordinated and systematically funded components of cyberinfrastructure. Accordingly, the NSF Advisory Committee on Cyberinfrastructure has chartered a Task Force on Software for Science and Engineering (SSE) to identify findings and form recommendations to NSF on how best to fulfill its mission of promoting the progress of science and to help meet the demand on software to deliver ubiquitous, reliable, and easily accessible computer and data services.

Software infrastructure has evolved organically and inconsistently, with its development and adoption coming largely as by-products of community responses to other targeted initiatives. Software *creation* is very often encouraged and a legitimate focus of NSF-funded projects. The long-term funding for the *evolution* and *maintenance* of software is often difficult to support through NSF. Support of software infrastructure for NSF-funded projects and large community initiatives generally depends upon funding sources that are one-time, sporadic, and domain-specific where the focus is on the science or engineering outcomes. Currently, there is no systematic and periodic process for determining software requirements and priorities across the NSF community. Nor are there generally accepted quantitative metrics for determining what software researchers most heavily use.

Good software needs to be developed in a comprehensive, end-to-end fashion. Software infrastructure should enable users to exploit, integrate, and cross-leverage evolving software tools. Some of the challenges include coordinating the interfaces between software and other components that need to interoperate to accomplish the science and engineering goals of interest. Increases in code complexity for addressing the paradigm shifts in computer architectures (including using accelerators and multi/many core chips) threaten to exceed the capacity of the research community for software development and support. Good software infrastructure not only meets needs that are recognized at the time of its design, but is extensible for the meeting of unanticipated needs for long periods between occasional fresh starts due to so-called "disruptive technologies". Furthermore, it is important that software infrastructure address issues related to open access, portability, reuse, composability, and dissemination. We are at a most opportune time for NSF to rethink the research, development, and maintenance of our software infrastructure.

The Task Force on Software for Science and Engineering has formed the following major recommendations on how NSF can best support the research, development, and maintenance of software infrastructure:

1. NSF should develop a multi-level (individual, team, institute), long-term program of support of scientific software elements ranging from complex applications to tools of utility in multiple domains. Such programs should also support extreme scale data and simulation and the needs of NSF's Major Research Equipment and Facilities (MREFC) projects.
2. NSF should take leadership in promoting verification, validation, sustainability, and reproducibility through software developed with federal support.
3. NSF should develop a consistent policy on open sources software that promotes scientific discovery and encourages innovation.
4. NSF support for software should entail collaborations among all of its divisions, related federal agencies, and private industry.
5. NSF should utilize its Advisory Committees to obtain community input on software priorities.

1 Introduction

The mission of NSF, "To promote the progress of science; to advance the national health, prosperity, and welfare; to secure the national defense…," is increasingly dependent on cyberinfrastructure, which includes hardware, networks, data, software, and trained users. *Software is a critical and pervasive component of the cyberinfrastructure for science and engineering.* It is the software that binds together the hardware, networks, data, and users such that new knowledge and discovery result from cyberinfrastructure. Furthermore, software must evolve to meet rapid changes in hardware architectures as well as new functionality that is demanded and results when communities advance in various disciplines. According to many federal reports, software is *a*, if not *the*, "grand challenge" of cyberinfrastructure. Yet, software is historically among the least coordinated and systematically funded components of cyberinfrastructure. Accordingly, the NSF Advisory Committee on Cyberinfrastructure has chartered a Task Force on Software for Science and Engineering (SSE) to identify findings and form recommendations to NSF on how best to fulfill its mission of promoting the progress of science and to help meet the demand on software to deliver ubiquitous, reliable, and easily accessible computer and data services.

Software infrastructure has evolved organically and inconsistently, with its development and adoption coming largely as by-products of community responses to other targeted initiatives. Software *creation* is very often encouraged and a legitimate focus of NSF-funded projects. The long-term funding for the *evolution* and *maintenance* of software is often difficult to support through NSF. Support of software infrastructure for NSF-funded projects and large community initiatives generally depends upon funding sources that are one-time, sporadic, and domain-specific where the focus is on the science or engineering outcomes. For example, ROMS – the Regional Ocean Modeling System, which is widely used by over 2,000 users worldwide via an open source license, has been funded for many years by ONR; NSF provided some initial funding to develop algorithms for data assimilations.

Software is a critical and pervasive component of the cyberinfrastructure for science and engineering.

The "software stack" now invoked by scientists and engineers consists of systems software (e.g., operating systems, file systems, compilers), middleware (e.g., file transfers, multi-model communication, provenance), libraries (e.g., numerical libraries, communication libraries), and applications (whose creation is often driven by specific objectives, but which then evolve to become shared more widely than anticipated). While the community has functioned with the organic and *ad hoc* development of this software stack, the stack is becoming fragile due to stresses from many directions: increasing complexity, dynamic and adaptive resource requirements, dependencies across layers of the software stack, hardware that responds to market forces that are not science-driven; increasing diversity of the community to be supported, which is becoming larger and on average broader and less computationally sophisticated

than the more technologically oriented "pioneers" who invent the software infrastructure; expansion of expected functionality; reliability requirements that become stricter, and so forth. Some of these causes of stress are signs of progress. For example, computational simulation and data analysis have been critical to innumerable scientific endeavors with the result that scientists who are not computing experts today are computing successfully in large numbers. However, future development and maintenance of the software infrastructure on which the computing community critically depends will be difficult if our support mechanisms continue in the current *ad hoc* mode. The confluence of these stresses, especially with respect to rapid hardware changes and ripeness of scientific opportunity, suggests that we are at a most opportune time for NSF to rethink the research, development, and maintenance of our software infrastructure. Furthermore, it is important that software infrastructure address issues related to open access, portability, reuse, composability, and dissemination.

The identification of software standards that deserve to be supported is one of the roles that NSF's peer-review processes can facilitate. However, today's *ad hoc* and loosely coordinated approaches to software infrastructure allow unanticipated breakthroughs and chances for new ideas to arise and influence the entire cyber-ecosystem; this must not be lost in a well-meaning attempt to make the ecosystem more efficient through designation of approaches as preferred or deprecated. Good software reliably and efficiently encodes expertise in processing data and delivers it across well-understood interfaces to users and other developers and integrators who require that expertise. However, some principles for software design and some metrics for software evaluation are subjective and controversial, and should not be prescribed too narrowly or rigidly. A balance must be preserved between standardization for efficiency and flexibility for innovation.

Good software needs to be developed in a comprehensive end-to-end fashion. Software infrastructure should enable users to exploit, integrate, and cross-leverage evolving software tools. Some of the challenges include coordinating the interfaces between software and other components that need to interoperate to accomplish the science and engineering goals of interest. Further, increases in code complexity for addressing the paradigm shifts in computer architectures (including using accelerators and multi/many core chips) threaten to exceed the capacity of the research community for software development and support. Good software infrastructure not only meets the needs that are recognized at the time of its design, but is extensible for the meeting of unanticipated needs for long periods between occasional fresh starts due to so-called "disruptive technologies".

Software has become an essential tool for knowledge discovery in many disciplines and often also itself serves as a representation of knowledge. Hence, there is an urgent need to dedicate increasing resources to software, especially given the architecture transitions anticipated for the coming decade. Because of NSF's prestige and the peer-consensus care with which it sets priorities, there is an important role for NSF in the global context of cyberinfrastructure. The resources that NSF can bring to the table to enable its own scientists and engineers to be productive are formidable and necessary. An equally necessary role for NSF is to stimulate investments of others and expand the reward structure for contributions to cyberinfrastructure in the international scientific workplace, whether in universities, government laboratories, or industry.

Any effort to rethink software infrastructure must involve stakeholders from academia, industry, and national laboratories, as well as the other basic science and mission agencies. Further, the focus of this effort should include all software in support of NSF's science and engineering

There is currently no systematic and periodic process for determining software requirements and priorities across the NSF community.

mission, not only the high-end software that is most visibility. There is currently no systematic and periodic process for determining software requirements and priorities across the NSF community. Nor are there generally accepted quantitative metrics for determining what software researchers most heavily use. In the absence of such inventories of requirements and usage, it will be difficult for NSF to do an optimal job of allocating resources to software projects. However, some sure-fire improvements to current programs are clearly possible and should be pursued simultaneously with improvements to tracking of needs and use.

In responding to the recommendations of the Taskforce on Software for Science and Engineering, NSF should keep in mind that while no single agency can influence the entire open global cyber-ecosystem, NSF can aspire to set forth compelling principles and examples and to offer compelling incentives for software compatibility. NSF's influence could be tremendous because of its dominance in university-based computer science and mathematics research in the United States and its willingness to fund leading edge, high-risk research. While NSF's recognized responsibilities in cyberinfrastructure are long-term, the SSE Taskforce is but a first step intended to facilitate discussion among NSF administrators and within the community.

NSF can aspire to set forth compelling principles and examples and to offer compelling incentives for software compatibility.

The next three sections, Sections 2 through 4, provide the analysis of the needs and opportunities of software infrastructure with respect to three areas: compute intensive science; data, federation, and collaboration; and software evolution. The first two areas represent the hardware and observational facilities that are expected to yield new scientific results and the third is needed to represent the complex cycle of software from creation to hardening and beyond. It is recognized that there is significant overlap among all three areas. The first two areas, however, are intended to represent the continuum between compute intensive science and data intensive science. Section 5 provides a discussion of the institutional barriers with respect to SSE followed by a brief summary of related activities ongoing in other agencies in Section 6. The findings and recommendations are summarized in Sections 7 and 8, respectively.

2 Compute Intensive Science

Advanced computing is an essential tool in addressing scientific problems of strategic international importance, including climate prediction, nanoscience, new materials, drug design, biomedical modeling, and next-generation power sources; it is equally essential to solve commercial and industrial problems in financial modeling, engineering, and real-time decision systems. For example, the Southern California Earthquake Center (SCEC) seeks to develop a predictive understanding of earthquake processes aimed at providing society with improved understanding of seismic hazards. In partnership with earthquake engineers, the SCEC researchers are developing the ability to conduct end-to-end simulations ("rupture to rafters") to extend their understanding of seismic hazards to include earthquake risks and risk mitigation strategies.

Simulations of physical entities are increasing in accuracy through finer scale approximations, inclusion of increasingly realistic physics, and integration of models across different scales. In other systems, process analysis and planning simulations involve integration of models across several modalities and behaviors of the system. For any scale of computing these changes invariably lead to increasing complexity in software and, for compute-intensive science, a dependency on layers of software from the application interface, mathematical and communication libraries, down to compilers and the operating system. The increasing software complexity and uncertainty about the computer architecture and programming model will impact the life-cycle of simulation software from inception to new science results.

The increasing software complexity and uncertainty about the computer architecture and programming model will impact the life-cycle of simulation software from inception to new science results.

Compute Systems

In recent years there have been a number of studies and research community workshops that have made the case for simulation and computational requirements; see [6,7,12, 14,15,23,25] to name a few. Many of these meetings have been focused on high-performance (petascale through exascale) computation – but their findings regarding algorithms and software are not restricted to extreme computing. For example, the report on exascale computing for energy and environment [23] notes the following:

> "*The current belief is that the broad market is not likely to be able to adopt multicore systems at the 1000-processor level without a substantial revolution in software and programming techniques for the hundreds of thousands of programmers who work in industry and do not yet have adequate parallel programming skills*."

Programmers targeting the desktop of tomorrow will face many of the same issues encountered with today's high-end computing. These issues include multi- and many-core programming challenges, dealing with heterogeneous computer architectures possibly requiring support for mixed arithmetic, and as noted above a dependency on a deep software stack.

Issues for computing software include the capacity to deal with large-scale concurrency and heterogeneity from the desktop architecture up to the peta- and exascale systems. It is expected that Moore's law is only likely to apply for the next decade after which time we will

have hit the limits of miniaturization; the days of increasing clock rates and instruction-level parallelism will be over. Hence processors are being built that have multiple cores and slower clock rates requiring software designed and written in a multi-threaded fashion. It is increasingly the case that desktops and the core building blocks for larger systems are made up of multicore chips with additional accelerator hardware such as a graphics processor unit (GPU) or similar. This adds increased complexity and a requirement for software that can behave appropriately in these heterogeneous environments.

Power usage is becoming increasingly important. Software that can interact with the hardware system to enable power-saving actions (such as powering down idle hardware components) to reduce power consumption need to be part of the design space.

The ability to move codes from one computer system to another and to have efficient and effective use of the resources are going to continue to be of great importance and likely to increase in difficulty due to the complexity of the systems and the software. There needs to be more research and development focused on automatic generation, auto-tuning, and other areas that will allow software to be developed at a high-level, yet will adapt to the underlying architecture.

Recommendation: *NSF should support the development of portable systems through such things as automatic code generation and auto-tuning approaches.*

Recommendation: *NSF should encourage close communication between chip designers, system builders, and software developers through appropriate collaborative research grants.*

The Software Stack

The layers of software underpinning any given simulation application will typically consist of a mix of open source community-supported software, commercial tools and libraries, and application-specific components developed by research students and post doctoral researchers [25]. In general the "software infrastructure" for compute-intensive applications is a rather *ad hoc* patchwork of supported and unsupported software [3]. In many cases key simulation codes grow organically, as research code is added to an existing body of code, resulting in unsustainable applications that cannot be easily verified, in which error propagation from one part of the code to others may not be well understood, and indeed the in-house developed software is not likely to perform efficiently across any number of computer platforms. Such shortcomings will increase with the increasing multilevel, complex hierarchy of the hardware platforms.

Application Software

The ACCI Task Force report on Grand Challenges [16] articulates well the requirements for the grand challenge applications in CS&E; similarly in [3] the software infrastructure is foreseen for that driven by applications in four areas of science, namely astrophysics, atmospheric sciences, evolutionary biology, and chemical separations. The requirements identified include some general requirements of scalable software systems; higher-level abstractions to allow application developers an easier development environment; the provision of efficient, portable "plug and play" libraries; and code generation techniques to support portability. Here, as in more recent reports, there is an urgent call and warning from specific research communities with notes like the following: "*The increases in code complexity could exceed the capacity of the national centers for software development and support."*

"The increases in code complexity could exceed the capacity of the national centers for software development and support."

The compute-intensive application challenges require research, support, services, education and training to affect a change in culture in some cases. Many applications involve multiple models at different scales or for different elements of the application. A lack of

standards in application programming interfaces, data models and formats, and interoperability of programming models makes this integration of different models challenging. This is equally true beyond the simulation code itself as it is usually part of a pipeline of activity – model creation, calculation, analysis and visualization. Without agreed upon interfaces and data formats, building this pipeline is difficult and time consuming. To enable portability without constant rewriting of code, we need different approaches including adaptive algorithms to adapt to problem characteristics and also architectural constraints.

Further, support and services are needed to provide guidance to application developers on best practices for software engineering, provisioning of software tools to assist in code development, and processes to ensure good computational components are understood and used, not reinvented. The route to sustainable software is complex and application communities cannot be expected to go it alone.

Recommendation: *NSF should support standards development in both application specific data formats, and generic requirements for multi-scale, multi-model integration.*

Development Environments

The development environment is the suite of software and tools that the application developer might use to create the application code. It includes the programming language, programming model, software frameworks, compilers and libraries as well as the debugging and optimization tools. As noted above, software longevity is going to be difficult to maintain in an ever-changing hardware environment. This will be true of development environment software and tools as well as the application software. The ACCI Task Force report on Grand Challenges notes "*The Message Passing Interface (MPI) based programming model based on an inherently flat architecture ……will need to be reinvented to meet application challenges….".* While this refers to high-performance applications, the same is true of desk-top application environments and standard programming models for programming heterogeneous nodes.

In the past, frameworks and toolkits, such as PETSc [18], have been developed that ease the burden on application developers by providing a collection of tools, interfaces, algorithms that allow a level of encapsulation and abstraction that is easy for applications to use effectively at the same time providing efficient use of the underlying resources. Research is required into new approaches to enable a simpler programming environment for application developers – allowing composability and portability of software components.

Numerical Libraries

While there have been great advances in computing hardware, algorithms and software libraries have contributed as much to increases in computational simulation capability as have improvements in hardware [14,15]. Increasing complexity of applications and resources can be dealt with only through the availability of good software. The fast moving developments in hardware architectures and the lack of software standards to support those developments are major challenges for the creation of a stable software infrastructure.

In many fields there are well-defined software components that provide the building blocks for computational simulations and more could be gained by this approach. Some studies [5,7, 9,12,23,25] have identified the cross application components that indicate the layers of software dependencies for the suite of applications. The International Exascale Software Project (IESP) [6] is developing a technology roadmap to allow an international collaboration on the development of the software infrastructure to support an exascale machine.

Recommendation: *NSF should support the development of new and sustainability of existing numerical libraries. This will provide a bootstrap for old and new application codes.*

Systems Software and Runtime Systems

The increasing heterogeneity of computer systems indicates that the software stack requires significant support from systems software and runtime systems to manage threads, schedule computations, ensure appropriate load balance across the system. The development of such systems software and runtime support is most likely to land at the feet of the hardware vendors and provides a key place for integration with the open source developments of numerical libraries, development environments, and of course application codes. The IESP roadmap recommends the use of application codes as co-design vehicles that will support this integrated approach to development of the software systems.

Recommendation: *Support is needed to enable collaboration with industry computer vendors and software developers.*

3 Data, Federation & Collaboration

In the 25 years since the establishment of the first NSF supercomputer centers, the performance of the fastest supercomputers has increased by six orders of magnitude, from 10^9 to 10^{15} floating point operations per second. This remarkable evolution and concurrent improvements in numerical methods have transformed how many disciplines study complex phenomena. Computational simulation is by now a mainstream method of scientific research and discovery, and as a result, both the number of users of computational methods and the demands for computing facilities and software has grown greatly.

In a more recent and in some respects yet more remarkable development, the past 10 years have also seen the emergence of new approaches to scientific and engineering discovery based on the analysis of large volumes of data [11]. Developments in sensor technology, laboratory automation, computational simulation, and storage technologies have together enabled a dramatic explosion in available data in many fields. Just 15 years ago, a popular book described methods for managing O (10^9) bytes (*gigabytes*) [27]. Now, individual experiments in high-energy physics can generate O (10^{15}) bytes (*petabytes*) per year; in other communities, individual instruments may generate terabytes per day, and the aggregate volume, distributed over many facilities, also reaches petabytes. This data deluge shows no signs of slowing — indeed, the pace of exponential growth in available data appears to be accelerating, driven both by technological innovation and competitive pressures, as researchers realize that more data leads to more rapid progress. Furthermore, the fact that data relevant to a problem solution comes from multiple sources, collected in different modalities, timescales, and formats, adds to the complexity. One of the greatest scientific and engineering challenges of the twenty-first century is to understand and make effective use of this growing body of information.

A third wave of change is the continued commoditization of computing, resulting in the emergence of both ever-more powerful campus computing resources, or resources consisting of multiple heterogeneous, interconnected computing platforms that may be geographically distributed, or resources such as those supported by commercial "cloud" computing service providers that leverage new economies of massive scale. The success of simulation and the data explosion drive an expanding need for computing; these technological and business model developments mean that this need can increasingly be met outside traditional supercomputer centers.

These developments demand a new view of cyberinfrastructure. No longer is it sufficient to focus attention on making a few high-end supercomputers usable by a relatively small number of expert users. Instead, a 21st Century cyberinfrastructure must recognize that computing will be performed by many more people, in many places; that for many researchers, data analysis will be as important as, or even more important than, simulation; and that research and innovation will become increasingly distributed and collaborative.

These developments have particularly challenging implications for the nature of scientific software, and the scientific community's needs for improvements in how that software is developed and supported. (It is noted that many of these needs have been identified by Jim Gray [10] in talks and papers delivered before his untimely disappearance.) In the following, we first review the nature of software needs, and then propose approaches to meeting those needs.

Changing Software Needs

Exponentially bigger data demands a fundamental change in the scientific work process. For example, not too long ago, a graduate student in biology might spend several years sequencing a single gene, producing ultimately a few thousand bytes of data. While information technology was used in this research, it was not the rate-limiting factor. Today, a graduate student can produce billions of base pairs per day using modern sequencing machines. The collection, reconstruction, integration, management, sharing, analysis, and re-analysis of this data, and also associated modeling tasks (e.g., hypothesis testing for statistical inference)—and the management of many such tasks over time—are the rate limiting steps in the research process. Each step typically requires sophisticated software—software that often does not exist (old standbys, such as Excel and LabView, can no longer cope), but that is beyond the skills and resources of the typical research group to create.

Another problem that will face almost all scientific disciplines in the future is the need for intelligent text mining to extract semantic information from the huge and growing literature. Hand annotation of both textual information and experimental data can only hope to reach a tiny percentage of the literature and data currently 'published'. Tools and technologies are therefore needed to automate the extraction of semantic information from text and data as well as tools to assist in annotation to capture the provenance of data sets. Application workflows are likely to become increasingly important as both experiments and simulations become increasingly complex and multidisciplinary.

The generation of large quantities of data also has a second-order effect on the scientific work process. As more data becomes publicly available, research increasingly often involves analysis of data produced by others. The research process frequently also becomes more collaborative. Again, significant software challenges emerge, relating for example to discovery, access, sharing, integration, analysis, and correlation (fusion) of data from multiple sources and locations, and the protection of data from unauthorized access and tampering.

Recommendation: *Funding agencies should also encourage agreement by the different research communities on ontologies, shared vocabularies, and data formats specific to their research fields.*

Agreement on such things will be important for the exchange and reuse of data by different researchers and by multiple research communities. Just as there is a social issue with recognizing computational science as a valid discipline worthy of academic rewards, so too, there is a need for recognition of data curators and data archivists who make possible the preservation and reuse of data.

The impact of the aforementioned changes on cyberinfrastructure needs is seen clearly in large instrumentation projects. Software is needed to cope with distributed generation and use of data tools for large scale data federation across multiple instruments and users. Software costs now dominate capital expenditure in many such projects. For example, in ground-based astronomical sky surveys, software costs may be one-quarter to one-half of total budget. Software constitutes 10% of the cost of the Ocean Observatory Initiative. Participants in such projects nevertheless complain of inadequate software and assert that software budgets cover only basic system operations, not the equally important work of data analysis.

Outside the relatively narrow world of big experiments, researchers across all NSF programs report tremendous problems with all aspects of the data pipeline. Groups with substantial internal expertise and resources assemble their own one-off solutions. Others fail to deal with the problems of managing and analyzing large datasets. Data sharing and

Software is needed to cope with distributed generation and use of data tools for large scale data federation across multiple instruments and users.

collaboration prove to be persistent problems. Across the board, there seems to be both large underinvestment and substantial duplication of effort.

The data explosion demands an extensive set of new tools and technologies so that researchers cannot only make sense of their own experimental data, but can also build on existing data and develop technologies to support reproducible research. Ease of use must be a priority, given the broad candidate user community. Ideally, these tools will be as easy to use as today's Web 2.0 technologies: blogs, wikis, social networks, tagging, RSS feeds and so on. In addition, semantic technologies to assist scientists in discover and aggregation of relevant datasets and tools that build on this semantic infrastructure to allow computer-assisted inference and interpretation of such combined datasets will be necessary to support the chain of data to information to knowledge. It is also likely that some or most of these tools and technologies will have a Cloud component and may involve the use of commercial Cloud services. The tools must also allow for a variety of different levels of security and different security technologies in setting up Grand Challenge collaboratories/Virtual Organizations. All of these technologies will be needed to build a powerful and intelligent cyberinfrastructure for the next generation of scientific challenges.

***Recommendation**: The data deluge requires an increased level of support for software development in the areas of data, federation, and collaboration—areas that have historically received less support than high-performance computing software. NSF should establish new programs to support software development and support in these areas. These programs should be designed to support the needs of not only high-end applications dealing with petabytes, but also the thousands of small laboratories struggling with terabytes.*

Data Requirements

Data collection. New sensors of many types are transforming data collection in many fields. Reliable software is needed for operating large numbers of sensors, collecting data from such sensors, synthesizing derived data from sensor output, and other related tasks. Such software needs to be developed in a more organized fashion to enable broad and robust use.

Laboratory instrumentation and management systems. There is a need for generic components to build Laboratory Information Management Systems (LIMS) that can be adapted to the needs of specific research fields and requirements. Integration of these systems with data curation, annotation, and analysis functions is becoming increasingly important.

Data modeling, semantics, and integration. In a more data-intensive research future, different data sets from different communities must often be compared and combined with new data in a variety of "scientific mash-ups." Integration of simulation data with experimental data as is common in the climate and weather modeling communities will also become increasingly important. Software is required to assist with data modeling, with the representation and exchange (and automated extraction) of semantic information, and with the mechanics of large-scale data integration.

Software is required to assist with data modeling, with the representation and exchange of semantic information, and with the mechanics of large-scale data integration.

Data management and analysis. Software for managing large quantities of data of different types, and for enabling compute- and data-intensive computations on that data, are key requirements in many fields. Technologies such as distributed file systems (e.g., HDFS, PVFS, Sector), data-parallel languages (e.g., MapReduce, Sphere), and parallel scripting languages (e.g., Swift [26]) have important roles to play, but will require considerable extension to deal with the challenges of next-generation data and analysis.

Data mining and statistical inference. Confronted with large quantities of noisy data, researchers turn to data mining and statistical

inference procedures to identify meaningful patterns. While research is needed to advance fundamental methods, there is also an urgent need to create scalable, usable implementations of known methods. Existing libraries of data analysis algorithms, such as those associated with the open source R software, have been tremendously useful, but do not scale to terabytes and petabytes of data. New efforts are required to build out scalable software systems that will facilitate the use of the most modern algorithms.

Text mining. A problem that will face almost all scientific disciplines in the future is text mining to extract semantic information from the huge and growing literature [20]. Hand annotation of both textual information and experimental data can only hope to reach a tiny percentage of the literature and data currently 'published'. Tools and technologies are therefore needed to automate the extraction of semantic information from both text and data as well as to assist in annotation to capture the provenance of data sets.

Workflows. Scientific workflows, which provide for the expression, invocation, documentation, and exchange of mashups, are likely to become increasingly important as both experiments and simulations become increasingly complex and multidisciplinary. Open source systems such as Kepler and Taverna have proven popular. Such systems need to be developed further to increase ease of use, support scaling to larger problems, address other aspects of the scientific discovery process, and incorporate provenance recording capabilities.

Visualization Requirements

The human visual system is an extremely powerful tool for discerning patterns and identifying features in data. Visualization tools play an important role in scientific data analysis. Existing tools provide powerful capabilities, but as in other areas, the need to deal with exponentially larger data volumes and to integrate across more data sources of different types leads to new challenges that current software is not capable of addressing. Visual data analysis, facilitated by interactive interfaces, enables the detection and validation of expected results while enabling unexpected discoveries in science. It allows for the validation of new theoretical models, provides comparison between models and datasets, enables quantitative and qualitative querying, improves interpretation of data, and facilitates decision-making. Scientists can use visual data analysis systems to explore "what if" scenarios, define hypotheses, and examine data under multiple perspectives and assumptions. They can identify connections between large numbers of attributes and quantitatively assess the reliability of hypotheses. In essence, visual data analysis is an integral part of scientific problem solving and discovery. This is far from a solved problem and many avenues for future research remain open and discussed in [13].

Federation and Collaboration Requirements

The most interesting data is usually elsewhere. Thus research depends on the ability to discover, negotiate permissions for, access, and analyze distributed data. The instruments that produce data and the computers used to analyze data may also be remote. To support these modes of use, tools are needed to address authentication, authorization, resource discovery, secure resource access, and other resource federation functions. These tools can build on and must often integrate commodity solutions (e.g., OpenID and SAML for authentication) but often require specialization for specific scientific use cases.

Systems such as the cancer Biomedical Informatics Grid (caBIG), Biomedical Informatics Research Network (BIRN), Earth System Grid (ESG), Open Science Grid (OSG), and TeraGrid involve large-scale deployments of resource federation (a.k.a. "grid") tools based on Globus and other software. Looking forward, we see a need for large increases in scale, deeper integration with campus infrastructures, and considerable improvements in performance, functionality, and usability, as the number of users of distributed cyberinfrastructure grows rapidly. The scale of unmet need in these areas is enormous.

To further encourage resource federation, funding agencies should also encourage agreement by the different research communities on ontologies, shared vocabularies and data formats. Agreement on such things will be important for the exchange and reuse of data by different researchers and by multiple research communities.

Establishing a Quantitative Basis for Resource Allocation Decisions

Given the reality of far more software needs than can feasibly be supported by government funding, it is important that the NSF and other agencies have access to reliable quantitative data concerning the usage of different software systems. One may debate what data is most meaningful and how to obtain that data reliably, but it seems uncontroversial to assert that data is useful. A few existing systems, notably Condor and Globus, incorporate usage reporting mechanisms that provide detailed data on how, where, and when their software is used, but for the most part, the only quantitative data available is counts of software downloads, a highly unreliable predictor of actual usage.

Quantitative usage data can also help inform software development and support activities, for example by showing what features of software are most used and what sorts of failures occur most frequently.

Recommendation: *Require that NSF-supported software incorporate automated usage reporting mechanisms to provide accurate data on usage. In this way, we can enable quantitative comparisons of the extent that different software systems are used within the NSF community.*

4 Software Evolution

All software must evolve to keep up with changes in systems, usage, and to include new algorithms and techniques. The scientific community has an interest in ensuring that the software it needs will continue to be available, efficient, and employ state-of-the-art technology. A recent NSF report [24] has defined sustainable software as software that is well-maintained and available for current systems. In supporting software evolution, we extend the concept of sustainable software to include the changes that are made to software to keep it effective and relevant; this often means adding new features, changing the structure to better fit new system architectures, and the adoption of new algorithms and techniques. This requires more than simply ensuring that existing software remains available as systems change. In this section, we consider options for supporting the evolution of software. We begin by considering open source software separately from proprietary and closed source software. Both are important, but each will require a different approach.

Supporting Open Source Software

Software codes exist but need to be maintained and updated as science and systems evolve. Successful scientific software, with a large community of users, typically has a life cycle of several decades. Such software thus spans multiple generations of hardware requiring constant updates to adapt to new architectural features including ever increasing degrees of parallelism at multiple granularities or heterogeneity, such as clusters with mixed commodity CPUs and GPGPUs. Additionally, software requires revisions and updates to provide enhanced functionally through the incorporation of improved models, algorithms or emerging techniques for sensitivity analysis, optimization and uncertainty quantification. By considering scientific software developed over the last several decades, we can identify three broad classes of open source software.

1. **Orphaned Software**. This is software that is no longer being developed and has no one interested in owning or developing the software. However, this software is still in use and serving the needs of the scientific community.
2. **Prototype Software**. This is software that is good enough to test an approach or illustrate a method, but it is not at a stage where the broader community can use it. Such software (also known as research software), is typically not well tested, and lacking well defined interfaces and user guides. Such software is still being developed or used by its developers; otherwise, it would be in the Orphan Software category.
3. **Healthy Software**. These are robust software codes, often developed by groups that are used by the broader scientific community. They may offer functionality that spans multiple scientific domains, such as middleware for performance modeling or data integration, or offer functionality required to promote scientific inquiry in specific domains along narrow themes.

Open source software sustainability requires support for the entire software lifecycle including effective and differentiated pathways for managing the three classes of software. Orphaned Software should be “re-homed” into a group that can provide support for it. Further developments should be focused on maintaining

the software in usable form to support the needs of its user base through activities to allow porting to new platforms and associated tuning, testing, and documentation. Two obvious models exist for funding support of orphaned software: either a direct funding to the group that supports the software or a support agreement paid by the users of the software.

Prototype Software that is still under development by its authors requires considerable assessment of its future potential to determine the type of support that may be most appropriate. For example, if the software reflects significant advances in algorithms or methodology but its potential for broad use is difficult to assess, it may be best to provided targeted support to the developers to improve the quality of the software. One approach is to use a *software institute* where the developers can bring their code, work with experts (and expert tools) to analyze, understand, and update their code to modern software engineering standards. The developers then take that code home and continue to work on it, but with a new understanding of software engineering principles and methods that could eventually enable its transformation into Healthy Software. An alternative is to create a *branch* of the code – freezing the development at a certain point. This approach deliberately creates Orphan Software, which can be supported as described earlier.

All software must evolve to keep up with changes in systems, usage, and the inclusion of new algorithms and techniques.

Healthy Software, which is in use in the scientific community while it continues to be developed, requires significant support for sustainability. The development of such software often involves groups of faculty and students. Now the central support needs concern the development effort required to update the software to enable its use on new and emerging hardware and the incorporation of new "state-of-the-art" algorithms and methodologies. These projects could benefit from support for scientific programmers or software engineers who can continue to maintain and engineer the software for effective use by the community, while faculty and students focus on research toward new advances. Although it is extremely important to avoid the use of public funds to develop software that competes with commercial code, significant support for software development staff is often vital for promoting open software evolution and use. For example, when such software provides functionality for a relatively small or specialized community or it addresses systems issues associated with new leadership class hardware, significant support may be needed for further development given the complex nature of underlying problems or the constant need to update software to maintain its "state-of-the-art" quality. If the usage landscape for such software changes over time, its eventual commercialization could be enabled and encouraged through appropriate licensing modes.

All three classes of software could benefit from common infrastructure and standards. For example, centralized mechanisms to provide bug tracking and initial problem determination can be a valuable service for both developers and users. Additionally, common infrastructure in the form of test systems, including both legacy systems for testing backward compatibility and new and highly parallel systems to enable new software updates and advances. Additionally, access to version control and collaborative software development frameworks can help promote sound software engineering practices, and group engagement and collaboration. Finally and most importantly, NSF should support and promote the development of community standards that are critical for enabling wider software development and use. For example, NSF provided support for travel to the MPI Forum meetings, which resulted in the MPI standard that has been pivotal in the broader development and use of parallel codes. NSF should support such community infrastructure and standardization efforts at multiple levels, including support for the formation of standards forums in a variety of topical areas, provisioning of shared community infrastructure for software lifecycle management

and support for conferences, workshops and training events to promote software evolution and sustainable open source software for science.

Open Source Software, Intellectual Property, and Industry-University Collaboration: Challenges and Opportunities

The landscape of science has been significantly changed by the rise of open source software, such as the LAPACK numerical library, the MPICH message-passing library, the VTK graphics toolkit, Grid collaboration toolkits such as Condor and Globus, and cluster toolkits such as ROCKS and OSCAR. Such open software has led to many large-scale national and international research projects that depend on sharing of software infrastructure across tens of institutions and hundreds to thousands of individuals, posing particular challenges in software sharing and licensing. Negotiating myriad institutional licenses has typically proven intractable, and almost all such projects have adopted some version of an open source software model, often a variant of the "BSD model" (derived from the original University of California at Berkeley license for UNIX), which allows reuse in new and diverse ways. We note there are several open source software license options [21]; the choice of which open source license to use is sometimes not obvious. Several open source licenses allow universities and researchers both to foster collaboration and sharing in addition to retaining the option to generate license revenues, create protectable intellectual property, or generate proprietary software from research software. With other open source licenses, such as the GPL or the QPL, the options for commercialization and intellectual property protection can sometimes be more complicated — but still often possible.

The landscape of science has been significantly changes by the rise of open source software.

***Recommendation**: NSF should recommend open source distribution of software developed through its programs, while also recommending that grantees be aware of different open source license options and requirements at the grantee's home institution.*

Supporting Commercial Software

Building an entire cyberinfrastructure on open source software is likely to be difficult to create and sustain. A "mixed source" approach that utilizes the strengths of commercial software companies in significant areas is much more likely to lead to a sustainable and affordable software infrastructure. Some key pieces of software may only have commercial versions or the current commercial versions may be substantially better than open source versions. Many would argue that compilers fit this description; parallel debuggers certainly do. Particularly for high-end platforms (such as the NSF Track 1 and Track 2 systems), there are many demanding and unique problems that are not faced by the commodity market. In such instances, NSF and government agencies have an important role to play towards supporting advances that are critical to promote their scientific missions. For example, NSF could fund customization to meet the needs of its scientific community; in the recent past, DOE has taken this approach with Totalview. More simply, NSF could pay for software much as it does for specialized hardware and other research instruments. Alternatively, NSF could seek to promote innovation, that is, fund the development of novel approaches rather than seek complicated extensions to existing software to fit highly specialized instances. Such funding could go to either commercial, non-commercial providers or industry-university partnerships with appropriate contractual and licensing agreements to enable both commercialization and broader use for research and education.

***Recommendation**: It is important to recognize that NSF should carefully avoid using public money to fund software efforts that compete unfairly with private industry. However, in cases where there is no viable market, such is often the case for specialized software for massively parallel computers, NSF (and other*

agencies) may need to support open source software, using one of the mechanisms described.

Collaboration and Software Development

Many software projects involve multiple groups; these groups are often geographically separated. Particularly for the "prototype" open source software, these groups often need to interact frequently. In many cases, this interaction makes use of the simplest tools – email and a source code control system. While there are many collaboration tools available, these are rarely used by groups who are creating (or extending) software as part of their NSF-funded activities. As an adjunct to the software center concept, there is an opportunity to develop and/or adopt more integrated tools to enhance collaboration among software development groups and between the software developers and their users.

5 Institutional Barriers

Researchers in the area of software spend significant time in the initial development of software, for which the focus is on the instantiation of a new idea, the widespread use of some software infrastructure, or the evaluation of concepts for a new standard. Correspondingly, the evaluation of effort associated with software needs to include the user base for a given software infrastructure, participation and publication within a software standards effort, or the level of maintenance required to evolve software to satisfy the dynamic needs of the user community. It is the case, however, that the conventional criteria of conference and journal publications and research grants are often used when evaluating research in the area of software. There is a lack of well-developed metrics for the impact and quality of scientific software. Unlike universally accepted citation-based metrics of papers, published citations of software is rarely practiced.

***Recommendation**: NSF should work with institutions to identify appropriate metrics that match the effort necessary for successful development and maintenance of scientific software.*

Requirements for cyberinfrastructure software are not unique to the US: similar needs arise in every country with an advanced scientific research and education enterprise. Common solutions to these needs can have two major advantages: they reduce total costs, by avoiding redundant effort, and they facilitate international cooperation within scientific disciplines that use these solutions, by reducing barriers due to different software systems. These factors are important in every area of cyberinfrastructure, but are particularly compelling in the case of data, federation, and collaboration software due to the frequently international nature of the associated collaborations. For example, in climate research, different teams worldwide build their own earth system models, but all teams cooperate on the comparison of simulation output from different models with each other and with observations.

***Recommendation**: Cooperation on cyberinfrastructure software development can both reduce total costs by reducing redundant effort and reduce barriers to scientific collaboration by avoiding the creation of noninteroperable software silos. However, such cooperation can be difficult because of different priorities and funding cycles. NSF should work with foreign funding bodies to establish programs that incentivize cooperative development and support for cyberinfrastructure software.*

6 Other Agencies

It is noted that many federal agencies and international agencies have significant investments in science software infrastructure. In the sections below, we summarize some of the software programs with these agencies.

Air Force Office of Scientific Research (AFOSR)

The Air Force Office of Scientific Research is the basic research component of the Air Force Research Laboratory. As such, AFOSR fosters fundamental research advancing the state of the art in a number of areas critical to continued dominance of air, space, and cyberspace. Within the broad scope of transformational opportunities supported by AFOSR, areas relevant to the scope of the present report include research in mathematical and computational sciences, and in information and computer sciences, aimed to identify, discover, and further foundational knowledge, for addressing challenges and creating new capabilities underlying the design, implementation and deployment of complex systems relevant to the Air Force.

Within this context, and related to advances in new approaches and capabilities for such systems, disciplinary as well as multidisciplinary research priorities include: new mathematical computational methods; advanced software methods for systems engineering and support of dynamic and complex adaptive systems that need to be highly-autonomic, composable, and evolvable; adaptive management of heterogeneous systems of sensors, and data and information fusion; dynamic integration of real-time data acquisition and control with advanced multiscale application models, and other dynamic data driven applications systems. Support environments of such systems require advanced, multicore-based computer architectures, computing models, and dynamic runtime environments, comprising of complex distributed computational, communication, and data acquisition and control platforms, ad-hoc, mobile, complex and heterogeneous communication networks, and distributed sensing networks.

Research to address such challenges and enable breakthrough advances is funded through AFSOR programs supporting academic research as well as fundamental and basic research in Air Force Research Laboratories. In addition, such research is often conducted in collaboration with other entities in DoD and other agencies supporting fundamental research, such as NSF, DOE, NASA, NOAA, NIST, etc. Programmatic modalities include programs supporting individual investigator projects, as well as Multidisciplinary University Research Initiative (MURI) programs, and technology transfer programs such as the Small Business Technology Transfer Program (STTR).

Department of Energy (DOE)

The Office of Advanced Scientific Computing Research in the Department of Energy's Office of Science supports multidisciplinary SciDAC (Scientific Discovery through Advanced Computing) projects aimed at developing future energy systems, studying global climate change, accelerating research in designing new materials, improving environmental cleanup methods, and understanding physics from the tiniest particles to massive supernovae explosions. SciDAC Centers for Enabling Technologies conduct a mix of research and software development to produce software solutions required by SciDAC scientific

applications, in such areas as solvers, file systems, data management, and distributed computing.

The SciDAC program has established SciDAC Institutes that are university-led centers of excellence intended to increase the presence of the program in the academic community and to complement the efforts of the SciDAC Centers for Enabling Technologies. Under SciDAC-2, the four institutes receive a total of $8.2M annually for a period of five years. Midterm reviews of the institutes occur after three years. The Institutes assist the SciDAC Scientific Applications with overcoming the challenges to effectively utilizing petascale computing; the centers are also given the flexibility to pursue new research topics and be responsive to the broader community.

National Institutes of Health (NIH)

The National Institutes of Health have many programs that support the development of software. One such program is BISTI (Biomedical Information Science and Technology Initiative), which is a consortium of representatives from each of the NIH institutes and centers that focus on biomedical computing issues at NIH. BISTI was established in 2000. The mission of BISTI is to make optimal use of computer science and technology to address problems in biology and medicine by fostering new basic understandings, collaborations, and transdisciplinary initiatives between the computational and biomedical sciences. BISTI coordinates research grants, training opportunities, and scientific symposia associated with biomedical computing. Proposal for such grants must include a software dissemination plan, with appropriate timelines. There is no prescribed single license for software produced through BISTI grants. The software, however, should be freely available to biomedical researchers and educators in the non-profit sector. The terms of the software availability should permit the commercialization of enhanced or customized versions of the software, or incorporation of the software or pieces of it into other software packages. Further, to preserve utility to the community, the software should be transferable such that another individual or team can continue development in the event the software is "orphaned".

Engineering and Physical Sciences Research Council (EPSRC)

EPSRC has a Software Sustainability program that is focused on research infrastructure that aids in the long term sustainability of software that enables high quality research. EPSRC has invested £5.0M into this program. The objective can be fulfilled through the following services:

- The development of software to an acceptable quality for wider deployment, through the application of additional software engineering to prototype software delivered by UK research projects.
- The maintenance of software that enables high quality research through the management of a repository for selected software.
- Community outreach and promotion to ensure effective update of the services that the infrastructure will provide.
- Engagement with the international community through activities such as the dissemination of e-research software, establishing best practices and standards, providing internationally recognized codes.

Proposals can be funded for up to five years in duration. Applicants are encouraged to engage the National Grid Service (NGS) with their projects.

7 Summary of Findings

- Software is a critical and pervasive component of the cyberinfrastructure for science and engineering.
- Software is a tool for new knowledge discovery in many disciplines and often also itself serves as a representation of knowledge.
- NSF does not adequately support software evolution and lifecycle costs.
- Software needs to address questions of open access, portability, reuse, composability and dissemination.
- Increasing complexity of applications, and resources can only be dealt with by the availability of good software.
- There is a lack of well-developed metrics for the impact and quality of scientific software. Unlike universally accepted citation-based metrics of papers, published citation of software is rarely practiced.
- The lack of NSF support for the development of robust prototype complex software frameworks and tools often results in repetitive development that can hinder the full potential of advances in science and engineering
- Increases in code complexity for addressing the paradigm shifts in architecture (including using accelerators and multi/many core chips) threaten to exceed the capacity of the research community for software development and support.
- Fast moving developments in the information technology environment and the lack of software standards to support those developments are a major challenge for the creation of new software tools.
- Uncertainty about the computer architecture and programming model will impact the evolution of simulation software from inception to new science results.
- The software infrastructure for science is often a patchwork of supported and unsupported software in a rather ad-hoc approach. In many cases key simulation codes grow organically as a new research code is added to an existing body of code, resulting in unsustainable applications that cannot be easily verified, where error propagation from one part of the code to others may not be well understood.
- Research is required into new approaches to enable a simpler programming environment for application developers – allowing composability, portability and ease of developing software.
- A lack of standards in application programming interfaces, data models and formats, and interoperability of programming models makes the integration of different pieces of software representing different models and different phases of the workflow challenging.
- Software for operating large numbers of sensors, collecting data from such sensors, synthesizing derived data from sensor output, etc., needs to be developed in a more organized fashion to enable broad and robust use.
- Integration of simulation data with experimental data, as is common in the climate and weather modeling communities, will be increasingly important in *all* areas. Software is required to assist with data modeling, with the representation and

exchange (and automated extraction) of semantic information, and with the mechanics of large-scale data integration.

- Software is needed to manage data acquisition from experiments and its integration with data curation, annotation, and analysis functions.
- Software is needed to cope with distributed generation and use of data tools for large scale data federation across multiple instruments and users.
- Ease of use and integration of scientific and engineering software with collaborative tools will be a priority.

8 Recommendations

Summary of Recommendations

1. NSF should develop a multi-level ((individual, team, institute), long-term program of support of scientific software elements ranging from complex applications to tools of use in multiple domains. Such programs should also support extreme scale data and simulation and the needs of MREFCs.

2. NSF should take leadership with promoting verification, validation, sustainability, and reproducibility of software with federal support.

3. NSF should develop a consistent policy on open sources software that promotes scientific discovery and encourages innovation.

4. NSF support for software should entail collaborations among all of its divisions, related federal agencies and private industry.

5. NSF should utilize the Advisory Committees to obtain community input on software priorities through workshops and town hall meetings involving the broad community.

While the above summary recommendations capture the essence of the deliberations of this taskforce we provide below additional detail that may be used to interpret these recommendations.

Programs

1. Under the SSE umbrella, NSF should support both of the following types of activities: (1) projects that are headlined by actual complex applications, which use a variety of software components in a vertically integrated manner ("end-user projects/science-pull") and (2) projects that develop and apply software components that

   Typical outcomes under the SSE program, at any of its levels (single investigators, teams, centers) include the hardening of prototype software, the integration of new software technology into existing software, and engaging in cross-disciplinary production collaborations with experts in the use of the software.

   a. Under SSE technology projects/science-push, NSF should support a variety of activities, including: developing and provisioning of software components, extension of applicability of software outside of the domain of original development, porting of software to architectures outside of the domain of original development so that it operates across all relevant architectural scales, development and promotion of community standards for software, development of new and maintenance of existing numerical libraries, training in software use, training in software development, and archiving and re-homing of orphaned software of enduring importance to the scientific or engineering community.

2. Under SSE, NSF should proactively support projects that merge proven simulation methods and proven data (experiments, observations, sensor inputs, etc.) at scale.

3. Under SSE, NSF should focus attention on the data and software needs of the major NSF research facilities (MREFCs).

4. SSE project budgets should accommodate adequate human resources at the level of post-graduate software engineer, and not penalize such allocations on the basis that there is insufficient training of student researchers. Participation of student researchers should also be encouraged through aspects of such projects that relate to doctoral dissertation-worthy work.

5. Under SSE, NSF should support the development of portable systems.

Policy & Practices

6. NSF should encourage best practices in validation and verification of the software that it sponsors under the SSE, including standard software engineering best practices (error reporting, unit testing during development, regression testing, versioning, automated code analysis, argument type-checking at interfaces, performance profiling, bug tracking, etc.) as well as application-specific best practices (physically relevant test harnesses, built-in modified equation analysis, built-in sensitivity analysis, built-in Jacobian checking, built-in declarations of verifiable properties of data objects, requirements for multi-scale and multi-model integration, argument value-checking at interfaces, code instrumentation that automatically reports on usage over the internet to developers, etc.).

7. NSF should explore the legal and technical issues with respect to the different open sources licenses to encourage a consistent policy on open sources software developed under SSE and provide information on the implications of the different licenses to the researchers.

8. NSF should encourage reproducibility (e.g., detailed provenance of results and data) in all computational results that it sponsors, including the preservation of the software and data used in the research.

9. NSF should seek to offer pathways to software sustainability, as sustainability is essential to encourage end-users to stake portions of their careers on the long-term availability of important components of the software infrastructure for which there is no commercial market.

10. NSF should promote discussion amongst its own personnel and with leadership at institutions where its principal investigators are employed, to consider development and provisioning of Complex Software Infrastructures activities, as meritorious in promotions and raises.

11. NSF should develop, acquire, and apply metrics for review of SSE projects that are complementary to the standard criterion of intellectual merit. Impact remains a key metric. One of the new metrics for merit review is that the community has come together to identify one or more common needs that are met by the proposed project

Collaborations

12. NSF should foster a healthy software industry through the SSE through: (1) avoiding competition with commercial industry when adequate software already exists, (2) sponsoring the acquisition of commercial software as part of the cost of doing research when adequate software exists, (3) encouraging collaborative University-Industry innovation, and transitioning into the commercial marketplace software developed under the SSE umbrella, (4) promote close communication between chip designers, system builders, and software developers, (5) encourage the formation of public-private transitions through new and innovative partnerships between academe and industry, and (6) provide SBIR-like programs to facilitate the commercialization process.

13. NSF should seek to fund SSE projects through participation from all relevant science and engineering Directorates and Offices, not by OCI alone, in order to insure that the investments are adequately funded and correctly purposed.

14. NSF should consider novel ways of teaming with other agencies of the U.S. government that support software research and development to cross-leverage advances providing for the common open-source infrastructure(s). NSF should sustain its current strength in supporting software innovation while increasing its role in the support of the lifecycle costs of such software in partnership with mission agencies like the Department of Energy and National Institutes of Health.

15. NSF should consider novel ways of teaming with agencies of other nations that support software research and development to cross-leverage advances providing for the common open-source infrastructure(s).

16. NSF should utilize the different Advisory Committees to obtain community input on software (e.g., orphaned codes) for which there is a need for sustainability and/or evolution. The input can be obtained via workshops, web-based surveys, professional societies, etc.

References

1. Abern, S., Alam, S., Fahey, M., Hartman-Baker, R., Barrett, R., Kendall, R., Kothe, D., Messer. O. E., Mills, R., Sankaran, R., Tharrington, A., and James B. White III, *Scientific Application Requirements for Leadership Computing at the Exascale*, ORNL/TM-2007/238, http://www.nccs.gov/wp-content/media/nccs_reports/Exascale_Reqms.pdf, Dec 2007.
2. *Challenges In Climate Change Science and The Role of Computing at the Extreme Scale*, Workshop Summary Report, Department of Energy, November 6-7, 2008, http://www.er.doe.gov/ascr/ProgramDocuments/Docs/ClimateReport.pdf.
3. Committee on the Potential Impact of High-End Computing on Illustrative Fields of Science and Engineering, National Research Council, *The Potential Impact of High-End Capability Computing on Four Illustrative Fields of Science and Engineering* http://www.nap.edu/catalog/12451.html, 2008.
4. Darema, F., *Report on CyberInfrastructures of Cyber-Applications-Systems & Cyber-Systems-Softwar*e, October 2009, preprint.
5. Darema, F., *Report on Industrial Partnerships in Cyberinfrastructure: Innovation through CyberInfrastructure Excellence*, October 2009, preprint.
6. Dongarra, Jack, Beckman, Pete, Aerts, Patrick, Cappello, Frank, Lippert, Thomas, Matsuoka, Satoshi, Messina, Paul, Moore, Terry, Stevens, Rick, Trefethen, Anne, and Mateo Valero, *The International Exascale Software Project: A Call to Cooperative Action by the Global High Performance Community*, 2009.
7. *Exascale Computing Study: Technology Challenges in Achieving Exascale Systems*, http://www.sdsc.edu/~allans/Exascale_final_report.pdf*, September 2008.*
8. *ExaScale Software Study: Software Challenges in Extreme Scale Systems*," DARPA Information Processing Techniques Office, Washington DC., September 14, 2009. http://users.ece.gatech.edu/~mrichard/ExascaleComputingStudyReports/ECSS%20report%20101909.pdf.
9. *Forefront Questions in Nuclear Science and the Role of High Performance Computing Summary Report - Summary Report*," Department of Energy, January 26-28, 2009. http://extremecomputing.labworks.org/nuclearphysics/PNNL_18739_onlineversion_opt.pdf.
10. Gray, Jim, on eScience: A Transformed Scientific Method, in T. Hey, S. Tansley, K. Tolle (Eds), *The Fourth Paradigm: Data-Intensive Scientific Discovery*, 2009.
11. Hey, T., Tansley, S. and K. Tolle (Eds), *The Fourth Paradigm: Data-Intensive Scientific Discovery*, 2009.
12. *International Exascale Software Project Roadmap*, 2009, http://www.exascale.org/mediawiki/images/4/42/IESP-roadmap-1.0.pdf.
13. Johnson, C. R., Moorhead, R., Munzner, H. Pfister, H., Rheingans, P. and T. S. Yoo, (Eds.): *NIH-NSF Visualization Research Challenges Report*; IEEE Press, ISBN 0-7695-2733-7,2006. http://vgtc.org/wpmu/techcom/?page_id=11.
14. Keyes, D. E. (Ed.). *A Science Based Case for Large Scale Simulation.* Office of Science. Department of Energy, 2003.
15. *Mathematics at the Interface of Computer Science and Industry, the Smith Institute for Industrial Mathematics and System Engineering,* Smith Institute for Industrial Mathematics and System Engineering, Oxford, UK, 2005.
16. Oden, J. T. (Ed.), *Report of the NSF-ACCI Task Force On Cyber Science and Engineering, Grand Challenge Communities and Virtual Organizations,* 2010.
17. Oden, J. T. (Eds.), *Revolutionizing Engineering Science through Simulation*, A Report of the NSF Blue Ribbon Panel on Simulation-Based Engineering Science, 2006. http://www.nsf.gov/pubs/reports/sbes_final_report.pdf.

18. *PETSc, Portable, Extensible Toolkit for Scientific Computation*, http://www.mcs.anl.gov/petsc**.**
19. Report to the President: *Computational Science: Ensuring America's Competitiveness*, President's Information Technology Advisory Committee, 2005.
20. Rzhetsky, A, Seringhaus, M, and M. Gerstein*, Seeking a new biology through text mining*, *Cell*. 2008 Jul 11;134(1):9-13.
21. St. Laurent, Andrew M., *Understanding Open Source and Free Software Licensing*, O'Reilly, 2004.
22. Sarkar, V., Harrod, W., and Allan Snavely, *Software challenges in extreme scale systems*, *Journal of Physics: Conference Series*, 2009, http://stacks.iop.org/1742-6596/180/012045.
23. Simon, H., Zachariah, T. and R. Stevens (Eds.), *Modeling and Simulation at the Exascale for Energy and the Environment*, June 2007, http://www.sc.doe.gov/ascr/ProgramDocuments/Docs/TownHall.pdf.
24. Stewart, Craig A., Almes, Guy T. and Bradley C. Wheeler (Eds.): *Cyberinfrastructure Software Sustainability and Reusability:* Preliminary Report from an NSF-funded Workshop Held 27-28 March 2009.
25. Trefethen, A. E., Higham, N. J., Duff, I. S., and P. V. Coveney, (Eds.) *Developing a High-Performance computing/numerical analysis roadmap, ,* 2009, http://www.oerc.ox.ac.uk/research/hpc-na.
26. Wilde, M., Foster, I., Iskra, K., Beckman, P., Zhang, Z., Espinosa, A., Hategan, M., Clifford, B. and I. Raicu, *Parallel Scripting for Applications at the Petascale and Beyond. IEEE Computer*, *42* (11). 2009, 50-60.
27. Witten, I., Moffat, A., and T. Bell, *Managing Gigabytes*, 1994.

http://www.nsf.gov/od/oci/taskforces