Center for Technology for Advanced Scientific Component Software (TASCS)

A Proposal Submitted to the DOE Office of Science

Program Announcements:	LAB 06-04 and DE-FG02-06ER06-04
Program Area:	Center for Enabling Technology
Program Office:	Office of Advanced Scientific Computing Research
Technical Contact:	Frederick C. Johnson

Applicant

Institution	*Principal Investigator*
Oak Ridge National Laboratory	David E. Bernholdt
PO Box 2008, MS 6016	(865) 574-3147
Oak Ridge, TN 37831–6016	bernholdtde@ornl.gov
Field Work Proposal:	ERKJD17

Participating Institutions

***Lead PI:* David E. Bernholdt, Oak Ridge National Laboratory**

Institution	*Institutional Lead co-PI*
Argonne National Laboratory (ANL)	Lois Curfman McInnes
Binghamton University (BU)	Madhusudhan Govindaraju
Indiana University (IU)	Randall Bramley
Lawrence Livermore National Laboratory (LLNL)	Gary Kumfert
Los Alamos National Laboratory (LANL)	Craig Rasmussen
Oak Ridge National Laboratory (ORNL)	James A. Kohl
Pacific Northwest National Laboratory (PNNL)	Jarek Nieplocha
Sandia National Laboratories (SNL)	Rob Armstrong
Tech-X Corporation (Tech-X)	Svetlana Shasharina
University of Maryland (UMD)	Alan L. Sussman
University of Utah (UU)	Steven G. Parker

Important Note

This is the technical portion of the TASCS proposal, as originally submitted to DOE in March, 2006. The project was funded at 25% less than the proposed amount, and the scope of work was reduced accordingly. The attached appendix provides a brief description of the differences between the funded project and the proposal. Please contact the Lead PI (above) with any questions.

Contents

Abstract

The initial SciDAC initiative developed the Common Component Architecture (CCA) and brought the benefits of component-based software engineering to high-performance scientific software. Scientific teams who have adopted the CCA are now realizing the advantages of this extensible environment, which facilitates software interoperability within and across scientific domains, addressing issues in programming language interoperability, domain-specific common interfaces, and dynamic composability. Teams increasingly report that the CCA has become integral to the future of their science.

We propose to extend the software component methodology, in close collaboration with a number of key application projects, through an interlinked series of initiatives, leveraging the component environment to develop powerful new capabilities. The initiatives focus on coupling parallel simulations, supporting emerging hardware and software paradigms for petascale computing, enhancing software quality and robustness, and dynamically adapting applications. We will continue to enhance the core CCA software environment, with emphasis on improving usability, and we will build a component ecosystem to provide more off-the-shelf components. Outreach activities include tutorials and other educational activities as well as collaborations with numerous applications, Centers for Enabling Technology, and Institutes.

1. Project Description

1.1 Background and Significance

Computational scientists face ever-increasing challenges in creating, managing, and applying simulation software to scientific discovery. These challenges, which arise from the growing complexity of the scientific problems and the rapid advances and increasing diversity in hardware platforms, impact researchers' productivity throughout the life cycle of their scientific software.

The next generation of scientific applications will be larger and more complex, and will require contributions from more diverse groups of developers; coupling simulations across multiple time- and length-scales will become the norm rather than the exception. These simulations will run on more complicated and diverse hardware platforms. The Department of Energy (DOE) Office of Science Strategic Plan [1] calls for a 100-fold increase in facility capabilities from 2004 to 2007, and full petaflop capabilities for open science by 2012; the President's FY2007 Budget Request would deploy a 500 TF system by 2008 [2]. Systems with $\mathcal{O}(10^4 - 10^5)$ processors are already being deployed [3]. This evolution will transform component technology from a useful tool for forward-thinking software developers to an indispensable strategy across the entire spectrum of computational science.

In other areas of computing, component-based software engineering (CBSE) is now widely used as a tool to help software and application developers address large-scale, complex software systems that must evolve over time and across platforms. The fundamental premise of CBSE is the organization of software into *components* with well-defined functionality which interact with each other through explicitly defined *interfaces*. The complexity of individual software modules is thus encapsulated, and of concern only to the implementer, while users of the components, need to worry only about the interfaces through which they interact with other components. This modularity helps software developers focus on one component at a time, as a major aid in addressing the complexity of large software systems. However, component-based approaches are not merely an aid to software *development*, but also provide benefits across the entire life cycle of simulation software. Their modularity simplifies group- and community-scale collaboration on developing and using software and facilitates reuse and interoperability of code across multiple applications. The component paradigm allows easy adaptation of software in response to performance, algorithmic, numerical quality, and other concerns both prior to and *during* execution. The evolution of software, including complex challenges such as coupling with other simulation software are also more natural in component environments. All of these are important and growing challenges as the computational science community seeks to expand its capabilities by harnessing coming petascale systems and enhancing the size, scope, and fidelity of their simulations.

This proposal builds upon and extends the work of the Center for Component Technology for Terascale Simulation Software (CCTTSS), sponsored by the Scientific Discovery through Advanced Computing (SciDAC) program (2001–2006). The CCTTSS has developed a component model tailored to the needs of high-end scientific computing (the Common Component Architecture (CCA)) and provided a reference implementation of the CCA environment. In collaboration with library developers, the CCTTSS has begun to establish a toolkit of components based on widely-used scientific libraries. Applications in a broad range of scientific disciplines, supported by the DOE and other agencies, have begun to adopt CCA technology in order to gain the numerous benefits that a component approach offers high-performance computing (HPC) scientific computing. The CCA has become recognized throughout the high-performance scientific computing community as the component architecture of choice.

Our vision for TASCS is to build upon and extend the established core of the CCA to meet the needs of

next-generation computational science software projects. Strongly guided by the needs of our application partners and collaborators, we will make the CCA environment more accessible and user-friendly, especially for those who employ only a subset of the CCA's features, and we will expand the component "ecosystem" to provide a richer selection of "off the shelf" components for use in the assembly of applications. We will work with our application partners on a number of technology initiatives, designed to leverage the component environment to provide applications with new and enhanced capabilities, such as (1) computing at the ultrascale, include a generalized approach to parallel coupling of disparate simulation codes, (2) automated and dynamic behavior to insure computational "quality of service", and (3) tools and techniques for writing better scientific software.

1.2 Results of Prior Support

Since 1998 the CCA Forum has focused primarily on the research and development to bring CBSE to high-performance scientific computing. Our Common Component Architecture (CCA) [4, 5] specifically addresses requirements of the scientific community, including performance [6–10], support for parallel [11] and distributed [12–19] computing models, language support and interoperability [20], and ease of adoption [21] which are not adequately addressed by "commodity" component models widely used in other areas of computing [22–24]. Our reference implementations provide environments to create and run CCA applications. And we have worked with numerous application groups to incorporate CCA technology into their applications. The net result is that the core CCA technology is now established, available, and being used in an increasing number of applications. Dramatic benefits of CBSE have been demonstrated in some applications, and a growing number of groups are building the CCA into their future software development plans. The CCA community is rapidly growing beyond the DOE-funded project that launched it.

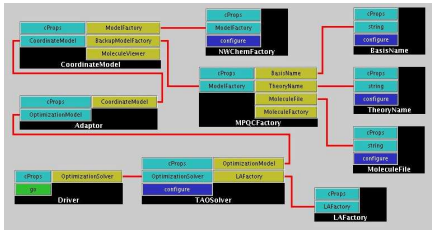

Figure 1.1: Chemistry application in the Ccafe GUI.

Defining a Component Architecture for High-Performance Scientific Computing. The CCA specification [11] defines the architecture and is designed for simplicity and minimalism. It defines the rights, responsibilities, and relationships among the CCA: software *components*, the *ports* or interfaces through which they interact, and the *framework* in which components are assembled into applications and executed.

CCA's minimalism implies that a well-structured piece of software must implement just one new method to become a CCA component. The CCA defines only procedural interactions between ports, and does not impose a particular parallel programming model on applications, making it more easily incorporated into existing parallel applications. Components using *different* parallel programming models can coexist [25], as demonstrated in an application simultaneously using MPI- [26], PVM- [27] and Global Array-based [28, 29] components. The port model for CCA components transparently accommodates both local high-performance and distributed implementations.

The CCA specification is at version 0.7.8, has been stable since April 2004, and with additions proposed here will meet our criteria for a "1.0" release. We see backward compatibility as essential and minimize the impact on existing users when modifying the specification (for example, the recently added `ComponentRelease` mechanism).

Developing the Core CCA Software Environment. A CCA-compliant framework allows components to be loaded, assembled to create scientific applications, and executed. The CCTTSS has developed frameworks targeting different computing environments. Ccaffeine [11], led by SNL, emphasizes HPC parallelism, with components located in the same address space passing arguments by reference rather than copying, and support for both Graphical User Interface (GUI) (Fig. 1.1) and scripted application composition. XCAT [18, 30–32] focuses on distributed computing environments, using the Proteus multi-protocol

library [33, 34] as the transport layer, and provides a bridge between CCA and web-services [35, 36]. The SCIRun2 [37, 38] framework supports a multi-threaded parallel programming model and a meta-component system for integrating CCA components with other component systems [39–43]. One element of this proposal is establishing standard to insure interoperability between CCA frameworks to accommodate mode complex computing environments.

The second major element of CCA is the Babel language interoperability tool [44], using the Scientific Interface Definition Language (SIDL) for programming language neutral expression of interfaces. Based on SIDL descriptions, Babel provides the glue (generated code and supporting libraries) so that and combination of supported languages (currently C, C++, Fortran77, Fortran95, Java, and Python) can be intermixed [20] in a single address space, without relying on messaging layers or interpreted middleware for maximal performance [45]. Because of Babel's central role in the interactions between components, it is key for the introduction of other new capabilities, such as the definition and enforcement of interface semantics [46–48], and remote method invocation (RMI) capabilities [49], both of which are to be extended in this proposal.

In addition to these is a collection of smaller tools and conventions to help configure, build, and install components. Collectively referred to as the "CCA base installation", this is a distillation of our experience incorporating CCA into existing applications and platforms. Because of the complexity of the configure/build/install process for large-scale applications, especially those involving multiple programming languages and many external libraries, these tools are continually improved. Preliminary work has also been done on the use of Integrated Development Environments (IDEs), such as Eclipse [50], to work with CCA software. These activities will play an important role in the new proposal.

CCA Components and the CCA Toolkit. The CCTTSS has begun an effort to establish a "component ecosystem" as an aid to users adopting the CCA. This effort has encompassed working with various communities to develop common interfaces to facilitate interoperability of libraries across a discipline [51, 52], including TOPS [53] solver components [54] and TSTT [55] mesh interfaces [56], as well as providing a collection of CCA components in the form of a general toolkit. Though not yet formally distributed, the CCA Toolkit currently contains 16 components [57], contributed on a volunteer basis and typically derived from CCA applications. Part of the proposed work is to broaden and extend the Toolkit, including more general utility components. Preliminary versions of these components were recognized by the DOE Office of Science as one of the top ten scientific achievements of 2002 [58, 59].

Changing the Way Application Scientists Approach their Software. The CCTTSS has worked with a broad range of applications, both SciDAC and others, to incorporate CCA technology. Current areas include astronomy, astrophysics, biological and medical simulations, chemically reacting flow, climate and weather modeling [60], combustion [61, 62], computational chemistry [63–65], data management [66, 67], optimization [65, 68], fusion and plasma physics modeling [69], linear algebra [70, 71], PDE solution [43, 51, 59, 72], materials science, molecular electronics, nanoscience, nuclear power plant simulations [73], load-balancing structured adaptive meshes [74, 75], unstructured meshes [51, 56], and visualization. These adopters are realizing a variety of benefits from the CCA. For some, the CCA makes geographically distributed collaboration on a large software project more manageable and more productive. For others, CCA helps scientific communities realize synergies from collaboration to create software that interoperates beyond the bounds of individual projects. Other groups focus on the CCA's language interoperability, or use it to facilitate the coupling of existing codes into multiphysics simulations. To illustrate, we briefly expand on just two examples of applications.

The SciDAC Computational Facility for Reacting Flow Science (CFRFS) project [76] has used the CCA to develop a toolkit for flame simulations. The toolkit currently includes more than 60 distinct CCA components, which can be assembled to carry out a broad range of simulations [62, 77]. Habib Najm, the lead PI, states [78],

"The CCA has provided us with a great framework for developing and maintaining reacting

> flow codes. By providing well-defined interfaces and encouraging a code structure based on a hierarchy of interchangeable, lightweight, the CCA-based CFRFS toolkit has resolved significant challenges with handling reacting flow code complexity."

Componentization also allows CFRFS to use state-of-the-art externally developed software and has helped combustion researchers extend its capabilities. Swapping out a *single* component allowed them to introduce higher-order discretizations, giving drastic reductions in both resolution capability and simulation time [62, 79].

Two groups of quantum chemistry researchers are using the CCA to extend the capabilities of the NWChem [80–82] and MPQC [83, 84] codes in ways they would not have considered feasible otherwise. A collaboration of PNNL, SNL, and ANL researchers used the CCA to access state-of-the-art optimization algorithms in the Toolkit for Advanced Optimization (TAO), and parallel data management in the Global Arrays package, to reduce run times up to 43% for determining a molecular structure [63]. The chemistry team is working with developers of GAMESS (Ames Lab, Iowa State University, and ORNL) [85, 86] to enable each chemistry code to use novel features of the others to perform calculations that no single package can do [81, 84, 86, 87]. In addition, the PNNL team achieved an order of magnitude improvement in performance for Hessian evaluation by using the CCA to construct a new hybrid algorithm using multiple levels of dynamic parallelism [88]. NWChem leader Theresa Windus states [89], "The hybrid algorithm would have been impractical to implement in a reasonable time without using the CCA; we see CCA-based technology as the foundation of future software development in chemistry."

These two examples are illustrative of the experience of many CCA users, who find that it allows them to explore approaches to software and scientific research that would have otherwise been extremely difficult to implement, and as a result realizing greater scientific productivity.

Nucleating a Broader Community for High-Performance Scientific Component Software. Although the CCTTSS project initially represented the entire CCA community, over its five year lifetime the CCA community has grown much larger and broader. Typically, around one quarter of participants in the quarterly CCA Forum meetings are not associated with CCTTSS, and roughly the same percentage of Forum meetings have been hosted by institutions outside of CCTTSS. The Forum's main mailing list currently includes 192 members. An increasing number of non-DOE projects are using and contributing to CCA development. We are aware of numerous CCA-related projects funded by the Dept. of Defense (DoD), European Union, National Aeronautics and Space Administration (NASA), National Institutes of Health (NIH), National Science Foundation (NSF), Northrup Grumman, and others.

Broader still is the influence of CCA ideas on the HPC community. Even those who are not using our implementation of components are increasingly adopting the ideas and design patterns of CBSE in their projects, due in large part to our outreach (evangelism) activities. Nowadays anyone who seeks to build an HPC component system first looks to the CCA. Other community-building activities of the CCTTSS have included more than 50 peer-reviewed scientific papers (including a Best Paper Award at the 2005 IPDPS conference [90]) and many more talks; organization of, and participation in scientific workshops and meetings; numerous tutorials; and education of the next generation of computer and computational scientists, through completion of Masters and PhD theses, incorporation of CCA into courses, and student internships at National Laboratories. This community development is an important and valuable complement to DOE's direct investment in the CCA.

1.3 Overview of Technical Approach

To address the needs of the next generation of computational science applications, we organize our work around four central thrust areas. **Component Technology Initiatives** (Sec. 1.5) focus on developing new and enhanced capabilities for applications. **The CCA Environment** (Sec. 1.6) itself must be supported for existing and new users, as well as extended to support the needs of the Initiatives. We will also strongly focus

on making the CCA environment easier to use. **The CCA Toolkit** (Sec. 1.7) will develop a "component ecosystem" by making more software available in component form, and by developing common interfaces to unify access to existing software. Finally, our effort in **User and Application Outreach and Support** (Sec. 1.8) focuses on assisting applications groups through direct interactions, documentation, and tutorial and example materials. The work we propose is strongly motivated by our interactions with numerous applications, and is tightly integrated, both across the focus areas and across the participating institutions. Each thrust's activities are described in more detail below.

1.4 Motivating Applications and Collaborations

With CCA technology already being used in over a dozen different fields of science, the CCTTSS has developed extensive interactions with a wide range of applications as well as with other enabling technology centers. TASCS will build upon and extend these interactions in both breadth and depth. The enhancements and initiatives that we propose are motivated by the needs of these application groups. From among the many applications that will ultimately benefit from these developments, we have selected a small number as exemplars, to provide specific guidance to the development activities and serve as testbeds.

The Center for Simulation of Wave Interactions with Magnetohydrodynamics (SWIM) project [91] was recently funded as an Fusion Simulation Project (FSP) prototype under the SciDAC program. The primary focus of this center is the coupling of different aspects of plasma physics to produce integrated simulations. As with many coupling efforts, this project starts with substantial bodies of existing code that cannot be completely rewritten to accommodate the integration. This situation motivates developing a Parallel Coupling Infrastructure (PCI) (Sec. 1.5.1) to support incrementally moving from disparate applications to progressively more tightly coupled, higher-performance integrated simulations. Because simulation components have to continue to function properly in standalone use even as they are adapted to support the integrated physics, extensive and detailed testing of components in both contexts will be required. This is a key motivator for the enhanced verification mechanisms of our Software Quality and Verification initiative in Sec. 1.5.3 and the "test harness" work of Sec. 1.6.3.

The DOE computational chemistry community currently anticipates an FY2007 call for large-scale community software centers focusing on high-end capabilities. Based on work by the CCTTSS, the major players in this community are already convinced that the CCA's component approach is central to the success of any community software integration and development effort [92]. However, to reach the petascale, chemistry applications need much better tools to express dynamic multiple-program multiple-data (MPMD) and Multi-Level Parallelism (MLP). We will collaborate with the developers of the NWChem parallel computational chemistry package [80–82, 93] to develop these capabilities as part of our Initiative on Emerging HPC Hardware and Software Paradigms (Sec. 1.5.2). Large, long-running chemistry calculations will also be the focus of work on providing fault tolerance capabilities in the CCA environment. In the computational biology community, heterogeneous computing environments are beginning to play an important role. We will work with on-going DOE effort in this area [94], to develop CCA capabilities for Field-Programmable Gate Array (FPGA) systems.

The CFRFS project [95], in collaboration with the CCTTSS, has developed an extensive toolkit for combustion simulation. The diversity of simulations enabled by the toolkit can make it challenging for users to confirm that various interchangeable components are indeed used properly. Another challenge for users of the combustion toolkit is making sound choices from among the available implementations and parameters, with suitable trade-offs among performance, accuracy, mathematical consistency, and reliability [96]. Both challenges motivate our work in interface semantics and verification described in Sec. 1.5.3. In addition, building on preliminary work by the CFRFS, CCTTSS, and Allen Malony's group at the University of Oregon, we plan to exploit the dynamic nature of components to provide "computational quality of service" (CQoS) capabilities, thereby allowing applications to adapt during execution to the specifics of a particular problem instance and its execution environment (Sec. 1.5.3 and 1.5.4); this CQoS work is further motivated

by simulations in quantum chemistry [92], high-energy accelerator physics [97], and fusion [98].

Collaborations with other Centers for Enabling Technology also figure prominently. We plan to continue our work with the Terascale Optimal PDE Solvers (TOPS) [53] and Terascale Software Tools and Technologies (TSTT)/Interoperable Technologies for Advanced Petascale Simulations (ITAPS) [55] centers on the development and implementation of common interfaces for solvers and meshing tools; this solvers work is partially motivated by new SciDAC projects, including core-edge transport simulations in fusion [98], beam dynamics simulations in high-energy accelerators [97], and electromagnetic modeling in nuclear physics [99]. We will also collaborate with the Scientific Data Management (SDM) center [100] to establish interoperability between the CCA environment and scientific workflow tools, such as Kepler [101]. As part of the CCA Toolkit thrust area (Sec. 1.7), all of this work will contribute to the component ecosystem. The SDM collaboration specifically targets needs of the Center for Plasma Edge Simulations (CPES) fusion integration project [102], which plans to begin with a loose coupling approach based on Kepler, but will eventually need higher-performance coupling, such as our proposed PCI initiative will provide.

The needs of applications directly, and the Component Technology Initiatives they motivate, will require extensions and enhancements to the CCA Environment (Sec. 1.6).

1.5 Component Technology Initiatives [Coordinator: L.C. McInnes, ANL]

Our proposed technology initiatives are based on the premise that in addition to aiding software development, the component environment can facilitate the development and deployment of new computational capabilities to benefit the entire life cycle of simulation software. Several initiatives support computational science at the ultrascale by exploiting the CCA's capabilities for *adaptivity within the software life cycle* to introduce new forms of parallelism and parallel coupling mechanisms for multiphysics applications, as well as to support heterogeneous/hybrid computing architectures and fault tolerance. Additional work exploits the component paradigm's support for *dynamic adaptivity during runtime* in response to performance, algorithmic, numerical quality, and other concerns.

1.5.1 Coupling Separately Developed Codes for Combined HPC Simulations [Coordinator: R. Bramley, IU]

Motivation. This initiative addresses a growing problem in scientific computing: merging multiple codes developed in isolation into a single high-performance multiphysics simulation [103–107]. Applications such as fusion energy simulation, climate modeling, space weather, and biological cell modeling are now combining simulations of disparate subphysics models into larger, integrated simulations. The set of software systems and tools that help tie together such multiphysics simulations is called a "Parallel Coupling Infrastructure (PCI)". Current coupling efforts have common features:

1. The motivation is achieving higher-fidelity models by coupling separately developed models.
2. The submodels have an existing base of codes not originally designed to interoperate with other codes.
3. The investment in the submodel software is too extensive to duplicate, and the codes are still under active use and development. Coupling of submodels must be done in an continuous and incremental manner concurrently, even while the individual codes advance scientifically.
4. Some or all of the constituent codes have stringent high-performance computing requirements, including scalable parallelism.

Presently, this is often done in an ad-hoc way, addressing only 2–3 codes in a specific area, and it often necessitates domain investigators redeveloping methods and infrastructure. Here we propose a more general evolutionary approach, providing a process and an infrastructure enabled by tools developed under the TASCS. A PCI is needed for SWIM [108] and CFRFS [109], would leverage work in TSTT/ITAPS (mediating different discretizations) [110], TOPS (solvers spanning the newly created multi-code simulations) [111], and SDM (workflow and data management) [112], and is a major focus of the Institute for Coupling High-Performance Simulations (ICS) [113] activities.

Approach. Our goal is a PCI reusable across many different multidisciplinary simulations. Experience from well developed areas [114–116] identifies an evolutionary process towards a unified simulations. TASCS will create and deploy the systems tools needed in a high-performance setting for each evolutionary stage.

A first (optional) stage uses files to transfer data among separate executables. HPC requirements imply parallel I/O, which can use write-combining and other intermediate optimizations [117, 118]. The second stage replaces the programs as black boxes with full component capabilities, using high-speed interconnects to transfer data [116, 119–123]. This stage need not require application code changes [124], and computational quality of service (CQoS) (Sec. 1.5.4 could optimize the ensemble calculation. The third stage uses a full-scale coupler component for asynchronous data sharing and concurrency among instances of the sub-physics components. The Community Climate System Model (CCSM) [125] is a successful example of this evolutionary process in a non-CCA context, where codes for ocean, sea-ice, atmosphere, etc. are integrated into a single simulation.

As part of the proposed work, we will develop a PCI technology to aid each stage of this evolution. Initially all components will be launched on a single system, such as the National Leadership Computing Facility (NLCF) [126]. When components do not require this level of capability and can be run more efficiently on nearby smaller scale clusters, the PCI must supply RMI-like capabilities. We will develop technologies that are general to a wide variety of application domains but serve to couple codes that begin as separately created sub-physics models and then are gradually integrated into a complete high-performance multiphysics simulation. The process uses a workflow structure that couples codes with adapters converting the output from one model to the input of another. This toolset will use parallel I/O methodology and draw on work proposed by the SDM. At the next level of integration files will be replaced with shared memory constructs that allow more efficient and asynchronous behavior.

We will leverage Kepler-based workflow tools [101, 127] from SDM to orchestrate the execution of and data translation between codes by adapting input and output files. These tools also can make explicit the innate concurrency among the components that comprise the workflow and make an excellent starting point for evolving workflow ensembles into HPC combined simulations. We will develop tools that support higher performance asynchronous communication, allowing the move away from file-based interaction to finer grained inter-component communication and greater parallelism. Adaptation of Argonne's Model Coupling Toolkit (MCT) [120, 128] provides a generalized tool for coupling these now asynchronous components, while preserving necessary synchronizations. Other parallel data redistribution capabilities will also be applied for efficient transfer and scheduling of parallel data exchanges [90, 129]. Collaboration with TSTT is already providing help with the difficult issues of converting data between different discretizations.

Following the staged approach outlined above, a natural connection will be made to fusion energy simulations the first year to assist in code coupling. As MCT is generalized, PCI efforts will extend to other applications. In each case PCI enables new types of simulations of higher fidelity than previously possible. The TSTT collaboration and evolution of MCT will help domain scientists with a generalized coupling infrastructure providing component orchestration, data interpolation, and data translation.

Impact. Our evolutionary approach to PCI will provide applications with a clear path from the relatively simple workflow capabilities being developed by the SDM center to fully-integrated concurrently-running high-performance simulations, while recognizing the realities that scientists can't afford to rewrite their codes from scratch, and the they must continue to evolve scientifically as standalone applications too. Though our development activities will focus on the SWIM and CFRFS applications, we will work through the ICS and other outreach activities to insure that it satisfies the needs of the broader community. The tools and technology we develop will be rolled out to other application projects as soon as it is ready.

1.5.2 Emerging HPC Hardware and Software Paradigms [Coordinator: J. Nieplocha, PNNL]

Motivation. The development of petascale computational science capabilities continues to face major challenges in adapting or developing software to effectively use emerging hardware environments. Coming petascale computer systems will be characterized by large processor counts (systems with $\mathcal{O}(10^4 - 10^5)$ processors are already being deployed [3]), and increasing use of heterogeneous and specialized environments, in which FPGAs, graphics processors, and other hardware are harnessed to accelerate general scientific computing. High processor counts will require applications to expose more parallelism. For situations where it doesn't make scientific sense to scale the problem size or resolution, using MPMD or MLP approaches, in which different groups of processes carry out different parallel tasks simultaneously can be used to increase parallelism, but are not easy to manage or code using current tools. Heterogeneous computing environments will require adaptation of programs to work in the presence (or absence) of various specialty interfaces, which are currently unique to each vendor. And finally, the sheer number of parts in petascale systems will pose significant issues with respect to hardware (and software) faults, some of which may be most effectively handled by fault-awareness at the application level.

The CCA design already naturally provides indirect support for these emerging computer platforms. Components are a natural means to express and encapsulate the new algorithms that will be required. They facilitate reuse of software across multiple applications and different disciplines, and make it easy to substitute alternate implementations (algorithms) to tune applications for particular hardware platforms. The purpose of this Initiative is to provide applications with new capabilities which *directly* support the transition to petascale computing.

Our work in this area is directly motivated by collaborations with on-going DOE projects in quantum chemistry [93] and computational biology [94], and will also involve collaborations with existing [130] and proposed [131, 132] research projects focusing on fault tolerance.

Approach. We will provide CCA users with more flexible and dynamic means to express application parallelism through the development of support for the management of process groups and multiple-component multiple-data (MCMD) applications. This work will develop standard CCA tools and interfaces, guided in particular by our prior experience with such applications in chemistry [88] and the threaded parallel "task graph" execution model provided by the Uintah Computational Framework [51, 133]. Towards this goal, we will collaborate with computational chemists in PNNL's Molecular Science Software Suite Group [93], and with whom we have pioneered the use of component technology to address MLP challenges. This application will also provide an opportunity to leverage CQoS work (Sec. 1.5.4) to determine the optimal size of the processor groups executing MCMD components.

The process group abstraction will be based on, and compatible with, those in MPI-2 [26], PVM [27], and Global Arrays [28, 29]. The MCMD management utilities can be thought of as a process-group-aware extension to the CCA specification's `BuilderService` interface to simplify dynamic modification of MCMD component applications. The new capabilities will be cast as optional extensions to the CCA specification and implemented as service components. These new capabilities will initially be demonstrated as part of this Initiative for geometry optimization for biological systems with NWChem, and subsequently rolled out to a broader user base.

Our work on heterogeneous computing environments will focus on developing an appropriate set of abstractions for the CCA environment to allow the platform-specific details of hardware accelerators to be encapsulated in specific components that use them, and transparent to the rest of the application (including replacing an accelerated component with a traditional implementation on platforms where the accelerator is not available). Though currently the hardware-level interfaces vary widely by vendor, we believe that an asynchronous computing model [134] will provide a suitably general approach, and can be implemented on top of the event model which will be added to the CCA specification (Sec. 1.6.2). We will demonstrate the use of this technology with proteomics application Polygraph [135, 136], which performs peptide sequence

matching against a database of experimental mass spectra. Several sub-steps of the matching have already been targeted for acceleration, and a variety of FPGA systems are available at PNNL and ORNL to demonstrate portability (see Sec. **??**, **??**). Additionally, we will collaborate with an ongoing PNNL computational biology project [94], where FPGA systems are being used, to develop CCA components and interfaces to simplify their use of heterogeneous computing environments.

With respect to fault tolerance, our approach is to insure that CCA-based applications can take full advantage of the capabilities being developed by other projects focusing on this research area. This involves first establishing a hierarchy for fault response in CCA applications. Basically, the CCA framework must "do no harm" when a fault occurs – conveying the fault information fault-aware application components, and in their absence, responding as sensibly and gracefully as possible on the application's behalf. Second, the CCA environment must interact appropriately with application-level fault tolerance tools. For example, it must be able to use a checkpoint/restart system to save its own state, and provide the appropriate services to application components so that they can do likewise. Our primary work in this area will focus on checkpoint/restart services, in collaboration with ongoing Forum to Address Scalable Technology for Operating Systems and Runtimes (FASTOS) Scalable Fault Tolerance project [130], and we plan to demonstrate this capability in coupled cluster algorithm of NWChem. Two SciDAC Centers for Enabling Technology [131, 132] are proposing deeper investigations into fault tolerance, and we will collaborate with them on integrating richer fault notification and response capabilities into the CCA environment through the development of standard interfaces to proposed system-level services or fault tolerance backplanes. We will also work with the CQoS initiative (Sec. 1.5.4) to determine how faults should be incorporated into their analysis and tools.

Impact. This Initiative will provide CCA users with new tools that will simplify and accelerate their development of true petascale applications on diverse hardware platforms. They will be able to flexibly and dynamically express higher levels of parallelism, transparently take advantage of specialize co-processing resources, and support intelligent application-level responses to the hardware failures that are inevitable on systems of this scale.

1.5.3 Software Quality and Verification [Coordinator: T.L. Dahlgren, LLNL]

Motivation. Scientific software — especially from third-party sources — can be very complex, making its correct use a major concern. Much of the scientific computing community relies on the most basic specifications; namely, method signatures and documentation. Unfortunately, such documentation is often incomplete or out of date, and cannot be automatically verified in traditional software environments. Extending interface definitions with semantic annotations, which can be automatically verified or enforced at composition or run time, will provide the developers of scientific software with a powerful tool to help them catch errors earlier and insure the correct use of the software.

This work is motivated by external collaborations and other activities within TASCS. Our on-going collaborations with the TOPS [53, 111] and TSTT/ITAPS [55, 56, 110, 137] SciDAC centers on the design of broadly applicable interfaces (Sec. 1.7) provide two examples of the types of software expected to benefit most from interface semantics and enforcement. The CQoS Initiative (Sec. 1.5.4) has similar needs with respect to the semantics of dynamic adaptation. Finally, more rigorous and verifiable interface contracts are an important and valuable complement to our planned development of a standardized test harness for semi-automatic unit testing of CCA components (Sec. 1.6.3).

Approach. This initiative focuses on interface semantics and verification for HPC scientific applications. The expressiveness and suitability of general-purpose specification mechanisms will be investigated for automated verification and behavioral adaptation tools applied across disciplines within computational science.

Component interfaces, expressed separately from the implementation in a language such as SIDL, can be extended with semantic information to provide concise, human-readable and machine-processable spec-

ifications. Examples include numerical properties of parameters and return values [46, 47], constraints on invocation sequences [138–143], restrictions on array distributions in parallel interfaces, and even specification of units for arguments representing physical quantities. The corresponding assertions can be verified at composition- or execution-time under the control of the component framework. Unlike traditional verification techniques based either on post-execution comparisons with prior or analytical results or algorithm-based fault tolerance techniques, this approach enables error detection closer to the point of failure. The result is improved testing, debugging, and runtime monitoring of software quality.

Through our collaborations, we will identify and define relevant component characteristics and constraints; pursue suitable interface semantics representations; and address the use of such representations in automated composition and runtime verification. We will build on existing work [46, 47, 144] — based on extensions to SIDL in the form of Design by Contract annotations [145] — to integrate static semantics. We will use the interface semantics to guide the development of component unit tests, as part of the component test harness outlined in Sec. 1.6.3. Framework-based alternatives for the expression and use of dynamic semantics for both runtime verification and adaptation will also be investigated. Once semantic interface specification and verification technologies are demonstrated within the context of our motivating applications, we will use the CCA Toolkit components as the first route to bring these new capabilities to a broader audience and roll the capabilities out to other applications.

Impact. This work, together with the development and deployment of the test harnesses of Sec. 1.6, will provide component and application developers with powerful tools to improve the quality of their software. It will help uncover bugs in component implementations and catch errors sooner. Furthermore, the interface semantics of this initiative will enhance the dynamic adaptation capabilities of the CQoS infrastructure of Sec. 1.5.4 by providing additional information for better informed decisions.

1.5.4 Computational Quality of Service and Adaptivity [Coordinator: L.C. McInnes, ANL]

Motivation. As computational science progresses towards ever more realistic multiphysics and multiscale applications, the complexity is becoming such that no single research group can effectively select or tune all of the components in a given application, and no single tool, solver, or solution strategy can seamlessly span the entire spectrum *efficiently*. Common component interfaces enable easy access to suites of independently developed algorithms and implementations. The challenge then becomes how to make sound choices dynamically during runtime among the available implementations and parameters, suitably compromising among performance, accuracy, mathematical consistency, and reliability.

We will address this challenge by developing tools for *computational quality of service* (CQoS) [146], or the automatic selection and configuration of components to suit a particular computational purpose. As further explained in [147], CQoS embodies the familiar concept of quality of service (QoS) in networking as well as the ability to specify and manage characteristics of the application in a way that adapts to the changing (computational) environment. Specific scientific applications that motivate this research are:

- *Parallel mesh partitioning in combustion simulations:* The SciDAC-funded CFRFS [76, 109] project is developing a CCA toolkit for flame simulations using block-structured adaptive meshes, which must be partitioned to enable parallel computing. Because no single partitioner is optimal [148], this CQoS project aims to choose an efficient partitioner and an appropriate configuration for a given mesh [96].
- *Resource management in quantum chemistry:* In collaboration with the proposed SAP project led by M. Gordon [92], we will use CQoS infrastructure to enable dynamic adaptation to available resources (e.g., memory and time) during molecular wavefunction determination and other quantum chemical subproblems [63]. We will also collaborate with the Emerging HPC Paradigms initiative (Sec. 1.5.2) on adaptively using MCMD and hybrid computing paradigms in quantum chemistry simulations [88].
- *Efficient solution of linear systems arising in accelerator and fusion simulations:* The solution of linear algebraic systems of equations often dominates the overall runtime of large-scale PDE-based simula-

tions, including some phases of modeling in the proposed SciDAC application projects *Accelerator Community Code Development and Discovery* [97] and *Framework Application for Core-Edge Transport Simulations* [98]. CQoS work here focuses on the automation of appropriate choices for algorithms and parameters of TOPS [53, 111] linear and nonlinear solver components [54].

Approach. While we expect that the logic involved in characterizing these problems and choosing appropriate solution strategies will be vastly different, we believe that the *software infrastructure* to analyze and characterize each problem and its potential solutions as well as the *software infrastructure* to implement a decision (once made by domain-specific logic) is similar and may be generalized [8, 149–151].

Fig. 1.2 illustrates our vision of how CQoS infrastructure will help to analyze, select, and parameterize components for these motivating applications. This diagram shows the two main facets of our CQoS tools: (1) *measurement and analysis infrastructure*, which combines performance information and models from historical and runtime databases along with interactive analysis, including statistical analysis and machine learning technology; and (2) *control infrastructure*, which encompasses decision-making components that evaluate progress based on domain-specific heuristics and metrics, along with services for dynamic component replacement. These two groups of CQoS tools, which may be employed both for initially composing and configuring an application, as well as for runtime control, are largely decoupled and interact primarily through a substitution assertion database. Preliminary research that has led to this approach is discussed in [8, 10, 146, 150, 152].

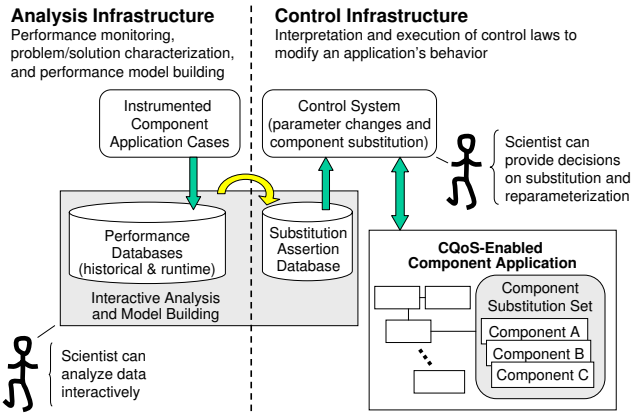

Figure 1.2: Overview of CQoS infrastructure.

This initiative exploits work on interface semantics in the Software Quality initiative (Sec. 1.5.3) and relies on infrastructure under development by A. Malony and S. Shende of the University of Oregon for performance monitoring, modeling, and analysis [153, 154]. We will also collaborate with the Performance Engineering Research Center (PERC) [155, 156] for performance tools and the TOPS [53, 111] project for solver components, and we will leverage relate work (e.g., [157–159]) where appropriate. A required CCA enhancement is development of an event model that supports dynamic CQoS behavior. A key facet of this work is development of a *CQoS Testbed*, including components drawn from the motivating scenarios discussed above, to test and validate CQoS infrastructure.

Related work on adaptive software for scientific computing includes [158–174],while work on semantic information and performance monitoring includes [138, 175–185]. Further details about motivation, related work, and proposed CQoS research, which cannot be included here due to space constraints, are discussed in [147].

Impact. This work will enable computational scientists to compose, substitute, and reconfigure software so that trade-offs can be made dynamically at runtime among performance, precision, underlying models, and reliability when choosing among available component implementations and parameters. Because we are developing the base CQoS infrastructure to be generally useful to high-performance scientific applications, we expect these tools to prove useful in application areas beyond those introduced above when the need arises for dynamically selecting and parameterizing algorithms/software during runtime.

1.6 The CCA Environment [Coordinator: G. Kumfert, LLNL]

The CCA environment is the foundation that all other CCA activities presume. It includes SIDL, the language interoperability tools, the CCA specification, the frameworks that implement the specification, and

a host of other tools and bits that help the CCA developers and their customers work effectively. We have subdivided this work into three main sections: **core support** (Sec. 1.6.1) to maintain our current capabilities, **enhancements** (Sec. 1.6.2) needed to support the proposed research initiatives, and **usability** (Sec. 1.6.3) improvements for our general customer base.

1.6.1 Core Tool Support and Maintenance [Coordinator: B. Allan, SNL]

Nothing in scientific computing stands still. As long as there are new compilers, new platforms, new tools, or even new versions of any of the above, there will need to be matching effort expended on core support. Moreover, new technical developments in the preceding Initiatives (Sec. 1.5) as well as the other activities in this section will place new demands on our core tools. This effort ensures current capabilities in the face of change — internal and external. Specific deliverables include maintaining current platforms (Linux, IBM AIX, and MacOS X) and porting to additional DOE high-end platforms; on-going "help desk" support for the core tool; updating and improving reference and developer documentation; and developing conformance tests for the CCA specification and comprehensive integration tests for the CCA toolkit.

1.6.2 Enhancements [Coordinator: T. Epperly, LLNL]

The enhancements we propose to the CCA environment are activities of general utility, responding to needs expressed by various applications groups, and the four Component Technology Initiatives. Enhancement activities are individually smaller in scope and carry less technical risk than the Initiatives, and target more fundamental capabilities of interest to a broader spectrum of CCA users.

CCA Specification. The CCA specification is intended to be extensible, making a distinction between user's components and a framework's services. We plan to augment the current specification with a suite of oft-requested services. These include a general `EventService` that would support both a publish/subscribe model with rich data transfer as well as a lightweight asynchronous interrupting event model for special cases. The currently prototypical `MPIService` and `CommandLineService` will be hardened and standardized. We intend to design a `GuiBuilderService` as an extension of our existing `BuilderService` so that the GUI can exist as a component. We will also design and implement mechanisms for supporting component sub-assemblies as new components, and define general mechanisms for saving and exchanging those sub-assemblies.

Bridging CCA to Other Frameworks. Although the CCA is the foremost component approach for HPC there are a broad range of other software environments important to scientific computing that have component-like characteristics. These include scientific workflow systems, such as the SDM center is developing [100]; dataflow environments, such as the Visualization Tool Kit (VTK) [186]; and "domain-specific" computational frameworks, such Cactus [187] and Earth System Modeling Framework (ESMF) [188]. Foreign frameworks can participate in the CCA universe by a variety of means, including 1) *directly* by porting modules as CCA compliant components, or 2) *indirectly* as CCA-like metacomponents when the behavior of the foreign system is "close enough" to the CCA model that it can masquerade as CCA or 3) *adapted* through a common distributed protocol such as Web Services.

Our first goal will be to create mechanisms for interoperability with workflow environments, such as Kepler [189], as motivated by our collaboration with the SDM project [100]. Kepler is being used by the CPES fusion project [102], and we envision the transition from workflow-based loose coupling to a more direct component-based coupling as a key early step in the development of many other coupled simulations (c.f. Sec. 1.5.1). Next, ESMF is currently being integrated into the CCSM. Because scientists both in the DOE climate community and at the National Center for Atmospheric Research (NCAR) wish to make use of CCA functionality, a collaboration is planned between NCAR and TASCS as part of the proposed work to make the more generic component functionality of CCA accessible from ESMF applications. Additional work will depend on application drivers, though interoperability with the VTK visualization environment

is one likely target, in collaboration with the Visualization and Analytics Center for Enabling Technology (VACET) [190].

SIDL/Babel. Incremental improvements will be in expanding the number of languages supported as well as the palette of available types expressible in SIDL.

New languages include support for Matlab [191] and Fortran2003 [192]. For Fortran2003, this proposal complements an existing Small Business Innovation Research (SBIR) project at Tech-X to develop the core bindings. Under TASCS, the Babel team will handle the configuration, coordinate on testing, and provide on-going customer support after the bindings are delivered.

The new types to be added to SIDL (and by extension every language binding that Babel supports) present a bigger challenge. We propose to add support for C-like structs [193] – heterogeneous collections of data passed between languages with minimal data copy for maximal performance (none between C, C++, and Fortran2003). We also propose to add a distributed array type to SIDL, which will be sufficiently general to allow most distributed array libraries to be used to provide the back-end functionality. Finally, we will implement a user-extensible "typemap" facility whereby users can define special-case types that have bindings in, say, two or three languages but are opaque types elsewhere.

1.6.3 Usability [Coordinator: C. Rasmussen, LANL]

This work is proposed in response to our experience and observations from conducting numerous tutorials and coding camps, and working individually with users over the last five years. While we have been very successful with our early adopters, we also learned a great deal where our technology could be made easier for the general user.

CCA-Lite. This activity focuses on creating a tiered CCA specification, giving users the ability to eschew full language interoperability afforded by the CCA specification in exchange for a more portable and less complicated development system. CCA-Lite will specify C components and static linking, only, and will be accessible from any language that can call and be called upon by C. Fortran95 will be made accessible in the near term by the Chasm Fortran language tools project [194, 195], the need for which is expected to be obviated over the next 2 years by the inclusion of the Fortran2003 standard's `BIND(C)` interoperability feature by most compilers. Lite requirements are: 1. Language interoperability between components and the CCA framework in three major languages for scientific programming: Fortran, C, and C++; 2. The composition of components into an application (program `main`) using the CCA `BuilderService` specification; 3. Static linkage of the final application; 4. An automated migration path to full CCA components.

There are trade-offs between these tiers that can be seen by tracing the interactions between four distinct entities: CCA Components, CCA Frameworks, CCA-Lite Components, and CCA-Lite Frameworks. The user's motivation for a pure Lite situation is simplicity: a few mainstream languages, static compilation, minimal glue code, easy to configure, build, debug, and greater portability to bleeding-edge leadership class machines. Lite components can work natively on a Lite-aware framework, but the interoperability guarantee between different Lite components may be limited, and will be carefully defined. Connecting a Lite component to a regular CCA component would be possible only with a full-fledged CCA framework containing bridging capabilities between the two regimes. Ccaffeine, for example, already does this between CCA components and the C++-specific "Classic" interface that predates SIDL/Babel. Similar bridging to Lite will be added. Code generation tools and/or libraries will be created to automatically promote a Lite component to be fully CCA compliant, providing the dynamic linking and language interoperability that all full CCA components share. CCA-Lite is a part of our Usability portfolio because it offers this tiered alternative that makes CCA more approachable to a wider audience.

Component Debugging and Testing. CCA is very effective and managing and hiding the complexity of multi-language programming. Paradoxically, testing and debugging are two commonly recurring stages of the software life cycle where those hidden details need to be exposed and managed differently. An

application restricted to C, C++, and Fortran can be managed with a standard debugger such as TotalView [196], gdb [197] or dbx [198]. Adding a component implemented in Python or Java in the mix mandates the use of a *combination* of debuggers, all attached to the same process.

We propose to develop and document tools and techniques to make multi-language debugging practical for application and component developers. This proposal focuses on three main approaches to improve component debugging: extensive runtime validation (Sec. 1.5.3), automatic tracing of component method calls [199, 200], and tools to manage the right combination of third-party debugging tools.

The compositional nature of component-based applications increases the importance, and potential, for proper testing of individual components, without necessarily assembling them into full blown applications. Testing of scientific codes remains a sporadic effort, due in part to the lack of test harnesses and tools specifically tailored to the needs of scientific programmers.

We propose to develop a component test harness to facilitate the development of unit tests for CCA components, modeled after tools such as JUnit [201] and PyUnit [202]. Together with our work on interface semantics (Sec. 1.5.3), this will provide a comprehensive infrastructure to support user software testing and verification.

1.7 The CCA Toolkit [Coordinator: R. Armstrong, SNL]

The purpose of this focus area is to develop and enrich the "component ecosystem" so that users of the CCA will be able to obtain an increasing number of the components they need "off the shelf". This work naturally divides into four inter-related activities.

Development Tools and the CCA Base Installation. As is typical for HPC software, CCA components are generally distributed as source code and must be properly compiled integrated into the user's work environment to be effective. The "CCA base installation" is the evolving set of tools, file naming conventions, and other scaffolding that encapsulates our experience and recommendations for a uniform approach to the development and delivery of CCA components intended to make the integration as easy as possible. Based on experience gained with applications and the CCA tutorial, we now have a basic architecture which need to be implemented and refined.

We will also enhance the base installation scaffolding with both commandline and IDE-based tools to assist with the more mechanical aspects of component development, along similar conceptual lines to "Ruby on Rails" [203]. We will develop Eclipse [50] plug-ins to act as a SIDL and CCA component "wizard", in close collaboration with the current and proposed work at LANL on the Eclipse Parallel Tools Project (PTP) [204, 205].

The Toolkit and CCA tutorial (Sec. 1.8) will serve as the primary testing grounds for the CCA base installation. The tools will be well-documented and made available to the general CCA community as well.

CCA Component Collection. The CCTTSS has begun developing a suite of components of general utility across a broad range of scientific applications. Though it is not yet formally distributed, the Toolkit currently contains 16 components [57] which have been contributed by their authors, and are primarily derived from their CCA applications. We propose to complement continued acceptance of voluntary contributions with a focused effort to add a coherent core of functionality to the Toolkit.

We will create a core toolkit comprised of a comprehensive, integrated set of utility, data, numerical, and scientific components, many based on widely-used libraries, which will allow users to assemble complete, if simple, scientific simulations almost entirely from the toolkit. This set of tools will facilitate rapid application development for scientific or numerical proofs of concept and experimentation, and can also provide valuable plug-and-play capabilities to existing HPC applications. The toolkit will create a minimal general purpose set of components to form generic HPC simulations "out of the box" including but not limited to:

- **Linear and Nonlinear Solvers.** These components will be developed in collaboration with members of the TOPS center [111] and will include linear and nonlinear algebraic solver components as well as

access to lower-level vector-matrix operations.

- **Structured and Unstructured Data Models.** These components will automatically decompose data across processors using the interfaces and implementations of the ITAPS [110] and Algorithmic and Software Framework for Applied Partial Differential Equations (APDEC) [206] centers.
- **General Purpose Utility Components.** These components are generally useful for coping with the parallel component world. These include an MCMD component (Sec. 1.5.2) for managing forms of parallelism richer than single-program multiple-data (SPMD), an event service component that provides system events as well as a generic event structure for communication between components, a command-line processing component for passing arguments to components without framework intervention, and many others. Also of general interest is a versatile parallel I/O component based on existing packages. These utilities will be developed by the TASCS as part of this proposal.

Although some utility components and interfaces will be constructed by TASCS, most of the CCA-specific value-added will come from automating the build and distribution of these components consistent with the CCA base installation. The underlying functionality will be drawn from other SciDAC centers and projects throughout the HPC community. Extensive documentation and examples will be included in the download, following the format of the CCA Tutorial Hands-On Guide [207].

Often developers will need to insert their own code into custom components to orchestrate the toolkit components. The toolkit will make heavy use of script-generated templates for general-purpose components such as drivers that act as the `main` program in a componentized application and other conveniences that developers commonly need.

The toolkit will also be the first testbed for the proposed technologies for the expression and enforcement of interface semantics under the Software Quality initiative (Sec. 1.5.3), thereby providing good examples of that utility for users not familiar to such ideas.

Community Interface Development. An important complement to the development of actual *components* for the Toolkit is the definition of common interfaces which can provide interoperability across multiple libraries in a community offering similar capabilities. The CCTTSS has been very successful in working with several communities on community interface specifications [54, 56, 63] and we plan to continue and expand this work in TASCS, and add component *implementations* to the Toolkit wherever possible. Specific collaborations are planned with fusion and accelerator physics projects [97, 98, 108]

CCA Component Repository. Accessibility is a key criteria for the uptake of Toolkit components. Therefore, we will establish a repository, providing a single reference location for users to look for component software. We plan to use a simple approach, insuring that components can be located based on their description and metadata, conveniently downloaded, built, and easily integrated with other toolkit components. We will develop a facility that identifies candidate components that satisfy CCA port dependencies, similar to, for example, the Comprehensive Perl Archive Network (CPAN) [208] or Maven [209].

1.8 User and Application Outreach and Support [Coordinator: D.E. Bernholdt, ORNL]

CCA's long-range goal is to fundamentally change how high-performance scientific software is developed and used. Consequently, we place a strong emphasis on outreach and support activities to broaden the awareness and adoption of CCA technology.

Application Support. These activities are an important *bidirectional* conduit for exchanging information and experience. Direct interactions between CCA users and developers continues to prove highly effective in propagating the use of CCA tools and design patterns in the wider HPC community. On the other hand, this close application connection enables the CCA team to gather a broader range of experience with component applications, allowing us to better serve both existing users and new adopters, and help focus our attentions

on areas most important and valuable to application groups. This proposal directly reflects our experience with application groups in CCTTSS.

TASCS envisions three different mechanisms to support its work with applications. First, we have built support into our Component Technology Initiatives for the necessary level of interaction for the targeted applications to insure that TASCS goals are achieved. Second, we have established collaborative ties to numerous projects, with which we plan to collaborate via Scientific Application Pilots (SAPs) proposals (Sec. **??**) and through "embedding" of CCA-related activities in the application proposal. Third, we have allocated a modest level of effort ($\sim$1 FTE per year) to provide support for other projects. Prioritization among opportunities will be based on the level of support required, the likelihood of contributing towards significant advances on the scientific side, and the prospects of obtaining new experience and knowledge of value to the wider CCA community.

User Outreach and Support. These activities are designed to assist and educate individual CCA users, regardless of their application connections and affiliations. Foremost among those activities is the CCA tutorial [207,210], which has been offered 20 times since 2001, including the last four Supercomputing conferences (2002–2005). The tutorial educates users about components and the CCA, and helps capture and disseminate examples of "best practices.". We plan a major revision to the tutorial to provide more sophisticated example applications, based primarily on CCA Toolkit components (Sec. 1.7), and continual updates as we better understand the "best practices" for use of component technology in scientific computing. The tutorial is also the initial testing ground for continuing development of the CCA base install template and related tools (Sec. 1.6.3).

Coding camps, another user support activity, are working meetings in which a group of users and experienced CCA tool developers spend as much as a week together in the same location. They give CCA users instant access to knowledgeable developers to assist in overcoming initial hurdles that might otherwise require a protracted exchange of emails or phone calls. They also provide CCA developers with an extremely valuable form of direct feedback that supplements more compressed bug reports and after-the-fact anecdotes.

We expect to work closely with the proposed Institute for the Support of the SciDAC Software Ecosystem (ISSSE) [211] both on software engineering processes for the CCA, and disseminating "CCA-friendly" software engineering processes to application groups, and with support staff at major supercomputer centers to insure that their users have the software they need available and user support questions are appropriately addressed.

Community Outreach. Providing shared computing and information resources continues to be a cornerstone of our broader community support and outreach. The `cca-forum.org` server maintained by CCTTSS provides nearly 100 users with a collaborative environment for information sharing and software development, and has inspired the development of similar "collaboration servers" for other projects, such as [212]. We plan major renovation to both the web and software development infrastructure of `cca-forum.org` as a part of this proposal. The web infrastructure will be modernized and we will greatly expand the content. Code hosting capabilities will be updated, or migrated to other SciDAC-supported infrastructure [212]. Through an in-kind contribution from Indiana University, we will also have access to a dedicated 18-processor cluster (see Sec. **??** for more information) that will be preloaded with up-to-date CCA core tools and Toolkit components as an additional resource for community code development, debugging, and general experimentation with CCA technology, even without new users needing to download and install the CCA tools and components locally.

We plan a broad range of other community outreach activities as well. We will work with software and hardware vendors to insure that commercial software tools (i.e., compilers, debuggers, profilers) will work with the CCA environment. We will also work to insure that future HPC computing environments will support component approaches and specifically the CCA by working with projects like the DOE Center for

Programming Models for Scalable Parallel Computing [213] and the Defense Advanced Research Projects Agency (DARPA) High-Productivity Computing Systems (HPCS) [214] efforts. Preliminary discussions have been started between members of the Unified Parallel C (UPC) [215, 216] consortium and the developers of Babel regarding interoperability between UPC and other important HPC languages. Center members have organized and participated in workshops and symposia on HPC component technology in conjunction with SIAM Parallel Processing and Computational Science and Engineering meetings, IPDPS, HPDC, and other venues. Educating the next generation of computational scientists is important, and our activities include at least four completed Ph.D. dissertations [217–220] with four in progress, participation in the annual ACTS Collection [221] workshop series, organized by Lawrence Berkeley National Laboratory (LBNL), student internships at Labs, and the incorporation of CCA into computer science and computational science courses at a number of universities.

1.9 Licensing, Dissemination, and the Software Life Cycle

Openness has long been a hallmark of the CCA Forum. The CCA specification is an open standard. Voting membership in the CCA Forum is earned and maintained by participation (in person or telecommuting) in at least two of the previous three open meetings. The core CCA tools supported by this proposal are already available under licenses conforming to the Open Source Initiative (OSI)'s definition of open source [222]. Moreover, individual software projects within the TASCS already operate in an open fashion; employing websites, mailing lists, discussion forums, open bug tracking systems, public source code repositories, workshops, and wikis to engage and support their users. Development of these tools is still dominated by their institution of origin, but contributions from outside the parent institution—even outside TASCS—are not uncommon, and will continue to be encouraged.

The "software life cycle" has a dual meaning for the CCA. On the one hand, we are working with users to introduce CCA tools and ideas into large-scale, production scientific applications, which requires a strong sensitivity to the life cycle of the application. In this sense, we have designed the CCA architecture to require minimal modification of existing software, and, through our work on the usability of the CCA environment (Sec. 1.6.3) and on the deployment of components (Sec. 1.7) will make CCA technology easier to incorporate at the software level. On the other hand, the CCA tools themselves have a life cycle separate from the applications that use them. Since we started from scratch in 1998, we have ample experience with producing software which is "born" at near-production quality levels, and maintaining it for a growing user base, sensitive to the quality of their software tools. This proposal addresses the importance of quality, well-supported software, devoting a significant effort to the essential tasks of maintaining and supporting the core CCA environment and porting it to the latest HPC platforms (Sec. 1.6.1). As part of this work, we will strengthen documentation, and bug tracking/support for the tools, and bring the extensive regression testing that has been done for Babel for several years (currently 20,000 individual tests, daily) to the other tools.

Longer term, we can envision various models to help sustain and grow the CCA software. One possibility follows the Apache and Mozilla Foundations [223, 224] by formally incorporating the CCA Forum as a non-profit foundation to hold and manage the community's intellectual property. Another possibility is the Red Hat [225] business model, where for-profit companies perform maintenance and support of the CCA and sell support contracts to paying customers. These, and other, options will be considered later in the project, according to our success in bringing component technology into the mainstream of scientific computing, the composition and needs of the user base, our DOE sponsors, and the TASCS team.

1.10 Team, Collaboration, and Management

The project team consists of eleven institutions: six national laboratories (ANL, LANL, LLNL, ORNL, PNNL, and SNL), four universities (BU, IU, UMD, and UU), and a research-based company (Tech-X). Eight of the organizations were the founding members of the CCA Forum in 1998, and constituted the team

for the earlier SciDAC CCTTSS project (2001–2006). The three new additions to the team (BU, Tech-X, and UMD) have been active participants in the CCA Forum for several years, and have strong collaborative ties with the CCTTSS, including joint publications [5, 90, 226, 227]. BU faculty members Chiu and Govindaraju began participating in the CCA as PhD students at IU and Lewis as PI of a CCA-related DOE Early Career Principal Investigator grant.

Though the team is large, we have a history of effective and productive collaboration stretching back, in most cases, seven or more years, at both the institutional and individual levels. Of the more than 50 scientific papers published by the CCTTSS, more than 40% are multi-institutional, and nearly all CCTTSS project activities have also been multi-institutional. Team members are also strong participants in the larger CCA community, through the CCA Forum and other activities. Quarterly Forum meetings provide frequent face-to-face interactions, and are usually accompanied by an additional half or full day of activity-specific meetings. Remote collaboration is a routine matter for team members, who have been using a variety of collaborative tools and resources (see Sec. **??**) to work together for many years already. Tasks are typically undertaken by a small, but multi-institutional teams.

Our approach to the management of the project recognizes both its administrative and technical aspects. Each institution has a lead co-PI, who is responsible for administrative and site-specific matters. Technical areas also have designated leaders, who are responsible for overseeing and coordinating the scientific work of the project. Decisions will be made by consensus of the leadership, with the Lead PI (Bernholdt) acting as arbiter and final decision maker as required. The Lead PI will also serve as the primary point of contact with DOE management.

Apart from the new partners, the only significant change in the leadership from the CCTTSS to TASCS is Bernholdt replacing Armstrong as the Lead PI of the SciDAC center, while Armstrong will continue to chair the CCA Forum. Separating these two leadership positions helps recognize growth of the broader CCA community and is the first step towards formalizing the Forum as an independent body with a distinct leadership and charter.

1.11 Overview of Work Breakdown and Schedule

We have proposed a comprehensive and tightly integrated program of work designed to raise the CCA to the next level of capability, utility, and robustness for petascale computational science. This work will be carried out by a well-established research team, used to working together effectively across institutional borders, and in close collaboration with a number of other SciDAC projects (current and proposed). The proposal comprises roughly 13 FTEs per year, distributed across the eleven participating institutions according to their involvement, resulting in a budget request of $3,989k in the first year, with increases for inflation in subsequent years (complete details in Sec. **??**).

The project is organized into four thrust areas, some of which have several sub-elements. Each major activity has a coordinator who has responsibility for oversight of the work. Each activity is a multi-institutional collaboration. The following tables provide a more detailed picture of the time lines and major milestones of each Focus Area of the project, including the participating institutions. Detailed breakdowns by institution can be found in Sec. **??**.

Table 1.1: **Summary of Milestones for Component Technology Initiatives**
Coordinator: L.C. McInnes, ANL; *Participants:* ANL, IU, LLNL, ORNL, PNNL, SNL, UMD; 4.6 FTEs

Year 1	*Year 2*	*Year 3*	*Years 4–5*
Parallel Coupling Infrastructure *Coordinator:* R. Bramley, IU; *Participants:* ANL, IU, ORNL, SNL, UMD; 1.1 FTEs *Motivating Applications:* fusion simulations			
• Implement file-based coupling. • Develop data management system. • Design coupler component.	• Develop parallel I/O component. • Implement coupler component. • Design generic data management tools.	• Develop CCA interfaces for tightly coupled codes. • Develop cross-discretization interfaces. • Complete data management system.	• Incorporate asynch data sharing & concurrency. • Apply CQoS optimizations to coupling assemblages. • Extend to new applications areas.
Emerging HPC Paradigms *Coordinator:* J. Nieplocha, PNNL; *Participants:* ORNL, PNNL; 1.2 FTEs *Motivating Applications:* biology and quantum chemistry simulations			
• Develop multi-level parallelism model. • Define abstract model for CCA hybrid apps. • Develop a fault-tolerance model for CCA	• Develop CCA model for processor groups. • Develop component interface for hybrid systems. • Implement fault-resilient Ccaffeine.	• Develop simple MCMD programming model. • Define hybrid interface for Cray XD-1. • Implement fault-tolerance services for components.	• Incorporate MCMD support for heterogeneous prog. models. • Implement fault-tolerant, hybrid & MCMD application components.
Software Quality and Verification *Coordinator:* T.L. Dahlgren, LLNL; *Participants:* LLNL, ORNL; 0.6 FTEs *Motivating Applications:* fusion simulations, CQoS initiative			
• Identify and define CQoS and domain-specific semantics; assess spec. mechanisms. • Design method invocation sequencing constraints enforcement.	• Develop semantics prototype(s). • Develop sequencing enforcement prototype in Babel/SIDL.	• Introduce semantic specifications into selected Toolkit components. • Evaluate semantics prototype(s). • Evaluate sequencing enforcement prototype.	• Revise and evaluate prototypes based on new, CQoS event and hybrid/MCMD models.
Computational Quality of Service (CQoS) *Coordinator:* L.C. McInnes, ANL; *Participants:* ANL, SNL; 1.6 FTEs *Motivating Applications:* combustion, quantum chemistry, accelerator, and fusion simulations			
• Populate CQoS testbed, define metrics, perform base experiments. • Collect application requirements and specify initial CQoS API. • Build database comp.	• Develop initial performance models for applications. • Complete design of overall CQoS strategy. • Develop proxy port generation for CQoS usage.	• Implement application control laws. • Implement event-driven version of control infrastructure. • Design APIs for general analysis engines.	• Extend CQoS to MCMD and hybrid computing models. • Apply CQoS tools to parallel coupling. • Stress test CQoS tools.

Table 1.2: **Summary of Milestones for CCA Environment**
Coordinator: G. Kumfert, LLNL; *Participants:* ANL, BU, LANL, LLNL, ORNL, SNL, UU; 4.5 FTEs

Year 1	*Year 2*	*Year 3*	*Years 4–5*
Core Tool Support and Maintenance *Coordinator:* B. Allan, SNL; *Participants:* ANL, LLNL, ORNL, SNL; 1.5 FTEs			
⟵ Support helpdesk and open bugtracking. ⟶			
⟵ Develop and maintain technical documentation. ⟶			
• Port CCA software stack to NLCF machines • Complete CCA Conformance Tests	• Automated conformance testing for all CCA frameworks.	• Automated integration testing for CCA base installation, tutorial source, and toolkit.	• Evaluate and port to new architectures as they emerge.

(continued)

Table 1.2: **Summary of Milestones for CCA Environment** *(continued)*

Year 1	*Year 2*	*Year 3*	*Years 4–5*
Enhancements	*Coordinator:* T. Epperly, LLNL;	*Participants:* BU, LLNL, ORNL, SNL, UU;	2.1 FTEs
• Adopt EventService, MPIService, and CommandLineService into standard. • Demonstrate support for BabeRMI in XCAT. • Add distributed arrays to SIDL/Babel.	• Adopt GuiBuilderService into standard. • Demonstrate CCA/Kepler interoperability. • Add structs to SIDL/Babel. • Add Fortran 2003 support to Babel. • Incorporate SOAP implementation module in BabelRMI and integrate with Proteus and XCAT.	• Finalize specification for Component sub-assemblies. • Develop specification for framework interoperability. • Demonstrate CCA/ESMF interoperability via WebServices. • Add Matlab support to Babel	• Demonstrate exchange of sub-assemblies between two CCA Frameworks. • Demonstrate CCA/VTK interoperability. • Demonstrate framework interoperability between CCA implementations. • Add typemaps support to SIDL/Babel. • Add parallel RMI support to SIDL/Babel.
Usability	*Coordinator:* C. Rasmussen, LANL;	*Participants:* LANL, LLNL, ORNL, SNL;	0.9 FTEs
• Draft CCA-Lite Spec and CCA-Lite Framework. • Document advanced component debugging techniques. • Design component test harness.	• Demonstrate connecting CCA-Lite components to CCA components in Ccaffeine. • Deploy component test harness.	• Demonstrate source-to-source conversion of CCA-Lite component to full CCA Component. • Integrate SIDL semantics enforcement into testing methodology. • Develop component tracing tools to facilitate debugging.	• Incorporate new SIDL features (esp. structs) into source-to-source conversion. • Evaluate tradeoffs in debugging and testing CCA-Lite vs. full CCA. • Apply test harness to selected toolkit components.

Table 1.3: **Summary of Milestones for the CCA Toolkit**
Coordinator: R. Armstrong, SNL; *Participants:* ANL, IU, LLNL, ORNL, PNNL, SNL, Tech-X, UU; 2.2 FTEs

Year 1	*Year 2*	*Year 3*	*Years 4–5*
⟵ Design, establish and, based on user feedback, iterate and improve CCA Base Installation ⟶			
• Design toolkit structure and contribute initial components to the Toolkit, and establish web distribution system	• Incorporate and promulgate Toolkit components into CCA tutorial and outreach activities, improve type and quality of the Toolkit repertoire.	• Add to Toolkit component improvements to CCA architecture since Year 1, e.g. MCMD components, templates, and CQoS plug-ins.	• Establish web-based/community process for approving/distributing component contributions from the community, as user base for Toolkit expands.

Table 1.4: **Summary of Milestones for Application and User Outreach and Support**
Coordinator: D.E. Bernholdt, ORNL; *Participants:* ANL, IU, LLNL, ORNL, PNNL, SNL, Tech-X; 1.8 FTEs

Year 1	*Year 2*	*Year 3*	*Years 4–5*
⟵ Deliver application and user support, incl. tutorials, coding camps, etc. ⟶			
⟵ Update tutorial and best practices documentation. ⟶			
• Revamp `cca-forum.org` web services.	• Revamp or migrate `cca-forum.org` code development services.		

2. Bibliography

[1] U.S. Department of Energy, Office of Science Strategic Plan, `http://www.sc.doe.gov/sub/Mission/Mission_Strategic.htm`, 2004.

[2] Raymond L. Orbach, Maintaining U.S. Scientific Leadership and Global Economic Competitiveness: FY07 Budget Request for the Office of Science, `http://www.sc.doe.gov/Sub/Newsroom/News_Releases/DOE-SC/2006/budget/2-%7%20-%20Master%20FY%2007%20final.pdf`, 2006.

[3] Top500 Supercomputers List, `http://www.top500.org/lists/2005/11/basic`, 2005.

[4] David E. Bernholdt, Robert C. Armstrong, and Benjamin A. Allan, Managing Complexity in Modern High End Scientific Computing through Component-Based Software Engineering, in *Proc. of HPCA Workshop on Productivity and Performance in High-End Computing (P-PHEC 2004), Madrid, Spain*, 2004.

[5] David E. Bernholdt, Benjamin A. Allan, Robert Armstrong, Felipe Bertrand, Kenneth Chiu, Tamara L. Dahlgren, Kostadin Damevski, Wael R. Elwasif, Thomas G. W. Epperly, Madhusudhan Govindaraju, Daniel S. Katz, James A. Kohl, Manoj Krishnan, Gary Kumfert, J. Walter Larson, Sophia Lefantzi, Michael J. Lewis, Allen D. Malony, Lois C. McInnes, Jarek Nieplocha, Boyana Norris, Steven G. Parker, Jaideep Ray, Sameer Shende, Theresa L. Windus, and Shujia Zhou, A Component Architecture for High-Performance Scientific Computing, *Intl. J. High-Perf. Computing Appl.* (2005), Submitted to ACTS Collection special issue.

[6] J. D. de St. Germain, A. Morris, S. G. Parker, A. D. Malony, and S. Shende, Integrating Performance Analysis in the Uintah Software Development Cycle, in *The Fourth International Symposium on High Performance Computing (ISHPC-IV)*, pages 190–206, 2002.

[7] Sameer Shende, Allen D. Malony, Craig Rasmussen, and Matthew Sottile, A Performance Interface for Component-Based Applications, in *International Workshop on Performance Modeling, Evaluation, and Optimization of Parallel and Distributed Systems (PMEO-PDS'03)*, 2003, also available as Technical Report SAND2003-8631, Sandia National Laboratories, Livermore, CA.

[8] J. Ray, N. Trebon, S. Shende, R. C. Armstrong, and A. Malony, Performance Measurement and Modeling of Component Applications in a High Performance Computing Environment : A Case Study, in *Proceedings of the 18^{th} International Parallel and Distributed Processing Symposium*, Los Alamitos, California, USA, 2004, IEEE Computer Society, also Sandia National Laboratories Technical Report SAND2003-8631, November 2003.

[9] Randall Bramley, Rob Armstrong, Lois McInnes, and Matt Sottile, High-Performance Component Software Systems, in M. Heroux, P. Raghavan, and H. Simon, editors, *Proceedings, 2004 SIAM Parallel Processing Conference (PP'04)*, page TBD, SIAM, 2005, invited chapter, accepted. Also available as Indiana University Dept. of Computer Science Technical Report TR068.

[10] A. Malony, S. Shende, N. Trebon, J. Ray, R. Armstrong, C. Rasmussen, and M. Sottile, Performance Technology for Parallel and Distributed Component Software, *Concurrency and Computation: Practice and Experience* **17**, 117 (2005).

[11] Benjamin A. Allan, Robert C. Armstrong, Alicia P. Wolfe, Jaideep Ray, David E. Bernholdt, and James A. Kohl, The CCA Core Specification In A Distributed Memory SPMD Framework, *Concurrency and Computation: Practice and Experience* **14**, 323 (2002).

[12] D. Gannon, S. Krishnan, A. Slominski, G. Kandaswamy, and L. Fang, Building Applications from a Web Service based Component Architecture, in Vladimir Getov and Thilo Kielmann, editors, *Component Models and Systems for Grid Applications. Proceedings of the Workshop on Component Models and Systems for Grid Applications*, Saint Malo, France, 2004, Springer.

[13] D. Gannon, J. Alameda, O. Chipara, M. Christie, V. Dukle, L. Fang, M. Farrellee, S. Hampton, G. Kandaswamy, D. Kodeboyina, S. Krishnan, C. Moad, M. Pierce, B. Plale, A. Rossi, Y. Simmhan, A. Sarangi, A. Slominski, S. Shirasuna, and T. Thomas, Building Grid Portal Applications from a Web-Service Component Architecture, *Proc. IEEE* **93**, 551 (2005), invited article.

[14] Dennis Gannon, Liang Fang, Gopi Kandaswamy, D. Kodeboyina, Sriram Krishnan, Beth Plale, and Aleksander Slominski, Building Grid Applications and Portals: An Approach Based on Components, Web Services and Workflow Tools, in *ACM/IFIP/IEEE Euro-Par*, 2004.

[15] Madhusudhan Govindaraju, Sriram Krishnan, Kenneth Chiu, Aleksander Slominski, Dennis Gannon, and Randall Bramley, Merging the CCA Component Model with the OGSI Framework, in *Proceedings of CCGrid2003, 3rd International Symposium on Cluster Computing and the Grid*, pages 182–189, Tokyo, Japan, 2003.

[16] Dennis Gannon, Rachana Ananthakrishnan, Sriram Krishnan, Madhusudhan Govindaraju, Lavanya Ramakrishnan, and Aleksander Slominski, *Grid Computing: Making the Global Infrastructure a Reality*, chapter 9, Grid Web Services and Application Factories, Wiley, 2003.

[17] Dennis Gannon, Randall Bramley, Geoffrey Fox, Shava Smallen, Al Rossi, Rachana Ananthakrishnan, Felipe Bertrand, Ken Chiu, Matt Farrellee, Madhu Govindaraju, Sriram Krishnan, Lavanya Ramakrishnan, Yogesh Simmhan, Alek Slominski, Yu Ma, Caroline Olariu, and Nicolas Rey-Cenvaz, Programming the Grid: Distributed Software Components, P2P and Grid Web Services for Scientific Applications, *J. Cluster Computing* **5**, 325 (2002).

[18] Madhusudhan Govindaraju, Sriram Krishnan, Kenneth Chiu, Aleksander Slominski, Dennis Gannon, and Randall Bramley, XCAT 2.0: Design and Implementation of Component based Web Services, Technical Report TR562, Department of Computer Science, Indiana University, Bloomington, 2002.

[19] Lavanya Ramakrishnan, Helen Nell Rehn, Jay Alameda, Rachana Ananthakrishnan, Madhusudhan Govindaraju, Aleksander Slominski, Kay Connelly, Von Welch, Dennis Gannon, Randall Bramley, and Shawn Hampton, An Authorization Framework for a Grid Based Common Component Architecture, in *Proceedings of the 3rd International Workshop on Grid Computing*, pages 169–180, Baltimore, Maryland, 2002, Springer Press.

[20] Scott Kohn, Gary Kumfert, Jeff Painter, and Cal Ribbens, Divorcing Langauge Dependencies from a Scientific Software Library, in *Proc. 10th SIAM Conf. Parallel Process.*, Portsmouth, VA, 2001, Also available as Lawrence Livermore National Laboratory technical report UCRL-JC-140349.

[21] Robert Armstrong and Renata McCoy, The Common Component Architecture: Fostering an Open Source Community in High Performance Computing, in *Proc. Of Advanced School for Computing and Imaging 2003 Conf.*, Heijen, Netherlands, 2003.

[22] Microsoft Corporation, Component Object Model Specification, `http://www.microsoft.com/com/resources/comdocs.asp`, 1999.

[23] Object Management Group, CORBA Component Model, `http://www.omg.org/technology/documents/formal/components.htm`, 2002.

[24] Sun Microsystems, Enterprise JavaBeans Downloads and Specifications, `http://java.sun.com/products/ejb/docs.html`, 2004.

[25] David E. Bernholdt, Wael R. Elwasif, and James A. Kohl, Communication Infrastructure in High-Performance Component-Based Scientific Computing, in Dieter Kranzlmüller, Peter Kacsuk, Jack Dongarra, and Jens Volkert, editors, *Recent Advances in Parallel Virtual Machine and Message Passing Interface. 9th European PVM/MPI User's Group Meeting Linz, Austria, September/October 2002. Proceedings*, volume 2474 of *Lecture Notes in Computer Science*, pages 260–270, Springer, 2002.

[26] MPI-2, Message Passing Interface Forum. MPI-2: Extensions to the Message-Passing Interface, `http://www.mpi-forum.org`.

[27] G. A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam, *PVM: Parallel Virtual Machine, A User's Guide and Tutorial for Networked Parallel Computing*, MIT Press, Cambridge, MA, 1994.

[28] J. Nieplocha, R.J. Harrison, and R.J. Littlefield, Global Arrays: A Nonuniform Memory Access Programming Model for High-Performance Computers, *The Journal of Supercomputing* **10**, 197 (1996), See http://www.emsl.pnl.gov:2080/docs/global.

[29] Jarek Nieplocha, Bruce Palmer, Vinod Tipparaju, Manojkumar Krishnan, Harold Trease, and Edo Apra, Advances, Applications and Performance of the Global Arrays Shared Memory Programming Toolkit, *Intl. J. High-Perf. Computing Appl.* (2006), Submitted to ACTS Collection special issue, to appear.

[30] Madhusudhan Govindaraju, Michael R. Head, and Kenneth Chiu, XCAT-C++: Design and Performance of a Distributed CCA Framework, in *Proceedings of the 12th Annual IEEE International Conference on High Performance Computing (HiPC)*, pages 270–279, 2005.

[31] Sriram Krishnan and Dennis Gannon, Checkpoint and Restart for Distributed Components in XCAT3, in *5th IEEE/ACM International Workshop on Grid Computing*, 2004.

[32] Sriram Krishnan, Randall Bramley, Dennis Gannon, Madhusudhan Govindaraju, Rahul Indurkar, Aleksander Slominski, Benjamin Temko, Richard Alkire, Timothy Drews, Eric Webb, and Jay Alameda, The XCAT Science Portal, in *Proceedings of SC2001, Denver, Colorado*, 2001.

[33] Kenneth Chiu, Madhusudhan Govindaraju, and Dennis Gannon, The Proteus Multiprotocol Library, in *Proceedings of SC2002, Baltimore, Maryland*, 2002.

[34] Kenneth Chiu, XBS: A Streaming Binary Serializer for High Performance Computing, in *Proceedings of the High Performance Computing Symposium 2004*, 2004.

[35] Deger Cenk Erdil, Kenneth Chiu, Madhusudhan Govindaraju, and Michael J. Lewis, A Proteus-Mediated Communications Substrate for LegionCCA and XCAT-C++, *Workshop on Component Models and Frameworks in High Performance Computing (CompFrame 2005)* (2005).

[36] Michael J. Lewis, Madhusudhan Govindaraju, and Kenneth Chiu, Exploring the Design Space for CCA Framework Interoperability Approaches, *Workshop on Component Models and Frameworks in High Performance Computing (CompFrame 2005)* (2005).

[37] C. R. Johnson, S. G. Parker, and D. M. Weinstein, Component-Based Problem Solving Environments for Large-Scale Scientific Computing, *Concurrency and Computation: Practice and Experience* **14**, 1337 (2002).

[38] Keming Zhang, Kostadin Damevski, Venkatanand Venkatachalapathy, and Steven Parker, SCIRun2: A CCA Framework for High Performance Computing, in *Proceedings of the 9th International Workshop on High-Level Parallel Programming Models and Supportive Environments (HIPS 2004)*, pages 72–79, IEEE Computer Society, 2004.

[39] S. Parker, K. Zhang, K. Damevski, and C. Johnson, Integrating Component-Based Scientific Computing Software, *Frontiers of Parallel Processing For Scientific Computing* (2005).

[40] Kostadin Damevski and Steven Parker, Imprecise Exceptions in Distributed Parallel Components, in Marco Danelutto, Domenico Laforenza, and Marco Vanneschi, editors, *Euro-Par 2004 Parallel Processing, 10 International Euro-Par Conference Pisa, Italy, August/September 2004 Proceedings*, volume 3149 of *Lecture Notes in Computer Science*, pages 108–116, Springer, 2004.

[41] K. Damevski and S. Parker, Parallel Remote Method Invocation and M-by-N Data Redistribution, in *Proceedings, Fourth LACSI Symposium*, Los Alamos Computer Science Institute, 2003, published on CD-ROM.

[42] K. Damevski, Parallel Component Interaction with an Interface Language Compiler, Master's thesis, University of Utah, 2003.

[43] S. G. Parker, A Component-based Architecture for Parallel Multi-Physics PDE Simulation, in *International Conference on Computational Science (ICCS2002) Workshop on PDE Software*, 2002.

[44] Tamara Dahlgren, Thomas Epperly, Gary Kumfert, and James Leek, *Babel User's Guide*, CASC, Lawrence Livermore National Laboratory, Livermore, CA, babel-0.11.0 edition, 2005.

[45] David E. Bernholdt, Wael R. Elwasif, James A. Kohl, and Thomas G. W. Epperly, A Component Architecture for High-Performance Computing, in *Proceedings of the Workshop on Performance Optimization via High-Level Languages and Libraries (POHLL-02), 16th Annual ACM International Conference on Supercomputing (ICS'02)*, New York, 2002.

[46] Tamara L. Dahlgren and Premkumar T. Devanbu, Adaptable Assertion Checking for Scientific Software Components, in Philip M. Johnson, editor, *Proceedings of the First International Workshop on Software Engineering for High Performance Computing System Applications (SE-HPCS)*, pages 64–69, Edinburgh, Scotland, UK, 2004, (also available as Lawrence Livermore National Laboratory Technical Report UCRL-CONF-202898).

[47] Tamara L. Dahlgren and Premkumar T. Devanbu, Improving Scientific Software Component Quality Through Assertions, in *Proceedings of the Second International Workshop on Software Engineering for High Performance Computing System Applications*, pages 73–77, St. Louis, Missouri, 2005, Also available as Lawrence Livermore National Laboratory Technical Report UCRL-CONF-211000, Livermore, CA, 2005.

[48] Tamara L. Dahlgren and Premkumar T. Devanbu, An Empirical Comparison of Adaptive Assertion Enforcement Performance, Technical Report UCRL-CONF-206305, Lawrence Livermore National Laboratory, 2004.

[49] Gary Kumfert and James Leek, Babel/SIDL Changes to Support RMI, Technical Report UCRL-TR-213497, Lawrence Livermore National Lab, 2005.

[50] The Eclipse project, `http://www.eclipse.org/`.

[51] Lois Curfman McInnes, Benjamin A. Allan, Robert Armstrong, Steven J. Benson, David E. Bernholdt, Tamara L. Dahlgren, Lori Freitag Diachin, Manojkumar Krishnan, James A. Kohl, J. Walter Larson, Sophia Lefantzi, Jarek Nieplocha, Boyana Norris, Steven G. Parker, Jaideep Ray, and Shujia Zhou, Parallel PDE-Based Simulations Using the Common Component Architecture, in Are Magnus Bruaset, Petter Bjørstad, and Aslak Tveito, editors, *Numerical Solution of PDEs on Parallel Computers*, volume 51 of *Lecture Notes in Computational Science and Engineering (LNCSE)*, pages 327–384, Springer-Verlag, 2006, Also available as Argonne National Laboratory technical report ANL/MCS-P1179-0704.

[52] Lois Curfman McInnes and Lori F. Diachin, Software Components in High-Performance Scientific Computation, Article in SIAM News, vol 38, Number 5, June 2005.

[53] D. Keyes (PI), Terascale Optimal PDE Simulations (TOPS) Center, `http://tops-scidac.org/`, 2005.

[54] B. Smith et al., TOPS Solver Components, `http://www.mcs.anl.gov/scidac-tops/solver-components/tops.html`, 2005.

[55] Terascale Simulation Tools and Technologies Center, `http://www.tstt-scidac.org/`.

[56] Carl Ollivier-Gooch, Kyle Chand, Tamara Dahlgren, Lori Freitag Diachin, Brian Fix, Jason Kraftcheck, Xiaolin Li, Eunyoung Seol, Mark Shephard, Timothy Tautges, and Harold Trease, The TSTT Mesh Interface, in *Proceedings of the 44th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, NV, 2006.

[57] CCA Forum, The CCA Component Toolkit, `https://www.cca-forum.org/wiki/tiki-index.php?page=Toolkit`.

[58] DOE Office of Science, Top Ten DOE Office of Science Achievements in 2002, `http://www.sc.doe.gov/sub/accomplishments/top\_10.htm`, 2002.

[59] Boyana Norris, Satish Balay, Steve Benson, Lori Freitag, Paul Hovland, Lois McInnes, and Barry Smith, Parallel Components for PDEs and Optimization: Some Issues and Experiences, *Parallel Computing* **28**, 1811 (2002), (also available as Argonne preprint ANL/MCS-P932-0202).

[60] J. Walter Larson, Boyana Norris, Everest T. Ong, David E. Bernholdt, John B. Drake, Wael R. Elwasif, Michael W. Ham, Craig E. Rasmussen, Gary Kumfert, Daniel S. Katz, Shujia Zhou, Cecelia DeLuca, and Nancy S. Collins, Components, the Common Component Architecture, and the Climate/Weather/Ocean Community, in *84th American Meteorological Society Annual Meeting*, Seattle, Washington, 2004, American Meteorological Society.

[61] Sophia Lefantzi and Jaideep Ray, A Component-based Scientific Toolkit for Reacting Flows, in *Proceedings of the Second MIT Conference on Computational Fluid and Solid Mechanics*, volume 2, pages 1401–1405, Boston, Mass., 2003, Elsevier Science.

[62] Sophia Lefantzi, Jaideep Ray, Christopher A. Kennedy, and Habib N. Najm, A Component-based Toolkit for Reacting Flows with High Order Spatial Discretizations on Structured Adaptively Refined Meshes, *Progress in Computational Fluid Dynamics* **5**, 298 (2005).

[63] Joseph P. Kenny, Steven J. Benson, Yuri Alexeev, Jason Sarich, Curtis L. Janssen, Lois Curfman Mcinnes, Manojkumar Krishnan, Jarek Nieplocha, Elizabeth Jurrus, Carl Fahlstrom, and Theresa L. Windus, Component-Based Integration of Chemistry and Optimization Software, *J. of Computational Chemistry* **24**, 1717 (2004).

[64] S. Benson, M. Krishnan, L. McInnes, J. Nieplocha, and J. Sarich, Using the GA and TAO Toolkits for Solving Large-Scale Optimization Problems on Parallel Computers, Technical Report ANL/MCS-P1084-0903, Argonne National Laboratory, 2003.

[65] Steven J. Benson, Lois Curfman McInnes, Jorge Moré, and J. Sarich, Scalable Algorithms in Optimization: Computational Experiments, in *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Albany, NY, 2004.

[66] Yu Ma and Randall Bramley, A Composable Data Management Architecture for Scientific Applications, in *Proceedings of Challenges of Large Applications in Distributed Environments*, Research Triangle Park, NC, 2005.

[67] Yu Ma, Randall Bramley, and Sun Kim, A Data Management Architecture for Computational Biology, Technical Report 606, Computer Science Dept, Indiana University, Bloomington, Indiana, 2005.

[68] J. Sarich, A Programmer's Guide for Providing CCA Component Interfaces to the Toolkit for Advanced Optimization, Technical Report ANL/MCS-TM-279, Argonne National Laboratory, 2004.

[69] Wael R. Elwasif, Donald B. Batchelor, David E. Bernholdt, Lee A. Berry, Ed F. D'Azevedo, Wayne A. Houlberg, E. F. Jaeger, James A. Kohl, and Shuhui Li, Coupled Fusion Simulation Using the Common Component Architecture, in Vaidy Sunderam, Geert Dick van Albada, Peter M. A. Sloot, and Jack J. Dongarra, editors, *Computational Science – ICCS 2005 5th International Conference, Atlanta, USA, May 22–25, 2005, Proceedings, Part I*, volume 3514 of *Lecture Notes in Computer Science*, pages 372–379, Atlanta, Georgia, USA, 2005, Springer.

[70] Jason Jones and Masha Sosonkina, SPARSKIT CCA Component, in *CompFrame 2005 Workshop on Component Models and Frameworks in High Performance Computing*, Atlanta, Georgia, USA, 2005.

[71] Randall Bramley, Kenneth Chiu, Shridhar Diwan, Dennis Gannon, Madhusudhan Govindaraju, Nirmal Mukhi, Benjamin Temko, and Madhuri Yechuri, A Component Based Services Architecture for Building Distributed Applications, in *Proceedings of the High Performance Distributed Computing Conference*, 2000.

[72] Boyana Norris and Ivana Veljkovic, Performance Monitoring and Analysis Components in Adaptive PDE-Based Simulations, Technical Report ANL/MCS-P1221-0105, Argonne National Laboratory, 2005.

[73] M. Daz, D. Garrido, S. Romero, B. Rubio, E. Soler, , and J.M. Troya, Nuclear Power Plant Simulators: A Component-based Approach, in *Proceedings of Applied Simulation and Modelling - 2005*, Benalmdena, Spain, 2005, IASTED.

[74] Johan Steensland and Jaideep Ray, A Partitioner-Centric Model for SAMR Partitioning Trade-Off Optimization: Part I, in *Proceedings of the 4th Annual Symposium of the Los Alamos Computer Science Institute (LACSI04)*, 2003.

[75] Johan Steensland and Jaideep Ray, A Heuristic Re-Mapping Algorithm Reducing Inter-Level Communication in SAMR Applications, in *Proceedings of the 15th IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS03)*, volume 2, pages 707–712, ACTA PRESS, 2003, also available as Sandia Technical Report, SAND2003-8310.

[76] H. Najm (PI), Computational Facility for Reacting Flow Science (CFRFS), `http://cfrfs.ca.sandia.gov`.

[77] B. A. Allan, S. Lefantzi, and Jaideep Ray, The scalability impact of a component-based software engineering framework on a growing SAMR toolkit: A case study, in *Proceedings of Parallel Computational Fluid Dynamics*, Elsevier/North Holland, 2005.

[78] Habib Najm, private communication, 2005.

[79] J. Ray, C. A. Kennedy, S. Lefantzi, and H. N. Najm, Using High-Order Methods on Adaptively Refined Block-Structured Meshes–Discretizations, Interpolations, and Filters, *SIAM Journal on Scientific Computing* (2006), in review.

[80] E. Aprà, T. L. Windus, T. P. Straatsma, E. J. Bylaska, W. de Jong, S. Hirata, M. Valiev, M. Hackler, L. Pollack, K. Kowalski, R. Harrison, M. Dupuis, D. M. A. Smith, J. Nieplocha, V. Tipparaju, M. Krishnan, A. A. Auer, E. Brown, G. Cisneros, G. Fann, H. Früchtl, J. Garza, K. Hirao, R. Kendall, J. Nichols, K. Tsemekham, K. Wolinski, J. Anchell, D. Bernholdt, P. Borowski, T. Clark, D. Clerc, H. Dachsel, M. Deegan, K. Dyall, D. Elwood, E. Glendening, M. Gutowski, A. Hess, J. Jaffee, B. Johnson, J. Ju, R. Kobayashi, R. Kutteh, Z. Lin, R. Littlefield, X. Long, B. Meng, T. Nakajima, S. Niu, M. Rosing, G. Sandrone, M. Stave, H. Taylor, G. Thomas, J. van Lenthe, A. Wong, and Z. Zhang, *NWChem, A Computational Chemistry Package for Parallel Computers, Version 4.7*, Pacific Northwest National Laboratory, Richland, Washington 99352–0999, USA, 2005.

[81] NWChem Home Page:, `http://www.emsl.pnl.gov/docs/nwchem/nwchem.html`.

[82] Ricky A. Kendall, Edo Aprà, David E Bernholdt, Eric J. Bylaska, Michel Dupuis, George I. Fann, Robert J. Harrison, Jailin Ju, Jeffrey A. Nichols, Jarek Nieplocha, T. P. Straatsma, Theresa L. Windus, and Adrian T. Wong, High Performance Computational Chemistry; Overview of NWChem a Distributed Parallel Application, *Computer Physics Communications* **128**, 260 (2000).

[83] C. L. Janssen, Quantum Chemistry on Clusters, *ClusterWorld* **2** (2004).

[84] The Massively Parallel Quantum Chemistry Program, `http://www.mpqc.org/`.

[85] M.S. Gordon and M.W. Schmidt, Advances in Electronic Structure Theory: GAMESS a Decade Later, in C. E. Dykstra, G. Frenking, K.S. Kim, and G.E. Scuseria, editors, *Theory and Applications of Computational Chemistry*, pages 372–379, Elsevier, 2005.

[86] The General Atomic and Molecular Electronic Structure System (GAMESS) Home Page, `http://www.msg.ameslab.gov/GAMESS/GAMESS.html`.

[87] Edward F. Valeev and Curtis L. Janssen, Second-order Moller-Plesset theory with linear R12 terms (MP2-R12) revisited: auxiliary basis set method and massively parallel implementation, *Journal of Chemical Physics* **121**, 1214 (2004).

[88] Manojkumar Krishnan, Yuri Alexeev, Theresa L Windus, and Jarek Nieplocha, Multilevel Parallelism in Computational Chemistry using Common Component Architecture and Global Arrays, in *Proceedings of SuperComputing*, ACM and IEEE, 2005.

[89] Theresa L. Windus, private communication, 2005.

[90] Felipe Bertrand, Randall Bramley, Kostadin B. Damevski, James A. Kohl, David E. Bernholdt, Jay W. Larson, and Alan Sussman, Data Redistribution and Remote Method Invocation in Parallel Component Architectures, in *Proceedings of the 19th International Parallel and Distributed Processing Symposium: IPDPS 2005*, 2005, Best Paper Award. Also available as Indiana University Dept. of Computer Science Technical Report TR064.

[91] Center for Simulation of RF Wave Interactions with Magnetohydrodynamics SWIM, `http://cswim.org/`.

[92] Mark Gordon (PI), Modernizing Legacy Codes: Improving Communication Layer and Scalability for Massively Parallel Systems, SciDAC Basic Energy Sciences SAP proposal, March 2006.

[93] T.L. Windus (PI), NWChem, Part of the Environmental Molecular Sciences Laboratory DOE Project.

[94] T.P. Straatsma (PI), Data Intensive Computing For Complex Biological Systems, DOE Project.

[95] (CFRFS) Computational Facility for Reacting Flow Science, `http://www.ca.sandia.gov/cfrfs/`.

[96] Jaideep Ray (PI), Ultrascale Strategies for Reacting Flow Simulations on Block-structured Adaptive Meshes, SciDAC Basic Energy Sciences SAP proposal, March 2006.

[97] Panagiotis Spentzouris (PI), Accelerator Community Code Development and Discovery Project, SciDAC High Energy Physics (HEP) Scientific Application proposal, March 2006.

[98] John Cary (PI), Framework Application for Core-Edge Transport Simulations, SciDAC Fusion Energy Sciences (OFES) Scientific Application proposal, March 2006.

[99] William Gropp (PI), Efficient and Scalable Enhancements for VORPAL, SciDAC Scientific Application Partnership proposal, March 2006.

[100] The Scientific Data Management (SDM) Home Page, `http://sdm.lbl.gov/sdmcenter/`.

[101] Kepler: Kepler Project, `http://kepler-project.org/`.

[102] C.S. Chang (PI), Center for Plasma Edge Simulation, `http://www.cims.nyu.edu/cpes/`, 2005, DOE SciDAC Fusion Simulation Prototype Center for Plasma Edge Simulations.

[103] Katarzyna Keahey and Dennis Gannon, PARDIS: CORBA-based Architecture for Application-level Parallel Distributed Computation, in *Proceedings of the 1997 International Conference on Supercomputing*, San Jose, CA, 1997.

[104] Peter Beckman, Pat Fasel, William Humphrey, and Sue Mniszewski, Efficient Coupling of Parallel Applications Using PAWS., in *Proceedings of the High Performance Distributed Computing Conference*, Chicago, IL, 1998.

[105] Kate Keahey, Pat Fasel, and Sue Mniszewski, PAWS: Collective Interactions and Data Transfers, in *Proceedings of the High Performance Distributed Computing Conference*, San Francisco, CA, 2001.

[106] Caltech Center for Simulation of Dynamic Response of Materials, `http://www.cacr.caltech.edu/ASAP/index.html`.

[107] University of Illinois at Urbana-Champaign, Center for Simulation of Advanced Rockets, `http://www.csar.uiuc.edu/`.

[108] Donald Batchelor (PI), Center for Simulation of Wave Interactions with Magnetohydrodynamics, SciDAC project, 2005-2007.

[109] Habib Najm (PI), Computational Facility for Reacting Flow Science, Existing SciDAC Basic Energy Sciences project, March 2006.

[110] Lori Diachin (PI), Interoperable Technologies for Advanced Petascale Simulations, SciDAC CET proposal, March 2006.

[111] David Keyes (PI), Towards Optimal PDE Simulations (TOPS), SciDAC CET proposal, March 2006.

[112] Arie Soshani (PI), Scientific Data Management Center, SciDAC CET proposal, March 2006.

[113] Alan Sussman (PI), Institute for the Coupling of High-Performance Simulations (ICS), SciDAC Institute proposal, March 2006.

[114] Climate and UCAR Global Dynamic Division, Community Climate System Model, `http://www.cgd.ucar.edu/csm/`.

[115] University Corporation for Atmospheric Research, Earth System Model Framework, `http://www.esmf.ucar.edu/`.

[116] K. Sklower, H. Robinson, C. R. Mechoso, L. A. Drummond, J. A. Spahr, and J. D. Farrara, The Distributed Data Broker: A decentralized mechanism for periodic exchange of fields between multiple ensembles of parallel computations, Technical report, Computer Science Dept., University of California, Berkeley, 2002.

[117] Rajeev Thakur, William Gropp, and Ewing Lusk, Data Sieving and Collective I/O in ROMIO, in *Proceedings of the Seventh Symposium on the Frontiers of Massively Parallel Computation*, pages 182–189, IEEE Computer Society Press, 1999.

[118] Rajeev Thakur, William Gropp, and Ewing Lusk, An Abstract-Device Interface for Implementing Portable Parallel-I/O Interfaces, in *Proceedings of the Sixth Symposium on the Frontiers of Massively Parallel Computation*, pages 180–187, 1996.

[119] M. Ranganathan, A. Acharya, G. Edjlali, A. Sussman, and J. Saltz, A Runtime Coupling of Data-parallel Programs, in *Proceedings of the 1996 International Conference on Supercomputing*, Philadelphia, PA, 1996.

[120] J. Larson, R. Jacob, I. Foster, and J. Guo, The Model Coupling Toolkit, in Vassil N. Alexandrov, Jack Dongarra, Benjoe A. Juliano, René S. Renner, and Chih Jeng Kenneth Tan, editors, *International Conference on Computational Science*, volume 2073 of *Lecture Notes in Computer Science*, Springer, 2001.

[121] Keming Zhang, Kostadin Damevski, Venkatanand Venkatachalapathy, and Steven G. Parker, SCIRun2: A CCA Framework for High Performance Computing, in *Proceedings of the 9th International Workshop on High-Level Parallel Programming Models and Supportive Environments (HIPS'04)*, pages 72–79, Santa Fe, NM, 2004, IEEE Press.

[122] Felipe Bertrand and Randall Bramley, DCA: A distributed CCA framework based on MPI, in *Proceedings of the 9th International Workshop on High-Level Parallel Programming Models and Supportive Environments (HIPS'04)*, pages 80–89, Santa Fe, NM, 2004, IEEE Press.

[123] Jae-Yong Lee and Alan Sussman, High Performance Communication Between Parallel Programs, in *Proceedings of 2005 Joint Workshop on High-Performance Grid Computing and High-Level Parallel Programming Models (HIPS-HPGC 2005)*, IEEE Computer Society Press, 2005.

[124] Felipe Bertrand, Yongquan Yuan, Kenneth Chiu, and Randall Bramley, An Approach to Parallel MxN Communication, in *Proceedings of the Los Alamos Computer Science Institute (LACSI) Symposium*, Santa Fe, NM, 2003.

[125] Lydia Harper and Brian Kauffman, Community Climate System Model, `http://www.ccsm.ucar.edu/`, 2004.

[126] National Leadership Computing Facility, `http://www.ccs.ornl.gov/nlcf/`.

[127] Ilkay Altintas, Adam Birnbaum, Kim Baldridge, Wibke Sudholt, Mark Miller, Celine Amoreira, Yohann Potier, and Bertram Ludaescher, A Framework for the Design and Reuse of Grid Workflows, in *Iternational Workshop on Scientific Applications on Grid Computing (SAG'04), Lecture Notes in Computer Science 3458*, 2004.

[128] MCT–The Model Coupling Toolkit:, `http://www-unix.mcs.anl.gov/mct/`.

[129] James A. Kohl, Torsten Wilde, and David E. Bernholdt, CUMULVS: Interacting with High-Performance Scientific Simulations, for Visualization, Steering and Fault Tolerance, *Intl. J. High-Perf. Computing Appl.* (2005), Accepted to ACTS Collection special issue.

[130] Jarek Nieplocha (PI), Scalable Fault Tolerance Project, DOE FastOS Program, 2005.

[131] Stephen Scott (PI), Center for Reliability, Availability, and Serviceability for Peta-Scale High-End Computing, SciDAC CET proposal, March 2006.

[132] Pete Beckman (PI), Center for Coordinated Fault Tolerance for High Performance Computing, SciDAC CET proposal, March 2006.

[133] J. Davison de St. Germain, John McCorquodale, Steven G. Parker, and Christopher R. Johnson, Uintah: A Massively Parallel Prolem Solving Environment, in *Proceedings of the Ninth IEEE International Symposium on High Performance and Distributed Computing*, August 2000.

[134] Nancy A. Lynch, *Distributed Algorithms*, Morgan Kaufmann Publishers, 1996.

[135] J.M. Malard, A. Heredia-Langner, D.J. Baxter, K.H. Harman, and W.R. Cannon, Constrained de novo Peptide Identification via multi-objective optimization, in *Proceedings of the 18th International Parallel and Distributed Processing Symposiun*, IEEE, 2004.

[136] William R. Cannon, Kristin H. Harman, Bobbie-Jo Webb-Robertson, Douglas J. Baxter, Christopher S. Oehmen, Kenneth D. Jarman, Alejandro Heredia-Langner, Kenneth J. Auberry, and Gordon A. Anderson, Comparison of Probability and Likelihood Model for Peptide Identification from Tandem Mass Spectrometry Data, *Journal of Proteome Research* **4**, 1687 (2005).

[137] David Brown, Lori Freitag, and Jim Glimm, Creating Interoperable Meshing and Discretization Technology: The Terascale Simulation Tools and Technologies Center, in *Proceedings of the 8th International Conference on Numerical Grid Generation in Computational Field Simulations*, pages 57–61, Honolulu, HI, 2002, Also available as Lawrence Livermore National Laboratory Technical Report UCRL-PRES-151494, Livermore, CA, 2002.

[138] Antoine Beugnard, Jean-Marc Jézéquel, Noël Plouzeau, and Damien Watkins, Making Components Contract Aware, *IEEE Computer* **32**, 38 (1999).

[139] Friedrich W. Beichter, Otthein Herzog, and Heiko Petzsch, SLAN-4 — A Software Specification and Design Language, *IEEE Transactions on Software Engineering* **SE-10**, 155 (1984).

[140] Sebastian Elbaum and John C. Munson, Investigating Software Failures with a Software Black Box, in *Proceedings of the 2000 IEEE Aerospace Conference*, pages 547–566, 2000.

[141] Sebastian Elbaum, Satya Kanduri, and Anneliese Amschler Andrews, Anomalies as Precursors of Field Failures, in *Proceedings of the 14th International Symposium on Software Reliability Engineering (ISSRE'03)*, pages 108–118, 2003.

[142] The Jass Page, `http://csd.informatik.uni-oldenburg.de/~jass`, 2003.

[143] David S. Rosenblum, A Practical Approach to Programming with Assertions, *IEEE Transactions on Software Engineering* **21**, 19 (1995).

[144] Tamara L. Dahlgren, Adaptive Enforcement of Component Interface Assertions, Technical report, Lawrence Livermore National Laboratory, Livermore, California, 2006, in progress.

[145] Bertrand Meyer, *Object-Oriented Software Construction*, Prentice-Hall, Upper Saddle River, NJ, 1997, Second Edition.

[146] Boyana Norris, Jaideep Ray, Rob Armstrong, Lois C. McInnes, David E. Bernholdt, Wael R. Elwasif, Allen D. Malony, and Sameer Shende, Computational Quality of Service for Scientific Components, in Ivica Crnkovic, Judith A. Stafford, Heinz W. Schmidt, and Kurt Wallnau, editors, *Proceedings of the International Symposium on Component-Based Software Engineering (CBSE7)*, volume 3054 of *Lecture Notes in Computer Science*, pages 264–271, Edinburgh, Scotland, 2004, Springer, (also available as Argonne preprint ANL/MCS-P1131-0304).

[147] Lois Curfman McInnes, Jaideep Ray, Rob Armstrong, Tamara L. Dahlgren, Allen Malony, Boyana Norris, Sameer Shende, Joseph P. Kenny, and Johan Steensland, Computational Quality of Service for Scientific CCA Applications: Composition, Substitution, and Reconfiguration, Technical Report ANL/MCS-P1326-0206, Argonne National Laboratory, 2006, Available via `ftp://info.mcs.anl.gov/pub/tech_reports/reports/P1326.pdf`.

[148] J. Steensland, *Efficient partitioning of dynamic structured grid hierarchies*, PhD thesis, University of Uppsala, Uppsala University Library, Box 510, SE-751, 20 Uppsala, Sweden, 2002.

[149] Johan Steensland and Jaideep Ray, A Partitioner-Centric Model for SAMR Partitioning Trade-Off Optimization: Part I, *International Journal of High Performance Computing Applications* **19**, 1 (2005).

[150] N. Trebon, A. Morris, J. Ray, S. Shende, and A. Malony, Performance Modeling of Component Assemblies with TAU, Presented at Compframe 2005 workshop, Atlanta, June, 2005.

[151] S. Bhowmick, L. C. McInnes, B. Norris, and P. Raghavan, The role of multi-method linear solvers in PDE-based simulations, in *Lecture Notes in Computer Science*, volume 2667, pages 828–839, 2003, Computational Science and its Applications-ICCSA 2003.

[152] P. Hovland, K. Keahey, L. C. McInnes, B. Norris, L. F. Diachin, and P. Raghavan, A Quality of Service Approach for High-Performance Numerical Components, in *Proceedings of Workshop on QoS in Component-Based Software Engineering, Software Technologies Conference, Toulouse, France*, 2003, (also available as Argonne preprint ANL/MCS-P1028-0203).

[153] Allen D. Malony (PI), Performance Engineering Technology for Scientific Component Software, DOE, Office of Science, Contract No. DE-FG02-03ER25561, 8/15/03 – 8/14/06.

[154] Allen D. Malony, Sameer Shende, and Alan Morris, Knowledge-based Parallel Performance Technology for Quality Optimization of Scientific Applications, DOE, Office of Science whitepaper in preparation, March, 2006.

[155] D. Bailey (PI), Performance Evaluation Research Center (PERC), `http://perc.nersc.gov/`, 2005.

[156] Bob Lucas (PI), Performance Evaluation Research Center (PERC), SciDAC CET proposal, March 2006.

[157] S. Bhowmick, V. Eijkhout, Y. Freund, E. Fuentes, and D. Keyes, Application of Machine Learning to Selecting Solvers for Sparse Linear Systems, Presentation at the 2006 SIAM Conference on Parallel Processing, San Francisco, CA, 2006.

[158] H. Liu and M. Parashar, Enabling Self-Management of Component Based High-Performance Scientific Applications, in *Proceedings of the 14th IEEE International Symposium on High Performance Distributed Computing*, IEEE Computer Society Press, 2005.

[159] V. Eijkhout and E. Fuentes, A proposed standard for matrix metadata, Technical Report ICL-UT 03-02, University of Tennessee, 2003.

[160] Jack Dongarra and Victor Eijkhout, Self-adapting Numerical Software for Next Generation Applications, *International Journal of High Performance Computing Applications* **17**, 125 (2003), also Lapack Working Note 157, ICL-UT-02-07.

[161] Thomas Eidson, Jack Dongarra, and Victor Eijkhout, Applying Aspect-Orient Programming Concepts to a Component-based Programming Model, in *Proceedings of the 17th International Parallel and Distributed Processing Symposium (IPDPS) April 22–26, 2003, Nice, France*, 2003.

[162] Self-Adapting Large-scale Solver Architecture, see `http://icl.cs.utk.edu/salsa`, 2006.

[163] R. Clint Whaley and Antoine Petitet, Minimizing development and maintenance costs in supporting persistently optimized BLAS, *Software: Practice and Experience* **35**, 101 (2005), `http://www.cs.utsa.edu/~whaley/papers/spercw04.ps`.

[164] R. Clint Whaley, Antoine Petitet, and Jack J. Dongarra, Automated Empirical Optimization of Software and the ATLAS Project, *Parallel Computing* **27**, 3 (2001), Also available as University of Tennessee LAPACK Working Note #147, UT-CS-00-448, 2000 (`www.netlib.org/lapack/lawns/lawn147.ps`).

[165] Shweta Sinha and Manish Parashar, System Sensitive Runtime Management of Adaptive Applications, in *Proceedings of the Tenth IEEE Heterogeneous Computing Workshop*, San Francisco, CA, 2001.

[166] R. Chowdhary, P. Bhandarkar, and M. Parashar, Adaptive QoS Management for Collaboration in Heterogeneous Environments, in *Proceedings of the 16th International Parallel and Distributed Computing Symposium (IEEE, ACM), 11th Heterogeneous Computing Workshop*, Fort Lauderdale, FL, 2002.

[167] Peter J. Keleher, Jeffrey K. Hollingsworth, and Dejan Perkovic, Exploiting Application Alternatives, in *The 19th International Conference on Distributed Computing Systems*, 1999.

[168] Cristian Tapus, I-Hsin Chung, and Jeffrey K. Hollingsworth, Active Harmony: Towards Automated Performance Tuning, in *Proceedings of SC02*, 2002.

[169] Jeffrey S. Vetter and Patrick H. Worley, Asserting Performance Expectations, in *Proceedings of SC02*, 2002.

[170] R. Bramley, D. Gannon, T. Stuckey, J. Villacis, J. Balasubramanian, E. Akman, F. Berg, S. Diwan, and M. Govindaraju, The Linear System Analyzer, in *Enabling Technologies for Computational Science*, Kluwer, 2000.

[171] E. N. Houstis, A. C. Catlin, J. R. Rice, V. S. Verykios, N. Ramakrishnan, and C. E. Houstis, A Knowledge/Database System for Managing Performance Data and Recommending Scientific Software, *ACM Transactions on Mathematical Software* **26**, 227 (2000).

[172] Michael O. McCracken, Allan Snavely, and Allen Malony, Performance Modeling for Dynamic Algorithm Selection, in *Proc. International Conference on Computational Science (ICCS'03), LNCS*, volume 2660, pages 749–758, Berlin, 2003, Springer.

[173] Richard Vuduc, James Demmel, and Jeff Bilmes, Statistical Models for Empirical Search-Based Performance Tuning, *International Journal of High Performance Computing Applications* **18**, 65 (2004).

[174] M. Sosonkina, Runtime adaptation of an iterative linear system solution to distributed environments, in *Applied Parallel Computing, PARA'2000*, volume 1947 of *Lecture Notes in Computer Science*, pages 132–140, Berlin, 2001, Springer-Verlag.

[175] N. Furmento, A. Mayer, S. McGough, S. Newhouse, T. Field, and J. Darlington, Optimisation of component-based applications within a grid environment, in *Proceedings of SC2001*, 2001.

[176] X. Gu and K. Nahrstedt, A scalable QoS-aware service aggregation model for peer-to-peer computing grids, in *Proceedings of HPDC 2002*, 2002.

[177] Noriki Amano and Takuo Watanabe, A Software Model for Flexible and Safe Adaptation of Mobile Code Pr ograms, in *Proceedings of the International Workshop on Principles of Software Evolution*, pages 57–61, Orlando, FL, 2002.

[178] Eric Wohlstadter, Stefan Tai, Thomas Mikalsen, Isabelle Rouvellou, and Premkumar Devanbu, GlueQoS: Middleware to Sweeten Quality-of-Service Policy Interactions, in *Proceedings of the 26th International Conference on Software Engineering (ICSE '04)*, pages 189–199, 2004.

[179] K. Bennett, P. Layzell, D. Budgen, P. Brereton, L. Macaulay, and M. Munro, Service-Based Software: The Future for Flexible Software, in *Proceedings of the 7th Asia-Pacific Software Engineering Conference (APSEC 2000)*, pages 214–221, 2000.

[180] R. Raje, B. Bryant, A. Olson, M. Augoston, and C. Burt, A quality-of-service-based framework for creating distributed heterogeneous software components, *Concurrency Comput: Pract. Exper.* , 1009 (2002).

[181] G. J. Brahnmath, R. R. Raje, A. M. Olson, M. Auguston, B. R. Bryant, and C. C. Burt, A quality of service catalog for software components, in *Proceedings of the Southeastern Software Engineering Conference*. `http://www.ndiatvc.org/SESEC2002/`, 2002.

[182] K. Whisnant, Z. Kalbarczyk, and R. K. Iyer, A Foundation for Adaptive Fault Tolerance in Software, in *Proceedings of the 10th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems*, pages 252–260, 2003.

[183] J. P. Loyall, R. E. Schantz, J. A. Zinky, and D. E. Bakken, Specifying and measuring quality of service in distributed object systems, in *Proceedings of ISORC '98*, 1998.

[184] David Reiner and Tad Pinkerton, A Method for Adaptive Performance Improvement of Operating Systems, in *Proceedings of the 1981 ACM SIGMETRICS Conference on Measurement and Methodology of Computer Systems*, pages 2–10, 1981.

[185] M. S. Feather, S. Fickas, A. van Lamsweerde, and C. Ponsard, Reconciling System Requirements and Runtime Behavior, in *Proceedings of the 9th International Workshop on Software Specifica tion and Design*, pages 50–59, 1998.

[186] Will Schroeder, Ken Martin, and Bill Lorensen, *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*, Kitware, Inc., third edition, 2004.

[187] T. Goodale, G. Allen, G. Lanfermann, J. Masso, E. Seidel, and J. Shalf, The Cactus framework and toolkit: design and applications, in *High Performance Computing for Computational Science - VECPAR 2002. 5th International Conference*, volume 2565 of *Lecture Notes in Computer Science*, pages 197–227, Spinger-Verlag, 2003.

[188] C. Hill, C. DeLuca, V. Balaji, M. Suarez, and A. da Silva, Architecture of the Earth System Modeling Framework, *Computing in Science and Engineering* **6** (2004).

[189] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludascher, and S. Mock, Kepler: an extensible system for design and execution of scientific workflows, in *16th International Conference on Scientific and Statistical Database Management*, pages 423–424, IEEE Comput. Soc., 2004.

[190] E. Wes Bethel (co PI) and Chris Johnson (co PI), Visualization and Analytics for Enabling Technology, SciDAC CET proposal, March 2006.

[191] MATLAB - The Language of Technical Computing, `http://www.mathworks.com/products/matlab/`.

[192] ISO/IEC JTC1/SC22 Committee, Information technology - Programming languages - Fortran - Part 1: Base Language, Technical Report ISO/IEC 1539-1:2004, International Organization for Standardization (ISO), 2004.

[193] Tom Epperly, Preliminary Thoughts on Introducing Structs to SIDL/Babel, Technical Report UCRL-TR-201771, Lawrence Livermore National Laboratory, 2004.

[194] C. E. Rasmussen, M. J. Sottile, S. S. Shende, and A. D. Malony, Bridging the language gap in scientific computing: the Chasm approach, *Concurrency and Computation: Practice and Experience* **18**, 151 (2006).

[195] Chasm: Language Interoperability Tools, `http://chasm-interop.sourceforge.net/`.

[196] TotalView, `http://www.etnus.com/TotalView/`.

[197] Richard M. Stallman, Roland H. Pesch, and Stan Shebs, *Debugging with GDB: The GNU Source-Level Debugger*, Free Software Foundation, ninth edition, 2002.

[198] *Debugging a Program with Dbx*, Sun Microsystems Inc., 2005.

[199] N. Trebon, A. Morris, J. Ray, S. Shende, and A. Malony, Performance Modeling of Component Assemblies with TAU, in *Workshop on Component Models and Frameworks in High Performance Computing*, IEEE, 2005.

[200] TAU's CCA Tools, `http://www.cs.uoregon.edu/research/tau/cca/`.

[201] JUnit, `http://junit.org/`.

[202] PyUnit, `http://pyunit.sourceforge.net/`.

[203] Dave Thomas and David Heinemeier Hansson, *Agile Web Development with Rails*, Pragmatic Bookshelf, 2005.

[204] Gregory R. Watson (PI), Center for Integrated Tools to Enhance Productivity, SciDAC CET proposal, March 2006.

[205] Salman Habib, Theoretical and Observational Studies of Dark Energy: Meeting the Computation and Data Challenge, SciDAC SAP proposal, March 2006.

[206] Phillip Colella, An Algorithmic and Software Framework for Applied Partial Differential Equations (APDEC), `http://davis.lbl.gov/APDEC/`.

[207] CCA Tutorials, `http://www.cca-forum.org/tutorials/`, 2005.

[208] Comprehensive Perl Archive Network, `http://www.cpan.org`, 2006.

[209] Apache Maven Project, `http://maven.apache.org`, 2006.

[210] CCA Tutorial Archives, `http://www.cca-forum.org/tutorials/archives/`, 2005.

[211] Jack Dongarra (PI), Institute for the Support of the SciDAC Software Ecosystems, SciDAC Institute proposal, March 2006.

[212] Oak Ridge National Laboratory, Welcome to SCICOMPFORGE.ORG!, `http://scicompforge.org`, 2006.

[213] Center for Programming Models for Scalable Parallel Computing, `http://pmodels.org`, 2006.

[214] High Productivity Computing Systems (HPCS), `http://www.darpa.mil/ipto/programs/hpcs/`, 2006.

[215] Unified Parallel C at George Washington University, `http://upc.gwu.edu`, 2006.

[216] Tarek El-Ghazawi, William Carlson, Thomas Sterling, and Katherine Yelick, *UPC: Distributed Shared Memory Programming*, John Wiley and Sons, 2005.

[217] Felipe Bertrand, *Data Redistribution and Remote Method Invocation in Parallel Component Architectures*, PhD thesis, Indiana University, 2005.

[218] Sriram Krishnan, *An Architecture for Checkpointing and Migration of Distributed Components on the Grid*, PhD thesis, Indiana University, 2004.

[219] Kenneth Chiu, *An Architecture for Concurrent, Peer-to-Peer Components*, PhD thesis, Indiana University, 2001.

[220] Madhusudhan Govindaraju, *An Open Framework Code Generation Toolkit for Distributed Systems based on XML-Schemas*, PhD thesis, Indiana University, 2002.

[221] The DOE ACTS Collection, `http://acts.nersc.gov`, 2006.

[222] Open Source Initiative (OSI), `http://www.opensource.org/`, 2006.

[223] The Apache Software Foundation, `http://apache.org/foundation/`, 2006.

[224] About the Mozilla Foundation, `http://www.mozilla.org/foundation/`, 2006.

[225] Red Hat, Inc., `http://www.redhat.com`, 2006.

[226] Wei Lu and Kenneth Chiu and Aleksander Slominski and Dennis Gannon, A streaming validation model for SOAP digital signature, *14th IEEE International Symposium on High Performance Distributed Computing HPDC-14* , 243 (2005).

[227] Felipe Bertrand, Kenneth Chiu, and Randall Bramley, An Approach to Parallel MxN Communication, in *International Journal of High Performance Computing Applications*, volume 19, pages 339–407, 2005.

3. Glossary of Acronyms

ANL Argonne National Laboratory

APDEC Algorithmic and Software Framework for Applied Partial Differential Equations

BU Binghamton University

CBSE component-based software engineering

CCA Common Component Architecture

CCSM Community Climate System Model

CCTTSS Center for Component Technology for Terascale Simulation Software

CFRFS Computational Facility for Reacting Flow Science

CPAN Comprehensive Perl Archive Network

CPES Center for Plasma Edge Simulations

CQoS computational quality of service

DARPA Defense Advanced Research Projects Agency

DoD Dept. of Defense

DOE Department of Energy

ESMF Earth System Modeling Framework

FASTOS Forum to Address Scalable Technology for Operating Systems and Runtimes

FPGA Field-Programmable Gate Array

FSP Fusion Simulation Project

GUI Graphical User Interface

HPC high-performance computing

HPCS High-Productivity Computing Systems

IDE Integrated Development Environment

ICS Institute for Coupling High-Performance Simulations

ISSSE Institute for the Support of the SciDAC Software Ecosystem

ITAPS Interoperable Technologies for Advanced Petascale Simulations

IU Indiana University

LANL Los Alamos National Laboratory

LBNL Lawrence Berkeley National Laboratory

LLNL Lawrence Livermore National Laboratory

UMD University of Maryland

MCMD multiple-component multiple-data

MCT Model Coupling Toolkit

MLP Multi-Level Parallelism

MPMD multiple-program multiple-data

NASA National Aeronautics and Space Administration

NCAR National Center for Atmospheric Research

NLCF National Leadership Computing Facility at ORNL

NIH National Institutes of Health

NSF National Science Foundation

ORNL Oak Ridge National Laboratory

OSI Open Source Initiative

PCI Parallel Coupling Infrastructure

PNNL Pacific Northwest National Laboratory

PTP Parallel Tools Project

PERC Performance Engineering Research Center

RMI remote method invocation

SAP Scientific Application Pilot

SBIR Small Business Innovation Research

SciDAC Scientific Discovery through Advanced Computing

SDM Scientific Data Management

SIDL Scientific Interface Definition Language

SNL Sandia National Laboratories

SPMD single-program multiple-data

SWIM Center for Simulation of Wave Interactions with Magnetohydrodynamics

TAO Toolkit for Advanced Optimization

TASCS Technology for Advanced Scientific Component Software

Tech-X Tech-X Corporation

TOPS Terascale Optimal PDE Solvers

TSTT Terascale Software Tools and Technologies

UPC Unified Parallel C

UU University of Utah

VACET Visualization and Analytics Center for Enabling Technology

VTK Visualization Tool Kit

A. Revised Scope of Work

Important Note

The TASCS proposal was funded at 25% less than the requested budget. This section briefly describes how the scope of work was modified during negotiation of the award to compensate for the reduced budget. This information applies to the initial award, which began in July 2006. For further updates, please refer to the latest TASCS Management Plan*, available from the Lead PI.*

In reducing our scope of work, we have had to pull back significantly in all aspects of the project. In doing so, we have tried to place a stronger emphasis on the most critical "user-facing" activities, and reducing our targets for the Component Technology Initiatives to compensate. Many of the reductions have been accomplished by shifting selected deliverables to future proposals and spreading out the remaining work over a longer period.

More detailed comments follow, keyed to the relevant sections of the original proposal.

Component Technology Initiatives (Sec. 1.5, pg. 8)

Parallel Coupling Infrastructure (Sec. 1.5.1, pg. 8) Because of uncertainties in the funding of partners in this highly collaborative initiative, and concerns about maintaining a critial mass with the reduced funding, we have decided to eliminate this Initiative. The remaining effort that had been allocated to this effort will be shift to enhancing the CCA Toolkit. This will include interfaces and components for parallel coupling and other types of components based on how our various collaborators fare in the funding process.

Support for Emerging HPC Hardware and Software Paradigms (Sec. 1.5.2, pg. 10) More speculative aspects of this work have been dropped, and the FT-related activities have been significantly reduced.

Software Quality and Verification (Sec. 1.5.3, pp. 11) More speculative aspects of this work have been dropped in order to focus on the connections with the CQoS Initiative.

Computational Quality of Service (Sec. 1.5.4, pg. 12) This will be scaled back, in part, by working with fewer external collaborators and reducing the breadth of topics within the general area of quality of service adaptation.

CCA Environment (Sec. 1.6, pg. 13)

Core Tools and Usability (including CCA Lite) (Sec. 1.6.1 and 1.6.3, pg. 14 and 15) These two areas will suffer less of a reduction than other aspects of the proposal. Deliverables have mostly been stretched out.

Enhancements (Sec. 1.6.2, pg. 14) Most of our reductions in this area were driven by the idea of focusing more narrowly on the needs of our core DOE user base, eliminating tasks that would have been quite useful to broader HPC computational science community. Examples include Matlab bindings for Babel, interoperability between CCA frameworks and environments like ESMF, VTK, and Cactus. Interoperability with the Kepler workflow environment, which will

be done in collaboration with the SDM CET, remains in our plans. Other activities have been stretched out over longer periods.

CCA Toolkit (Sec. 1.7, pg. 16) Like the Core Tools and Usability activities, the Toolkit is vitally important to the user experience of the CCA. Therefore this area has also been cut less, and then bolstered by the effort shifted from the PCI Initiative.

User Outreach and Applications Support (Sec. 1.8, pg. 17) Consistent with earlier guidance that we should rely on SAPs and application funding as our primary means of interacting with applications, we have allowed this area to be reduced. This means we will have less manpower available to help those applications with which we do not have direct, funded connections.

The following tables update those in Sec. 1.11 (pg. 20).

Table A.1: **Summary of Milestones for Component Technology Initiatives**
Coordinator: L.C. McInnes, ANL; *Participants:* ANL, IU, LLNL, ORNL, PNNL, SNL, UMD

Year 1	*Year 2*	*Year 3*	*Years 4–5*
Emerging HPC Paradigms *Coordinator:* J. Nieplocha, PNNL; *Participants:* ORNL, PNNL *Motivating Applications:* biology and quantum chemistry simulations			
• Develop multi-level parallelism model. • Define abstract model for CCA hybrid apps.	• Develop CCA model for processor groups. • Develop component interface for hybrid systems.	• Develop simple MCMD programming model. • Prototype hybrid interface for example application.	• Incorporate MCMD support for heterogeneous prog. models. • Implement hybrid & MCMD example application components.
Software Quality and Verification *Coordinator:* T.L. Dahlgren, LLNL; *Participants:* LLNL, ORNL *Motivating Applications:* fusion simulations, CQoS initiative			
• Identify and define CQoS and domain-specific semantics; assess spec. mechanisms.	• Develop semantics prototype(s). • Design method invocation sequencing constraints enforcement.	• Introduce semantic specifications into selected Toolkit components. • Develop sequencing enforcement prototype in Babel/SIDL.	• Evaluate semantics prototype(s). • Evaluate sequencing enforcement prototype. • Revise and evaluate prototypes based on CQoS evolution.
Computational Quality of Service (CQoS) *Coordinator:* L.C. McInnes, ANL; *Participants:* ANL, SNL *Motivating Applications:* combustion, quantum chemistry, accelerator, and fusion simulations			
• Collect application requirements, define metrics, perform base experiments. • Build database component.	• Populate CQoS testbed and specify initial CQoS API. • Develop initial performance models for applications. • Develop proxy port generation for CQoS usage.	• Complete design of overall CQoS strategy. • Implement application control laws. • Implement an asynchronous control infrastructure.	• Design APIs for general analysis engines. • Create a generic CQoS framework for HPC applications. • Stress test CQoS tools.

Table A.2: **Summary of Milestones for CCA Environment**
Coordinator: G. Kumfert, LLNL; *Participants:* ANL, BU, LANL, LLNL, ORNL, SNL, UU

Year 1	*Year 2*	*Year 3*	*Years 4–5*
Core Tool Support and Maintenance *Coordinator:* B. Allan, SNL; *Participants:* ANL, LLNL, ORNL, SNL			
⟵ Support helpdesk and open bugtracking. ⟶			
⟵ Develop and maintain technical documentation. ⟶			
• Port CCA software stack to NLCF machines	• Complete CCA Conformance Tests	• Automated conformance testing for all CCA frameworks.	• Evaluate and port to new architectures as they emerge.
Enhancements *Coordinator:* T. Epperly, LLNL; *Participants:* BU, LLNL, ORNL, SNL, UU			
• Adopt EventService and MPIService into standard. • Demonstrate support for BabelRMI in XCAT.	• Demonstrate CCA/Kepler interoperability. • Add structs to SIDL/Babel. • Add Fortran 2003 support to Babel. • Incorporate SOAP as module in BabelRMI, integrate with Proteus/ XCAT.	• Finalize specification for Component sub-assemblies. • Develop specification for framework interoperability. • Release full fledged version of XCAT.	• Demonstrate exchange of sub-assemblies between two CCA Frameworks. • Demonstrate framework interoperability between CCA implementations. • Extend BabelRMI communication modules for new CCA applications.
Usability *Coordinator:* C. Rasmussen, LANL; *Participants:* LANL, LLNL, ORNL, SNL			
• Draft CCA-Lite Spec and CCA-Lite Framework. • Document advanced component debugging techniques. • Design component test harness.	• Preliminary integration of CCA-Lite test framework with Ccaffeine framework. • Deploy component test harness.	• Demonstrate connecting CCA-Lite components to CCA components in Ccaffeine. • Integrate SIDL semantics enforcement into testing methodology. • Develop component tracing tools to facilitate debugging.	• Demonstrate source-to-source conversion of CCA-Lite component to full CCA Component. • Evaluate tradeoffs in debugging and testing CCA-Lite vs. full CCA. • Apply test harness to selected toolkit components.

Table A.3: **Summary of Milestones for the CCA Toolkit**
Coordinator: R. Armstrong, SNL; *Participants:* ANL, IU, LLNL, ORNL, PNNL, SNL, Tech-X, UMD, UU

Year 1	*Year 2*	*Year 3*	*Years 4–5*
⟵ Design, establish and, based on user feedback, iterate and improve CCA Base Installation ⟶			
• Design toolkit structure and contribute initial components to the Toolkit, and establish web distribution system	• Incorporate and promulgate Toolkit components into CCA tutorial and outreach activities, improve type and quality of the Toolkit repertoire.	• Add to Toolkit component improvements to CCA architecture since Year 1, e.g. MCMD components, templates, and CQoS plug-ins.	• Establish web-based/community process for approving/distributing component contributions from the community, as user base for Toolkit expands.

Table A.4: **Summary of Milestones for Application and User Outreach and Support**
Coordinator: D.E. Bernholdt, ORNL; *Participants:* ANL, IU, LLNL, ORNL, PNNL, SNL, Tech-X

Year 1	*Year 2*	*Year 3*	*Years 4–5*
⟵ Support applications in adopting and using CCA. ⟶			
⟵ Deliver user support, incl. tutorials, coding camps, etc. ⟶			
⟵ Update tutorial and best practices documentation. ⟶			
• Revamp `cca-forum.org` web services.	• Revamp or migrate `cca-forum.org` code development services.		