Center for Component Technology for Terascale Simulation Software (CCTTSS)

http://www.cca-forum.org/ccttss
PI: Rob Armstrong (rob@ca.sandia.gov)

Executive Summary
September 2001

Vision:

While the complexity of scientific simulations increasingly necessitates the combined use of software developed by different groups, this has been difficult for many of DOE's sophisticated packages due to differences in implementation languages, programming styles, and calling interfaces. Initial software interoperability work includes direct interfacing between tools with complementary functionalities (for example, between a library that provides finite element discretizations of PDEs and another library that provides linear solvers). However, because one-to-one interfacing requires writing new translations of interfaces and data for each tool that provides a particular functionality, this approach is excessively labor intensive and does not facilitate the incorporation of new tools. Thus, the CCTTSS advocates using a dynamic component-based approach, which clearly separates (1) community-defined domain-specific interfaces and (2) underlying implementations that may be developed by different groups.

Component technology represents an important new tool for software development. Unfortunately, commodity component models that are widely used in industry—such as CORBA, DCOM, and Enterprise JavaBeans—do not address parallelism and other needs of high-performance scientific software. The Common Component Architecture (CCA) component approach specifically targets the needs of large-scale, complex, high-performance, scientific simulations. We have demonstrated the basic principles of such a system. The CCTTSS focuses on taking the CCA from a conceptual prototype to a full-fledged, high-performance scientific component architecture. The technology developed within the Center will provide the following benefits:

- Leverage and exploit DOE's legacy software investment by removing barriers to software re-use, thereby enabling collaborative software development rather than individual efforts.
- Accelerate software development and reduce costs; computational scientists will develop applications through the integration of existing community components rather than build monolithic applications from scratch.
- Enable the integration of high-performance simulation codes with desktop industry software through component software bridges, thereby avoiding the re-creation of existing technology.

Major Goals and Technical Challenges:

The goal of the CCTTSS is to develop component technology for high-performance parallel scientific computing that enables the interoperability and re-use of DOE simulation software. Components allow software developers to describe the calling interfaces of libraries and applications in a manner that hides low-level details, such as implementation language, compiler, parallelism, or location on a network. Developers therefore do not need to be concerned with questions such as "Can my Fortran program call this C++ solver library that was parallelized with MPI?" Components encapsulate the knowledge, experience, and work of other scientists, and they provide building blocks that speed application development.

Our work targets research, development, and deployment activities in four areas:

- *We will develop a single, integrated reference implementation of a component architecture* that addresses the unique needs of high-performance massively parallel scientific simulations, including both single-component-multiple-data (SCMD) and distributed environments. This work will include specifications and support services to enable the creation of high-performance parallel applications from components.
- *We will develop a suite of scientific components* and will make this software available to the community through a web-based component repository. In particular, we will modify existing parallel libraries—for example, linear and nonlinear solvers, mesh management, optimization, fault tolerance, steering, and visualization—to use component technology and will collaborate with other teams to develop additional components, including those for scientific data associated with structured and unstructured meshes.

- *We will specify and implement technology for parallel data redistribution*, as necessary to support parallel data exchange for scientific model coupling. Our interfaces will generalize existing parallel data collection and coupling systems and will build on the CCA scientific data component interfaces.
- *We will promote and support the integration of CCA technology into SciDAC applications and general user software*. In addition to pursuing a general program of education and outreach to prospective users, we will work closely with adopters of CCA to provide a tight feedback loop between users and developers. Special efforts are planned within the Center that focus on the chemistry and climate modeling application domains as early adopters and sources of demonstrations.

New capabilities that may be of particular interest to applications scientists include:

- *Tools for language interoperability*, supporting seamless language calls across Fortran 77, Fortran 90, C, C++, Java, and Python;
- *Common deployment methods for all CCA-compliant software* through a web-based repository;
- *Increased interoperability of libraries* such as math solvers via plug-and-play components, thereby enabling users to experiment with different solver approaches; and
- *Support for parallel data redistribution between parallel components*, useful for model coupling, visualization, load balancing, and other problems that entail moving data between two parallel simulations running on differing numbers of processors.

Major Milestones and Activities:

- Year 1:
  - Assess requirements of other SciDAC projects; promote and support the CCA component approach through tutorials, workshops, etc. (ongoing throughout all years).
  - Define interfaces and develop various base component prototypes (low-level services, scientific data, solvers, visualization, parallel data exchange, GUI builder, etc.).
  - Complete candidate SCMD framework specification and implementation; complete candidate distributed computing CCA runtime environment.
- Year 2:
  - Deploy base component prototypes; integrate base component implementations; define and implement second-level component extensions; propagate language interoperability.
  - Define applications interfaces; begin production componentization for chemistry and climate models.
- Year 3:
  - Deploy integrated SCMD/distributed framework to applications teams; deploy component repository.
  - Streamline component and framework implementations; implement fault tolerance and recovery mechanisms; generalize and standardize system specifications where appropriate.
- Years 4 and 5:
  - Continue research, development, and outreach activities; release final version of integrated framework.

Current Connections with Other SciDAC Projects:

CCTTSS collaborations with other SciDAC projects include the following:

- *Advanced Computing for 21st Century Accelerator Science and Technology* (PIs: K. Ko and R. Ryne; CCTTSS liaison: C. Rasmussen) – Explore using CCA technology to assist with software integration and language interoperability.

- *Advanced Software for the Calculation of Thermochemistry, Kinetics, and Dynamics* (PI: A. Wagner; CCTTSS liaison: D. Bernholdt) – Create component-based software development methodologies and define interfaces that facilitate the coupling of electronic structure and kinetics software.

- *Collaborative Design and Development of the Community Climate System Model for Terascale Computers* (PI: R. Malone; CCTTSS liaison: D. Bernholdt) – Develop component-based software design methodologies and integrate the CCA with current componentization efforts within the climate community.

- *Collaboratory for Multi-Scale Chemical Science* (PI: L. Rahn; CCTTSS liaison: R. Armstrong) – Use CCA technology in analysis tools being developed as a part of the collaboratory (in conjunction with the projects *A Computational Facility for Reacting Flow Science* and *Terascale High-Fldelity Simulations of Turbulent Combustion with Detailed Chemistry).*

- *A Computational Facility for Reacting Flow Science* (PI: H. Najm; CCTTSS liaison: J. Ray) – Develop components for flame simulations (e.g., time integration, chemical mechanism reduction, and adaptive meshing).

- *Middleware Technology to Support Science Portals* (PI and CCTTSS liaison: D. Gannon) – Use CCA technology in the design of grid science portals to support synchronous and asynchronous collaboration.

- *Scalable Systems Software ISIC* (PI: A. Geist; CCTTSS liaison: J. Kohl) – Integrate CCA software with scalable systems tools to allow CCA-based applications to be launched, monitored, and managed efficiently on terascale computational systems.

- *Terascale High-Fldelity Simulations of Turbulent Combustion with Detailed Chemistry* (PI: A. Trouve; CCTTSS liaison: J. Ray) – Leverage components developed in the project *A Computational Facility for Reacting Flow Science* (e.g., adaptive meshing, computation of chemical rates and transport properties).

- *Terascale Optimal PDE Simulations ISIC* (PI: D. Keyes; CCTTSS liaison: L. McInnes) – Create suites of interoperable solvers and define common interfaces for PDE, eigenanalysis, and optimization components.

- *Terascale Simulation Tools and Technologies ISIC* (PI: J. Glimm; CCTTSS liaison: L. Freitag) – Create interoperable meshing and discretization technologies and define common interfaces for scientific data components.