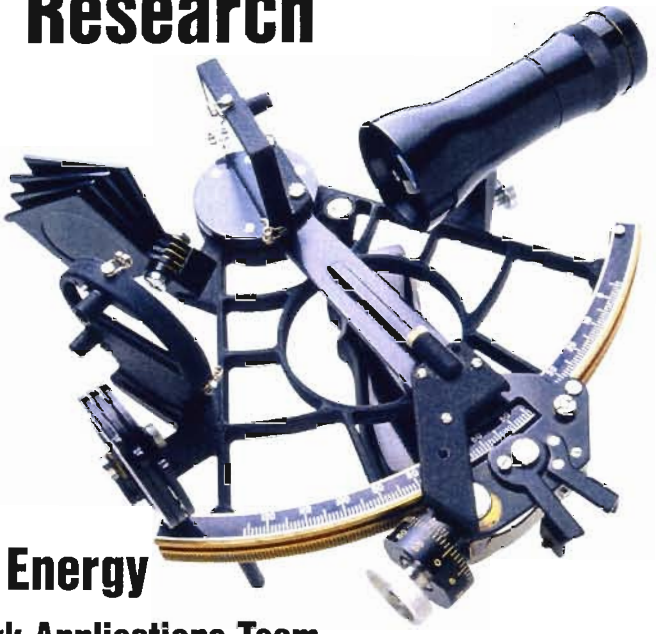
PNNL-SA-32668

Collaborative Problem-Solving Environments

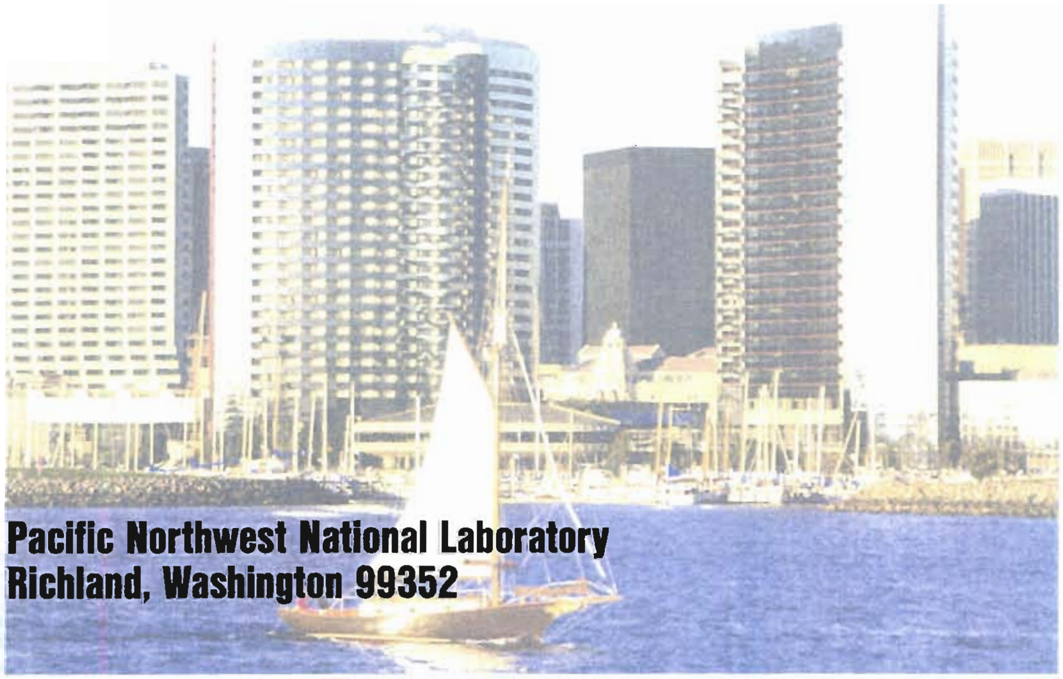
Proceedings for the Workshop on CPSEs for Scientific Research

San Diego, California
June 29 to July 1, 1999



Sponsored by the
U.S. Department of Energy
High Performance Network Applications Team
of the Large-Scale Networking Working Group

Contract No. DE-AC06-76RLO 1830



Pacific Northwest National Laboratory
Richland, Washington 99352

Summary

A workshop on collaborative problem-solving environments (CPSEs) was held June 29 through July 1, 1999, in San Diego, California. The workshop was sponsored by the U.S. Department of Energy (DOE) and the High Performance Network Applications Team of the Large Scale Networking Working Group, and attracted the participation of approximately fifty attendees with diverse professional backgrounds. Workshop attendees came from universities (e.g., University of Washington, University of Michigan, CMU, USC), companies (e.g., HP, SGI, IBM), research organizations (e.g., San Diego Supercomputing Center, Scripps Research Institute, NCSA), DOE national laboratories (e.g., PNNL, ORNL, ANL, SNL), and other federal research laboratories (e.g., NOAA, NIH, NASA).

The motivation for the workshop was to bring together researchers and developers from industry, academia, and government to identify, define, and discuss future directions in collaboration and problem-solving technologies in support of scientific research. After a decade of cultivation, research and development in collaboration environments and integration frameworks have produced theories and technologies that provide basic levels of support and functionality. For this workshop, we sought to step beyond the present foundation of collaborative problem-solving technology and capabilities to project into the future—evolving current research and technology towards new theories, designs, and architectures that would meet the needs of modern and future scientific work.

During the workshop, six technical presentations were given by a group of professionals from industry, academia, and DOE national research laboratories. The presentations were designed to provide attendees exposure and insight into the diverse and ground-breaking research and development efforts occurring within the CPSE community. Topics of presentation ranged from the development of collaborative and distributed architectures to the study of augmented and virtual reality environments. Brimming with newfound knowledge and motivation from the technical presentations, workshop attendees participated in a series of technical breakout sessions where general CPSE topics were further dissected and explored. Topics under discussion covered technical issues in distributed resource management, data management, scientific collaboration and research, design strategies, and future system architectures for CPSEs.

One of the goals of the workshop was to gather and establish a community of CPSE researchers and developers inside and outside the DOE complex. We hope to continue this workshop on an annual basis, where participants may use the workshop as a forum to regularly discuss and extend CPSE ideas, concepts, and technologies.

PACIFIC NORTHWEST NATIONAL LABORATORY
operated by
BATTELLE
for the
UNITED STATES DEPARTMENT OF ENERGY
under Contract DE-AC06-76RLO 1830

Contents

Introduction

Collaboration environments have largely evolved from the field of computer-supported cooperative work (CSCW). To a certain extent, CSCW studies have focused on the analysis, development, and use of low-level communication mechanisms (e.g., text, audio, video), and concepts and issues surrounding the context of collaboration (e.g., user presence, multi-user dimensions (MUDs), 3D virtual worlds). In contrast, problem-solving environments (PSEs) have often been associated with integration frameworks and component-based architectures. Such frameworks and architectures link computational resources at the level of the computer system and network. From these groundwork efforts and results in CSCW and system architectures, a logical and pragmatic direction for future research and development would be to effectively bridge the gulf between theory and practice by bringing these concepts and technologies closer to the actual collaborative research that scientists conduct. Collaborative scientific research is largely a process of collaborative problem-solving. It involves "higher-order" collaboration beyond the simple ability to communicate and "higher-order" analyses and processes beyond the simple ability to access and link computational resources.

Collaborative problem-solving is a complex, multi-faceted concept. Gallopoulos, Houstis, and Rice define a PSE as a

> computer system that provides all the computational facilities necessary to solve a target class of problems. These features include advanced solution methods, automatic or semiautomatic selection of solution methods, and ways to easily incorporate novel solution methods. Moreover, PSEs use the language of the target class of problems, so users can run them without specialized knowledge of the underlying computer hardware or software.

This seemingly simple view of PSEs encapsulates many powerful concepts and technical challenges. At the lowest level, PSEs encompass general concepts and capabilities in a variety of technical areas includeing data management, resource management, security, distributed computing, and real-time data sources and instrumentation. This level focuses mainly on supporting low-level computer-based objects of computation and communications. At a higher-level, integration frameworks and component-based architectures are emphasized to manage and link primitive problem-solving resources, data, and tools. This level focuses on providing infrastructure to manage and support low-level objects and capabilities, and to make these objects and capabilities available to different domains and scientific problems. At the highest level, problem-solving tools and capabilities are geared to support specific scientific research functions in areas such as scientific workflow, scientific data and record management, and experiment and model design. This level focuses on providing high-level capabilities and tools to directly support the scientific work of domain scientists. Overall, Collaborative Problem-Solving Environments (CPSEs) promote the convergence of a vast landscape of technical ideas and implementations.

CPSEs also promote convergence among disciplines and people. To say the least, the involvement of domain scientists is crucial to the development of CPSEs. Domain scientists represent and seize the domain knowledge that is at the very center of collaborative problem-solving. Conversely, computer scientists possess the understanding of computer technology and its potential applications. They function as the technologists that bring computer technology to bear, assist, and improve the capabilities of the domain scientists. Yet, with respect to the development of problem-solving tools and environments, domain and computer scientists often work in isolation. Domain scientists tend to construct specific, hardwired applications and tools to support their specific fields and needs. Meanwhile, computer scientists tend to build general system architectures that may fail to address the needs of any scientific field. One of the main objectives of the CPSE workshop was to provide a forum in which domain and computer scientists may share their perspectives, ideas, and requirements. Through its purpose, agenda, and participants, the workshop aimed to improve the level of interaction among different CPSE stakeholders and the quality and capabilities of the CPSEs that stakeholders create.

Given the basic challenges for CPSEs described above, the workshop targeted exploration and discussion on five fundamental research areas in the development of CPSEs. The five areas are as follows:

- Distributed resource management—CPSEs manage various kinds of scientific, computational, and people resources. What are useful strategies and models for managing such diverse resources?
- Data management—Scientists and computers work with exorbitant amounts of data. What kinds of data are relevant to CPSEs and how should they be supported?
- Scientific collaboration and research—CPSEs support collaboration and research processes of scientists. What are important collaborative and problem-solving activities for scientists, and how should they be captured?
- CPSE design strategies—CPSE development require substantial commitment and participation among domain and computer scientists. What are relevant design strategies that will facilitate effective CPSE development for both domain and computer scientists?
- Future CPSE architectures—Collaboration and problem-solving capabilities in CPSEs are generally provided through some form of an integration platform or framework. What are the attributes of an effective CPSE architecture that will support the current and future collaborative and problem-solving needs of scientists?

These research areas were investigated with vigor during the workshop. Beyond this workshop, the above topics represent important research directions for the CPSE community to continue to explore and to evolve.

The workshop may be seen as a catalyst for the establishment of a community of researchers and developers interested in computer-supported collaborative problem-solving. Our aspirations for the workshop exist at different levels. For individual attendees, we hope that each will become an active contributor and motivator in the emerging field of CPSEs, and that the contacts and relationships each attendee developed during the workshop will expand and persist over time. For the field of CPSEs, we hope that the workshop was a seminal event for CPSEs—setting direction and building momentum for CPSE research and development for years and decades to come.

Technical Presentation Abstracts

Introduction

Six technical presentations were given during the CPSE workshop. Technical speakers came from different sectors effectively representing industry, academia, and the DOE national research laboratories. Overall, the topics of presentation provided a diverse cross-section of experiences and ideas in CPSEs.

From the domain science perspective, Dr. Farnam Jahanian of the University of Michigan and Dr. Arthur Olson of the Scripps Research Institute presented computing environments that supported specific scientific domains. Dr. Jahanian discussed the Space Physics and Aeronomy Research Collaboratory (SPARC), which allows atmospheric and space physics scientists to collaborate on experiments via online collaboration tools. Dr. Olson discussed the application of specific integration platforms and tools, such as the Advanced Visualization System (AVS) and Python, in the development of new computational methods, visualizations, and data access mechanisms for structural molecular biology. These two presentations provided a strong view of problem solving and CPSEs from the perspective of the domain scientist.

Dr. William Johnston and Dr. Edward Chow of National Aeronautics and Space Administration (NASA) described general computing architectures for problem solving. Dr. Johnston presented NASA's Information Power Grid (IPG), which is a distributed computing and data management environment intended to provide NASA scientists and engineers with basic engineering, research, and development capabilities. Dr. Chow described another NASA system known as the Intelligent Synthesis Environment (ISE). ISE is a collaborative engineering system that integrates the functions of mission planning, engineering design, manufacturing, and operations. Both presentations centered on the development of general architectures that may be applied across different domains and scientific problems.

Dr. Shahrokh Daijavad of IBM's T.J. Watson Research Center presented yet another context for collaboration and problem solving. Although the focus of most of the technical presentations was on supporting engineering and scientific work, Dr. Daijavad described a web-based collaboration environment for conducting virtual meetings. In his presentation, Dr. Daijavad emphasized the analysis and design strategies and processes that his project successfully employed in developing collaboration environments.

Finally, Dr. Thomas Furness of the University of Washington provided a futuristic look at collaboration and problem solving by exploring the potential applications of collaborative 3D virtual environments and media. Through a virtual environment known as GreenSpace and other 3D collaboration technologies, Dr. Furness painted an intriguing picture of how immersive and augmented environments may forever change the way people interact and work together over computers. Dr. Furness' presentation encourages us to look beyond conventional paradigms and technologies to discover more novel forms of collaboration and problem solving.

Collaborative Environments for Team Science: The Space Physics and Aeronomy Research Collaboratory

Farnam Jahanian
University of Michigan
Department of EECS
Ann Arbor, MI

Abstract

The Space Physics and Aeronomy Research Collaboratory (SPARC) project is a multi-institution multidisciplinary effort whose focus is to create an experimental test-bed for scientific collaborative research work over the Internet. SPARC brings together researchers in upper atmospheric and space physics from around the world, providing them a set of online collaboration tools and workspaces that link together scientific instruments, archived data sets, and theoretical models. The collaboratory is itself a subject of study by computer and behavioral scientists who are developing and refining the tools and organizational structures that will make such real-time, online collaborative research commonplace.

SPARC operates in two modes: real-time campaign and electronic workshop. A real-time campaign is a concerted effort to simultaneously collect real-time data from instruments around the world. In this mode, a scientist is likely to be interested in unfolding phenomena and communicate with other scientists on operational planning or predictive guesses on the state of the system. An electronic workshop is a scheduled cyber-gathering with specific goals in mind. Electronic workshops allow scientists to plan research, analyze archived data, and manipulate and understand complex computational models. This mode relies on access to archived data or preprocessed data presented by participating scientists, as well as shared access to multimedia tools. As an important byproduct of this system, SPARC provides outreach and educational products at appropriate pedagogical levels, providing the general public exciting and fresh information on space science. SPARC supports both synchronous and asynchronous work and the transitions made between them. Its design also provides dynamic adaptation to heterogeneity in available resources, administrative domains and individual preferences, as well as high variability in communication and computational resources available to users.

This talk presents an overview of the SPARC project. It also discusses lessons learned from the design, development, and deployment of SPARC, and the evolution of scientific collaboratories during the last several years.

Grids as Production Computing Environments: The Design and Implementation of NASA's Information Power Grid

William E. Johnston, Dennis Gannon[(1)], and Bill Nitzberg[(2)]
Numerical Aerospace Simulation Division
NASA Ames Research Center
Moffett Field, CA

Abstract

Information Power Grid (IPG) is the name of NASA's project to build a fully distributed computing and data management environment—a Grid. The IPG project has near-, medium-, and long-term goals that represent a continuum of engineering, development, and research topics. The overall goal is to provide the NASA scientific and engineering communities a substantial increase in their ability to solve problems that depend on, or could benefit from, the use of large-scale and/or dispersed resources: aggregated computing, diverse data archives, remote laboratory instruments and engineering test facilities, and dispersed human collaborators.

The approach involves defining and building an infrastructure and services that can locate, aggregate, integrate, and manage resources from across the NASA enterprise. An important aspect of IPG is to produce a common view of these resources, and at the same time provide for distributed management and local control.

The primary elements of IPG consist of

- facilities for constructing collaborative, application-oriented workbenches/problem-solving environments across the NASA enterprise based on the IPG infrastructure and applications; constituting the primary science and engineering interface to Grids
- independent, but consistent, tools and services that support various programming environments for building applications in widely distributed environments
- tools, services, and an infrastructure for managing and aggregating dynamic, widely distributed collections of resources—CPUs, data storage/information systems, communications systems, real-time data sources and instruments, and human collaborators
- a common resource management approach that addresses system management, user identification, resource allocations, accounting, security, etc., and at the same time provides for local control of resources
- an operational Grid environment incorporating major computing and data resources at multiple NASA sites to provide an infrastructure capable of routinely addressing larger scale, more diverse, and more transient problems than is possible today.

In this talk we will describe our progress in building IPG and the issues that have surfaced during the project.

[1] gannon@cs.indiana.edu
[2] nitzberg@nas.nasa.gov (MRJ Technology Solutions, NASA contract NAS2-14303)

Collaborative Engineering Environment Infrastructure for the NASA Intelligent Synthesis Environment Program

Edward Chow
Jet Propulsion Laboratory, NASA
Pasadena, CA

Abstract

NASA is in the process of developing its next generation engineering system under the Intelligent Synthesis Environment (ISE) program. The ISE program will provide an environment where science, mission planning, engineering design, manufacturing, and operations are integrated into a single virtual platform available to NASA team members and partners anywhere in the world. The Collaborative Engineering Environment (CEE) is one of the five elements in the ISE program. This presentation will give an overview of the NASA ISE program and discuss specifics of the collaborative infrastructure that is being set up by the ISE/CEE.

GreenSpace: Discovery through Shared Knowledge Networks

Thomas A. Furness III
Human Interface Technology Laboratory
University of Washington
Seattle, WA

Abstract

We anticipate that pervasive problems in our civilization, such as population growth, hunger, global environment, and growing consumption of energy resources, will be solved ultimately by our ability to deal with complexity. Complexity not only results from the great number of factors that are present, but also because the people who will solve these problems have a diversity of cultures, native languages, time zones, professional backgrounds, and life experiences based on their own personal journey on earth. Richness is in this diversity if only we could bring minds together in a more robust way.

As a potential new medium for collaboration, we envision the concept of a virtual common (which we term GreenSpace) where minds can be brought together through 3D interaction in multiple sensory and psychomotor modalities. The GreenSpace would contain a repertory of tools that allow participants to excavate and exploit data resources and discover new deposits of insight. This concept would allow users to solve pervasive problems that collectively confront the nations of the world. Dr. Furness will discuss experiences in building and exploring virtual environments for group knowledge acquisition and discovery, problem solving, design, and educational delivery. These GreenSpaces take advantage of the richness of diversity, while at the same time provide unique tools to deliver bandwidth to the brain and between brains.

Integrating Tools for Structural Molecular Biology—an Execution-Centric Approach

Arthur Olson and Michel Sanner
The Scripps Research Institute
La Jolla, CA

Abstract

We will describe the work done in our laboratory to facilitate computational research in bridging scales in structural biology from atoms to cells. This goal necessitates a computational environment that enables rapid tool prototyping, methods development, and integration. For over 10 years we have used Advanced Visualization System (AVS), a viewer-centric, data-flow programming environment, to prototype new visualization methodologies. While we have made significant progress with this approach, it has some serious limitations. Over the past two years we have explored the use of Python, an object-oriented scripting language, to go beyond the viewer-centric paradigm, to a more flexible execution-centric approach to computational interoperability.

This talk will describe some of the computational challenges embodied in the underlying science, and focus on our current approaches to overcome these challenges by using Python as a "glue-layer" to tie computational methods, data access, and visualization together. We discuss the rationale for our choice of Python, extending Python, integration of methods with Python, and some specific examples of applications of this approach to problems in structural molecular biology.

Virtual Meetings or "Meetings at the Desktop"

Shahrokh Daijavad
IBM T.J. Watson Research Center
Hawthorne, NY

Abstract

In this talk, I will briefly describe a recently proposed project by IBM Research and Boeing to co-develop a set of web-based tools and applications to support intra- and inter-company types of virtual meetings. The proposal expands on the work that the two groups of researchers in IBM and Boeing have done in the past few years on virtual meetings and synchronous collaboration technologies.

Although collaborative solving environments for scientific research bring unique characteristics to the table, we strongly believe that many of the issues considered for asynchronous and synchronous inter- or intra-company collaborative planning or brainstorming sessions apply equally well to scientific collaborations. We will describe a methodology that starts by observing and studying real meetings, followed by identifying a set of requirements for the tools and applications, and finally by using an iterative method to develop and study these tools. We are particularly interested in combining asynchronous and synchronous collaboration environments, in electronic notetaking during meetings and in video capture and indexing to allow replay of multimedia "minutes" associated with meetings.

Breakout Sessions

Introduction

To support integrated collaborative problem-solving, we need a new genre of theories, strategies, and tools layered on top of the current foundation of CSCW and integration framework research and work. At a fundamental level, we need to better comprehend how we facilitate the collaborative problem-solving process with computers, and what strategies to follow to identify viable computer-mediated solutions. To address these points, we identified five topics to be discussed in a two-day, two-pronged approach. On the first day, workshop participants broke into three groups to discuss distributed resource management, data management, and scientific collaboration and research as these topics relate to CPSEs. These three topics were considered key technical concepts or areas for CPSEs. On the second day, workshop participants separated into two groups to discuss design strategies and system architectures for CPSEs. The goal was to apply our findings from the previous day to the processes of designing and developing real CPSEs.

The five CPSE topics covered during breakout sessions are described as follows:

1. **Distributed resource management for collaborative problem-solving** – Scientists employ different kinds of scientific and computational resources in their research work. We seek to identify the various classes of resources that scientists may apply, the conceptual organization or model from which these resources are employed, and the mapping of these resources to specific applications, computers, and data. In our investigation, we wish to explore and identify viable and novel approaches to resource management for collaborative problem-solving. Important questions we might ask include the following:

    - What are typical and critical scientific and computer-based resources used in collaborative problem-solving?
    - How are scientific and computer-based resources intermixed and applied?
    - How should resources be managed and accessed?
    - How should new resources be incorporated?
    - How should resources be distributed across networks?
    - What are useful abstractions of scientific and computer-based environments for interacting with and applying resources?

2. **Data management for collaborative problem-solving** – Data management issues abound in collaborative problem-solving. As scientists conduct research and experiments, they encounter and produce many different forms of data. Scientific experiments may produce extraordinary amounts of raw data that need to be filtered and digested by scientists. Scientists may annotate experimental data to document its scientific attributes. They may convert the data to different formats to perform different kinds of analysis or to feed the data to other experiments. They may reference metadata to review the data's history and transformations. We want to understand the characteristics of the data that scientists employ, how that data is used, and what approaches are reasonable to organize and manage the

data in support of collaborative problem-solving. Important questions we might ask include the following:

- What kinds of data should be supported?
- How do we maintain the data internally?
- What kinds of metadata should be supported?
- How should data be accessed? What is the interface?
- What are useful abstractions for interacting with and using data?

3. **Understanding, capturing, and supporting scientific collaboration and research** – To support collaborative problem-solving, we need to comprehend the very nature of how scientists collaborate and solve problems. CPSEs need to address real scientific problems in ways that are driven by and consistent with the activities, processes, and interactions usually conducted by scientists. We seek to understand the research process workflow carried out by scientists in their domains, the internal and external collaborations that scientists develop, and various forms of knowledge and expertise that scientists share in their collaborative endeavors. Our analysis of these areas could then serve as a basis for designing and implementing higher-order systems and tools that enhance the scientist's ability to solve specific problems. Important questions we might ask include the following:

   - What are the different ways scientists collaborate to conduct research?
   - In what ways do remote collaboration capabilities support problem-solving?
   - How do we capture and represent the intelligence, expertise, and experience of scientists?
   - How do we best facilitate the dissemination of knowledge and experience among collaborating scientists?
   - How do we capture, represent, and share the research process workflow carried out by scientists?
   - How does the workflow decompose into atomic steps and actions?

4. **Strategies for designing CPSEs** – Collaboration and problem-solving are essential human qualities. As such, the development of CPSEs for scientific research demands the active participation of domain scientists who will carry out the collaborative and problem-solving functions. Traditional software engineering methods may need to be replaced or augmented to better support the direct involvement of scientists in system analysis and design. We seek user-centered and participatory analysis and design techniques that bring domain scientists in as system analysts and designers, capture collaboration and problem-solving requirements for scientific research, and evolve these requirements into specific design solutions. Important questions we might ask include the following:

   - What are the unique software engineering challenges in developing CPSEs?
   - How do we capture domain problems and solutions and incorporate them into a computer-based environment?
   - How do we most effectively involve users?

- How do we best gather requirements for collaborative problem-solving?
- How do we best design CPSEs?
- How do we make CPSEs flexible and adaptable to different classes of users?

5. **Future system architectures for CPSEs** – Various collaboration and problem-solving environments have sprung up in academic and commercial research and development efforts. Our goal is to combine the theories and capabilities of such environments into a unified architecture for developing CPSEs. We no longer view collaboration and problem-solving as separate attributes, but as a synergistic account of how scientists solve problems together. We seek to extrapolate and evolve current collaborative and problem-solving system frameworks to support this new concoction. Important questions we might ask include the following:

   - How do we combine collaborative and problem-solving technologies in a useful way?
   - What are the primary functional components of a CPSE?
   - How do we accommodate extensibility (i.e., adding new collaboration and problem-solving tools)?
   - How do we accommodate system-level modifications (i.e., data formats, database interfaces, application interfaces, computer interfaces)?
   - How do we accommodate legacy applications?
   - How should system features be organized to best support collaborative problem-solving?

1. Distributed Resource Management for Collaborative Problem-Solving Environments

Participants

Ray Bair	Fred Johnson	Carmen Pancerella
Henri Casanova	Bill Johnston	Bahram Parvin
Edward Chow	Carl Kesselman	Dave Thurman
Nenad Ivezic	Karen Schuchardt	

Introduction

The distributed resource management session consisted of a mix of technology providers and PSE developers. We focused on gaining an understanding of the following issues:

- the components of a distributed resource management system
- the different types of CPSEs and how they might benefit from distributed resource management
- the current state of the art in distributed resource management and the limitations of these systems
- developments for resource management systems to be useful (and used) in the development of CPSEs and the aspects of CPSEs (if any) that would not be addressed by the normal evolution path of resource management systems.

Important Issues

We identified the resource management requirements of CPSEs and the currently available resource management systems and why these systems are not being used to develop existing CPSEs. We then discussed changes required to increase the use of resource management infrastructure by CPSE developers.

Current State of the Art

Distributed resource management addresses the issue of providing capabilities or services to the CPSEs These resources include not only computers but also storage systems, networks and display devices. and data sets and display devices.

An examination of common CPSE paradigms, such as workflow management, tool integration, and collaborative tools, shows that distributed resource management shows up in many different forms.

- **Workflow Management**: resource discovery and scheduling, quality of service, fault-management, accounting, cross-enterprise sharing, event management, policy, and access control
- **Tool Integration**: use of software as a resource (code and license management), format conversion, interface discovery, and resource characterization
- **Collaborative Tools**: sessions, people, and organizations as resource.

In general, the resource environment in which CPSEs operate can be characterized as follows:

- The environment contains a large number of potential resources.
- No centralized control exists; each resource is separately administered and operates under different policy.
- The set of resources is dynamic, due to failure of additional new services. Furthermore, the behavior of the resources themselves is dynamic, with capability varying over time.
- The resource set is highly heterogeneous. This heterogeneity spans many dimensions, including hardware, software, policy, and capability.
- The operating environment is hostile. Resources and applications must protect themselves against unauthorized use.

Resource Management Characterization

It is constructive to consider a generic characterization of the resource management process to understand the requirements for CPSEs. We agreed that resource management actually refers to the life-cycle of a service, which includes allocation, deallocation, monitoring, and control. The resource being managed is in some state, which can be observed by monitoring the resource. Monitoring can be event-based or query-based. Explicit state changes can be caused via a control operation. All of the above operations (allocation, deallocation, monitoring, and control) are subject to access control and local policy.

As mentioned above, the resource being managed is characterized by its state. Additional characterization includes general characteristics (i.e., metadata) and the name and location of the resource. To support these activities, the resource management element must contain naming and discovery services, as well as the basic management capabilities.

Several current technologies address the basic requirements of a distributed resource management system for CPSEs. These technologies include Grid services, like Globus, or services produced by the Grid Forum; more traditional distributed computing systems, such as Enterprise JavaBeans, CCA, CORBA and other OMG services; and T.120-based collaboration tools and workflow management systems. However, we observed that PSE implementation tends to not use infrastructure services but rather reimplement basic services from scratch.

Barriers and Challenges

While some of the reluctance to use existing tools can be attributed to "not invented here," some genuine limitations exist with the current distributed resource management systems. Depending on the service, the limitations include the following:

- a focus primarily on computational resources, at the expense of other resource types such as networks and storage
- limited application to enterprise level, rather then supporting inter-enterprise resource management
- limited availability of advanced services such as advanced reservation, scheduling, and fault recovery
- closed, unextensible implementations
- lack of implementation of published specifications (e.g., CORBA security).

In spite of these limitations, we expect future developments in these systems to address the resource management requirements of CPSEs. These advances will include the explicit management of noncomputational resources, advanced control management, requirements matching between user and resource provider usage patterns, and component technologies. In addition, we expect emerging technologies, such as HP's E"speak, agent systems (like those proposed by OMG and FIPA), and teleimmersion/video conferencing technologies to provide further resource management solutions. Furthermore, standard activities, such as OMG and the Grid Forum, will increase the availability, and hence utility, of distributed resource management technologies to CPSEs.

The group felt that an overriding challenge to develop resource management abstractions that facilitate the introduction of advanced resource management strategies into domain-specific CPSE design and development. We lure developers away from the temptation to develop their own one-off tools only when the infrastructure is an order of magnitude easier or better then the current situation.

Recommended Actions

In general, the requirements of the CPSE community are in line with the general trends of Grid technology. However, specific aspects of CPSEs should serve as drivers in the design and development of Grid services. For example, the collaborative nature of CPSEs places a special requirement on group security and group control structures. While this infrastructure is important for other application areas as well, it is of particular importance to CPSEs; thus, the problem-solving community should participate in and drive the development of these aspects of the infrastructure.

Insights and Revelations

Based on the discussion in the workshop and within the breakout group, it is my opinion that emerging Grid technologies are particularly well suited to addressing the requirements of CPSE developers. While CPSEs will stress some facets of Grid, it does not appear that any requirements are significantly out of line with the trends we are seeing in the overall Grid community. To this end, CPSE developers should actively participate in the Grid Forum.

Existing Grid technologies can already have a significant impact on the functionality that a developer can provide to a PSE end use. It would appear that the most significant impediment is the lack of Grid that enables PSE development tools, such as components, CPSE-specific resource management strategies, etc. Based on the results of this meeting, I would focus on existing Grid technologies with the overall goal of improving the reuse of Grid-related technologies across different CPSE implementation and domains.

2. Data Management for Collaborative Problem-Solving Environments

Participants

Ulrika Axen	Olle Larsson	Elena Mendoza
Chaitan Baru (Moderator)	Anita LoMonico	Michel Sanner
Gary Black	Richard May	Paul Saxe
Donald Denbo	Richard McGinnis	Vaidy Sunderam
Al Geis		

Introduction

The session began with a presentation by the moderator on issues related to *interoperability*. Data and service interoperability among computing environments was identified as a fundamental prerequisite to enable CPSEs. In later discussions, we also identified security as another important prerequisite.

The initial group discussion focused on identifying the specific data management aspects of CPSEs. We attempted to identify data management requirements of PSEs and compared requirements of PSEs versus CPSEs, i.e., what are the additional requirements imposed by the need for collaboration in CPSEs. Requirements for PSEs include the need for data to be available online; support for metadata, including processing and access history and annotations; the ability to search using metadata (i.e., the use of metadata as a finding aid for identifying data sets of interest); the ability to deal with large data sets via filtering and similar techniques; and visualization of data sets, especially for large amounts of data. In addition, CPSEs require interoperability, security, and the ability to deal with metadata *quality* issues.

We then organized deliberations along the following topics: user data management requirements for future CPSEs; survey of the state of the art; anticipated technical advances relevant to CPSEs; and suggested actions for moving CPSEs forward. These discussions are summarized below. It is possible to organize the various topics along four major themes: *interoperability*, *metadata*, *security*, and *usability/performance* (the last category reflects the fact that good performance is essential for building effective and usable CPSEs).

Important Issues

Interoperability

Participants in CPSEs will have data they wish to share with other members of the CPSE. How can sharing be achieved? Participants may have their own schemes and formats in which they produce data. The assumption is that these schemes/formats will continue to be used in CPSEs, (i.e., we cannot assume that users will be willing to change data formats only to participate in CPSEs. In the most general case, each participant may produce data in a different format, yet all participants would like to share data with each other.

Tools must be provided to translate data from one format to another. The most effective approach to facilitate sharing is to provide conversions from the varied formats into a common format, and from the common format to each individual format. While significant challenges may exist in implementing such a scheme and in building the necessary tools, this approach is still feasible. A standard for common data format could be based on the Extensible Markup Language (XML).

Once the data format issue is addressed, the next significant challenge is the translation of "concepts" and "vocabulary" among the domains. Data format translation is an interoperability issue at the "syntax" level, while concept translation is a "semantic" interoperability issue. Thus, even if the CPSE employs XML as the common format for data exchange, that still does not solve the problem of correctly interpreting the contents of the corresponding XML documents. For example, two different scientists/domains may use keywords such as "rain" and "precipitation." The two terms may or may not refer to equivalent concepts, depending on the context in which they were defined and in which they are being used. A need clearly exists to identify and define the metadata required to aid in semantic translations. Because the definition of terms is generally within a "context," it is necessary to understand how to represent that context and how to exchange context information in a CPSE. The solutions are to try to move toward a common vocabulary and ontology and/or provide mappings of terms. Some of the science communities are beginning to move toward such standards. For example, it was mentioned that the geophysics community has defined a hierarchical system of language.

In summary, defining a common standard for translation is a prerequisite for sharing data.

Metadata

Scientific metadata is needed to properly use data sets. The metadata can be used as a "finding aid" (i.e., for users to query and identify data sets of interest).

The data interoperability issue is applicable to metadata as well. How do two metadata owners interoperate? Once again, it is important to start with standards for metadata. While such standards are beginning to emerge in several science disciplines, in many cases, we felt such standards were ineffective. First, the standards seem to change almost as soon as they are proposed and codified. Second, the reality may very well be that while the standards exist, they are not being adhered to. These problems lead to two major issues:

1. CPSEs must be able to deal with situations where metadata standards are changing ("any standard will be changed the instant it is proposed," was a statement heard during the discussion), requiring the ability to model, process, and query "versioned" metadata.

2. A system is required to *monitor and police* the adherence to standards and *assess* the "*quality*" of metadata from a given information source. Because metadata is the key to searching, translating, and

understanding the associated data set, we felt that compliance with standards should be "policed" in some way. The group expressed a strong opinion that the *quality* of metadata is of major concern. In addition to policing the adherence to standards, methods for measuring and quantifying the quality of metadata are required. To facilitate these methods, a need exists to arrive at metrics and benchmarks.

We also observed that multiple metadata structures and standards already exist in many disciplines. Thus, even within a given discipline and for a single data set, it may be necessary to support many kinds of metadata "views" for that data set, each associated with a particular "standard." As mentioned earlier, the ability to define common standards is a prerequisite to sharing metadata.

Security

We strongly felt that a lack of proper and proven security methods would be a definite barrier to participation in CPSEs. The willingness to freely share data and information greatly depends on being confident that the data will be accessed by only the people for whom it was intended. Providing a comprehensive access control mechanism, and fine-grained access control, is also essential. Data providers must be able to decide which information they wish to provide/publish and which to keep proprietary, to the lowest level of detail possible.

While the collaborators in a CPSE may implicitly trust each other, the general issue is whether they all trust the underlying security infrastructure and mechanisms provided by the system. Robust, trusted security mechanisms, including authentication and access control, are key to the successful operation of CPSEs.

Usability/Performance

Collaboration may occur in both synchronous ("online") and asynchronous ("offline") environments. In either case, necessary data should be online and easily accessible in an interactive fashion. Datasets should be amenable to automated searching. Software tools must be designed to read data interactively or online. The data may be "local" to a PSE/CPSE or accessible from a remote database. Data replication may be needed to improve performance.

PSEs are generally not set up to handle large datasets. Additionally, CPSEs almost always include remote participants and possibly mobile clients. Thus, work is needed on filters and filter structures to reduce the amount of data transferred from the data servers to the clients in the collaboration environment. Another approach for improving usability and performance is to employ data-viewing techniques to display summary versions of large individual data sets and/or large result sets. To facilitate fast and efficient data searches, appropriate indexing and searching techniques are needed.

State of the Art

Interoperability

Interoperability among environments is being facilitated by the emergence of *ad hoc* as well as formal standards to represent metadata and data in several communities. These standards are at the syntactic (data format) as well as semantic (vocabulary, ontology) levels. In the former case, the standards specify how information is to be represented and encoded. In the latter case, the standards also specify the terms employed and their meanings, which allows for correct interpretation of the content. Languages such as XML, RDF, XML-Data, DCD attempt to provide standards at the syntactic level, which are applicable regardless of the domain in which they are used. Standards for semantics are domain-specific and require discussion and consensus within each domain.

Once syntactic standards have been agreed upon, a technology that can help interoperability is software for "wrappers." Wrappers translate "legacy" data sources from "native" data formats to the common, agreed upon format (e.g., XML). Early work has begun in XML wrapping technologies. Other technologies that can help with interoperability include standards-based API libraries that offer a common environment in which a variety of tools can be used in a "plug-and-play" fashion.

Metadata

Standards are emerging for the types of metadata to be collected and associated with data sets in each domain. Automated methods for creating and storing metadata are becoming available. XML-based standards for representing and storing metadata are also beginning to emerge.

More work is required in all of the above areas so that all necessary metadata is available in the future for users interested in analyzing and re-examining data and information. A remaining challenge is determining the *quality* of metadata.

Security

The commercial sector is making many strides in security technology because security is a cornerstone of e-commerce. Existing authentication and encryption systems, including Kerberos, SSL, X.509, will be very useful in CPSEs. However, as technology is being applied, it is revealing its weaknesses and complexity. As users become aware of the shortcomings of some of these technologies, they may demand more from CPSEs.

One area that will clearly require more attention is access control. CPSE participants will be very interested in "deep integration" of access control in the CPSEs that will allow for fine-grained access control mechanisms.

Usability/Performance (Interactivity)

Access to large data sets and the ability to analyze large data sets is improving due to improvements in processing, storage, and networking technologies. Efficient external memory algorithms are helping to deal with out-of-core data sets; i.e., individual data sets that cannot be accommodated in their entirety in main memory. In addition, new search and indexing techniques, intelligent agent technology, and feature extraction and progressive visualization techniques are making it possible to more effectively handle very large result sets.

Barriers and Challenges

A broad challenge is to keep current and potential users of CPSEs abreast of technology developments. Some collaboration systems, standards, tools, and techniques already exist to serve as a starting point in certain areas, for those interested in starting simple CPSEs. Because everyone interested in CPSEs may not know about such technologies, the first challenge is to compile information on currently available technologies and to make that information available to appropriate communities interested in starting CPSEs.

As a general strategy, it is also important to encourage the development of tools that make it easier to share data and to collaborate. One observation was that in many cases minor annoyances with the data formats and/or tools discouraged the sharing of information. Thus, fixing some of the simpler problems may go a long way in encouraging the use of CPSEs. For example, many people develop data translators, but they only build a "one-of-a-kind" tool for their own use. No incentives exist to build extensible tools that may be used later by others. Incentives should be provided to build and extend such tools to enable community-wide use.

Another emerging trend that creates new challenges in CPSEs is the move toward remote use of scientific instruments. This trend requires technologies that can better support interactivity in CPSEs so that remote users feel they are physically close to the instrument. It also requires distributed concurrency control and synchronization techniques to ensure that only one person is in control of any given instrument at any given point in time.

Interoperability

Challenges in interoperation of heterogeneous platforms/clients are beginning to be met partially by systems such as XML, Java, and CORBA, but more work needs to be done.

The ability to share data by defining views of data, data collections, and a data manipulation methodology have also been partially met. However, many data models exist with a need to specify view definition capabilities on them. Some translation issues remain as well.

Metadata

Many issues regarding the quality of metadata need to be addressed, including quality metrics and standards for benchmarking. An approach for policing the standards and providing annotations in cases where the metadata standards have not been met is required. Methods should also exist to determine the quality of metadata and detect when the metadata has been "corrupted." Where possible, automated warp to obtain the necessary metadata should exist. Interoperability at the metadata level also requires the development of common vocabularies and concepts. An important type of metadata that can help search data sets is index metadata.

Security

Authentication and access control requirements are partially met. A major challenge will be "psychological issues", i.e., willingness to trust the security software infrastructure. Tamper-proof schemes are also needed to limit access such as data set *watermarks* and keys (*tickets*).

While the security, encryption, and access control challenges are partially met, constant shifts in technology will remain. This issue has an impact on the ease of use and maintenance of the security infrastructure, as well as the overhead required to keep up with changes. These impacts affect the longevity of any scheme that is used and, ultimately, users' believability and buy-in into any security scheme.

Usability/Performance (Interactivity)

Many issues related to data caching have already been studied. Intelligent prefetching and "progressive" transmission techniques are required as well as effective ways to deal with tertiary storage and enormous databases.

While local/remote data access challenges have been partially met, better tools for data discovery, metasearch, and search refinement are needed.

Challenges exist in visualization of remote data sets. Issues include asynchronous vs. synchronous access to data; feature extraction and discovering data signatures (i.e., finding the one piece of data you want), and annotation of visualized data.

Recommended Actions

- To support metadata and data interoperation, we encourage the use of a semistructured data model that can address a broad range of needs. In particular, attention should be paid to the use of XML-based standards for representing metadata and, where possible, the data as well.
- We recommend support for a CPSE infrastructure that promotes tool interoperation using systems such as CORBA or emerging standards such as Jini.

- For ubiquitous security in the future, we recommend studying the use of smart cards, which can help in overcoming the users' "psychological" barrier to security related to system complexity.
- We encourage the use of digital library search techniques and intelligent agents that can greatly enhance the information discovery and search capabilities as well as usability of CPSEs.
- We encourage use of standard data formats and vocabularies to further encourage interoperability among environments and thereby facilitate CPSEs.
- We encourage the development of wrappers and translators that can translate between legacy/native formats to the common format.
- We encourage building tools using component technologies that can enhance the extensibility of the tool and support the plug-and-play model where participants can bring their tools and/or others tools on their data in a CPSE.
- We feel deep integration of access control mechanisms is important. While participants wish to share information in a CPSE, they are also quite keen on carefully controlling who sees what data (and for how long).
- Finally, an advantage of CPSEs is the ability to archive sessions and "replay" sessions. There is a need for technology that can support easy archiving, indexing, retrieval, and replay of CPSE sessions.

Insights and Revelations

Broadly speaking, it is important to find the "right" type of problem for CPSEs to be successful; i.e., find problems that can be effectively solved collaboratively in a computer-aided problem-solving environment. For continued success, users must be able to answer the question, What does a CPSE save in time and/or money? A proper answer to this question will help set expectations in the minds of prospective users of CPSEs. Another way to set expectations is to encourage prospective participants to ask the question, What do I expect from a CPSEs? By answering this question, the participant may learn in advance the effectiveness of using a particular CPSE for a particular problem.

It should also be noted that CPSE can be constructed in a variety of environments including those where all participants are "local"; i.e., on the same computer system or in the same physical location. A CPSE may be constructed where the collaboration need not occur in real-time; thus, offline or asynchronous collaboration could exist. In some cases, CPSEs may be used to collaborate only on portions of a problem; i.e., there can be levels of collaboration and not necessarily full collaboration, which also relates to data sharing. Collaboration could occur by sharing data only partially and not fully.

It is important that a CPSE not put additional burdens on users. From a data management perspective, users should be able to create and use data as before without having to do something special just because they are using the data in a CPSE.

Documenting the scientific process is extremely important to facilitate collaboration. This issue implies the need for a workflow-like system that is sufficiently complex to serve all the needs of a problem-solving environment. In addition, some experiments are done over time and the system should provide the necessary persistence.

An important insight from this workshop was the need for quality metrics associated with metadata. Although standards established for metadata may exist, these standards may not always be adhered to. Thus, given a data collection, a need exists to evaluate the quality of the metadata of that collection and, ideally, characterize the metadata quality using some metric. There is also a need for benchmarks against which the metadata quality can be measured. In the long run, a review system is needed where users can obtain reviews of the metadata quality of a given information source (of course, this need gives rise to the cyclic problem; i.e., it may be necessary to obtain reviews of the reviews).

Finally, users will not freely share information and will not feel comfortable about participating in CPSEs unless they are confident about system security.

3. Scientific Collaboration and Research for Collaborative Problem-Solving Environments

Participants

Kevin Brunner	Richard Kouzes	Marek Podgorny
Shahrokh Daijavad	Mike Leuze	Kimberly Rasar
Terry Disz	Stu Loken	Mary Anne Scott
Lori Freeland	Cheryl Marks	Eswaran Subrahmanian
Larry Jackson	Jim Myers	Ravi Subramanya
Farnam Jahanian	Arthur Olson	Christine Yang

Introduction

In this session, we addressed two major questions:

- What is the nature of research and collaboration?
- How can we use information about the nature of research and collaboration in designing and implementing systems and tools?

The first half of the session was a brainstorming activity discussing the following questions:

- What are the different ways scientists collaborate? Do scientists and engineers collaborate differently?
- What processes are fundamental to the research process or collaboration activities? Which are dictated or shaped significantly by technology?
- Which workflow components are common and which are application-specific?
- Do we expect scientists to adapt to technology or technology to adapt to scientists?
- How do we begin to use information about the nature of research, the collaborations scientists and engineers develop, and the types of knowledge and expertise that are shared to shape the environments we build?

During the second half of the session we began to answer specific questions suggested by the conference organizers:

- What are the different ways scientists collaborate to conduct research?
- In what ways do remote collaboration capabilities support problem-solving?

- How do we capture and represent the intelligence, expertise, and experience of scientists?
- How do we best facilitate the dissemination of knowledge and experience among collaborating scientists?
- How do we capture, represent, and share the research process workflow that scientists carry out?
- How does the workflow decompose into atomic steps and actions?

Due to time constraints, after discussing the first four of these questions, we shifted our focus to the following general topic areas:

- What aspects of this topic are important in developing CPSEs?
- What are the user requirements for future CPSEs relevant to your topic area?
- What are the major challenges?
- What actions are recommended to move CPSEs forward?

We ended the session with a discussion of ways to know whether a successful collaboration has taken place. We concluded that in a successful collaboration:

- All parties involved receive benefit.
- Things are accomplished that cannot be done without collaboration.
- Time and money are saved.
- Close working relationships are developed and sustained.

Insights into the nature of collaboration that emerged during the discussions included the following:

- The basic unit of scientific collaboration tends to be a senior researcher and a small number of assistants, such as graduate students, post-docs, research associates, or junior researchers. This unit seems to exist across a wide range of scientific disciplines.
- The willingness of researchers to collaborate is influenced by many things, including proximity of the research to the commercial world and the nature of the resources used (scarce vs. readily available).

The session went well, the discussions were relevant, and participation was broad-based. However, the time allowed for the session was too short to address the issues more than superficially. While most of the suggested topics were discussed, considerable effort was required to organize discussion results for a presentation.

Important Issues

Important issues for scientific collaboration and research for CPSEs include the following:

- accommodating different models of collaboration
  - large scale vs. small scale
  - tightly coupled vs. loosely coupled
  - synchronous vs. asynchronous
  - persistent vs. temporary
  - consensus-seeking vs. general exploration
  - proprietary vs. public
  - interdisciplinary vs. intradisciplinary
- determining the impact of technology on current practice
- capturing, annotating, storing, organizing, filtering, and retrieving appropriate information
- supporting collaboration through different stages of research
- finding the balance between user needs and technical capabilities
- integrating domain-specific contributions into the environment.

Current State of the Art

Because the session topic was the nature of scientific collaboration and research, we did not address current state of the art.

Barriers and Challenges

The barriers and challenges to scientific collaboration and research for CPSEs include the following:

- understanding and capturing the user's research and workflow processes
- capturing and delivering knowledge at the appropriate level
- dealing with cultural and semantic differences across disciplines
- integrating problem-solving capabilities with collaboration capabilities in a seamless, natural way
- encouraging and facilitating collaboration while protecting intellectual property
- consolidating diverse user and developer views of CPSEs
- defining metrics
- dealing with proprietary and public domain tools
- educating and training users and developers
- marketing CPSE successes and promoting CPSE usage.

Recommended Actions

The following are lists of recommended actions to encourage and enable CPSEs:

General:

- increase research and development funding to reach critical mass
- encourage interagency collaboration
- build test-beds to demonstrate diverse capabilities
- support outreach and focused marketing efforts
- provide incentives to future users
- promote standards for data and process exchange
- employ open-source models.

Scientific:

- identify research communities that CPSEs must support
- involve domain scientists in the development process
- identify current and future resources and work environments
- improve models of the research and workflow processes.

Insights and Revelations

Most of the important issues related to CPSEs were addressed, but the workshop helped identify an additional concern. Ontology development is greatly needed. A fundamental function of any collaboration tool is the building of shared ontologies that allow collaborating parties to communicate at the semantic level. Often this functionality is not acknowledged and is only marginally or informally addressed. This approach, however, will tend to result in brittle systems that cannot be easily maintained or extended to address real-world collaboration issues.

4.1 Strategies for Designing Collaborative Problem-Solving Environments I

Participants

Ray Bair	Richard May	Mary Ann Scott
Terry Disz	Richard McGinnis	Ravi Subramanya
Debbie Gracio	Suzanne Dunn McNeil (Facilitator)	Dave Thurman
Richard Kouzes	Paul Saxe	Val Watson (Moderator)
Stu Loken		

Approach and Group Composition

This group addressed questions posed by the Pacific Northwest National Laboratory (PNNL) team and, despite major differences in opinions, worked well together to create these findings and recommendations. The facilitator did an excellent job of keeping the discussions on track, moving, and inclusive of all participants' viewpoints. We discussed the following specific topics:

- important issues in developing CPSEs
- current practice in CPSE development
- barriers and challenges to effective CPSE development
- recommended actions.

The PNNL team provided the same topics as those listed above except the team used the word "design" instead of the word "development." (Ray Bair confirmed that the scope of our discussion should be the development of CPSEs rather than just the design.)

The group members were from DOE laboratories, NASA laboratories, universities, a supercomputer center, and a private company. The group also had a mix of domain scientists (users of CPSEs), computer scientists (developers of CPSEs), and managers of CSPE projects.

Some group members were concerned that the term, "Collaborative Problem-Solving Environment," means greatly different things to different individuals. So at the beginning of the meeting, the following crude definition of CPSEs was accepted as a reasonable definition for this exercise:

An environment to facilitate research/communication with electronic tools over geographically dispersed areas. Typical elements may include

- phone
- e-mail
- video conferencing

- immersive environments
- software
- shared windows
- electronic notebooks
- chat box.

Important Issues

The most critical and difficult aspect of the development process is "gaining an understanding of the problem and the resources." Communication between the domain scientists who will use the systems and the computer scientists developing the systems is usually difficult. The potential user does not know what is possible and the developer does not know exactly what the user needs. Therefore, the development process should be an iterative process of prototypes and continuous improvements, which facilitates the education of both the users (on what is possible) and the developers (on what works best for the users). (The developers should not forget to ask the question, "Are CPSEs even relevant for this problem?")

Other important issues include

- understanding the user's problem-solving process in addition to the problem to be solved. Unfortunately, modeling what people do when they solve problems is not easy.
- identifying "champion" users and creating good communication with users
- barriers to implementation due to geographic distribution (discussed in detail in the section "Barriers and Challenges" below).

One member pointed out that, "the state-of-practice is never adequate."

Current Practice in CPSE Development

We believe that current CPSE development practices are the same as other software development practices. The development groups use whatever practices they have been using for other projects. (A difference exist from group to group, but not from project to project within the same group.) Some of the practices currently in use are

- rapid development/prototyping - cyclical
- development of components (Lego concept)
- sustained test-beds (making the lifetime long enough to be useful to users)
- building on existing tools and techniques.

Barriers and Challenges

We created the list of barriers below and then ranked their importance. However, the ranking is very problem-dependent, so these rankings represent only an average ranking over a mix of problems. (Considerable controversy existed about these rankings.)

Most important:

- ability to understand scientists' needs
- multi-platform environments
- cost/benefit issues for user participation.

Next in importance:

- sustained maintenance and support of systems
- geographic distribution
- latency
- bandwidth limitations
- "more moving parts" (i.e., more complexity)
- quality of service
- ownership/control
- time zone differences
- nonuniformity of resources
- dynamic user interfaces (e.g., 3D, immersive).

Next in importance:

- security/privacy issues
- proprietary issues with commercial software
- inability to transfer technology to the private sector so that it continues to be used
- organizational issues
- intellectual property issues
- budget
- politics
- resources.

Recommended Actions

The following recommendations were not ranked in importance:

- encourage collaboration at the organizational level
- create interdisciplinary teams

- use domain scientists and computer scientists in a participatory development
- include diverse computer scientists (e.g., data management, user interface, network)
- treat all players equally (use the term "we" rather than "them" when referring to other participants in the development)
- market/educate management about the need for integrated design teams
- identify opportunities and "champion users"
- gradual implementation (one member characterized this recommendation as, "First give the users what they want and then give them what they need")
- use rapid development methods
- design for flexibility and conduct frequent reviews (to help prevent locking out new technologies that may become available during development)
- use a systems analysis approach for modeling the problem-solving process
- define a software environment and tools (we felt that we should go beyond specifying of an architecture by specifying the specific environment and tools)
- select cross-platform tools (with the advent of handhelds and other devices, heterogeneity will increase in the future)
- use common data formats
- use standards
- use multiple sustained testbeds
- include measurements of success
- include training/education of users/designers (meant to include education in schools).

Insights and Revelations

Although the breakout sessions accomplished the desired goal, the moderators should have strived for more forward-looking sessions. Most of the issues, barriers, and recommendations extracted from the group reflect the current state and were already known to the attendees. (The PNNL team is commended for trying to inspire more forward-looking issues with Tom Furness' presentation, but the inspiration did not appear to carry over to the breakout sessions.) Therefore, the group should explicitly solicit forward-looking issues though the website and/or the authors of the workshop report should try to surface some forward-looking issues. The following are examples of the type of issues we can try to add to the report

Example of Issues Likely to be Important in the Future

Projected technologies of the future:

- Grids (such as the NASA IPG) with a layer of services should be available and are likely to be used extensively by scientists for solving complex problems.
- More sophisticated computer-user interfaces should be available which may include

- displays with greater resolution, greater fields of view, and smaller footprints (e.g., those described by Tom Furness)
- voice
- speech recognition
- gesture recognition
- situation awareness (intelligence about what the user is doing, what resources exist and located, and with who the users collaborates)
- automatic authentication from personal features.

Significant improvement in the capabilities and performance/cost ratios of "personal" computers should continue.

Issues related to the future technologies above:

- Resource access (to data, people, and computing resources) will be obtainable from the grid services, and most scientists who are solving very complex problems will already be using these services.
- Authentication and security for remote resources will be handled by the grid services.
- The low cost of "personal" computing power combined with the effectiveness of the sophisticated computer-user interfaces will make the use of "thin clients" inappropriate.
- We will probably be unable to send highly dynamic, high-resolution, large field-of-view images in pixel form (even with scientifically acceptable compression) over the network in the near future. (The infinite bandwidth concept will not materialize in the near future because the number of user accesses to the network will increase and the volume of information per access required to drive the sophisticated interfaces will increase and offset the increases in line speeds. Many applications will continue to be limited by network bandwidth unless the information sent over the net can be tailored with some intelligence about application to reduce the required bandwidth.)
- The attractiveness of web browsers will diminish as "personal computers" begin to automatically handle the access to resources and as the inefficiencies of working through http servers become apparent. (These inefficiencies should become apparent when users begin to have highly interactive remote sessions with complex 3D dynamic scenes.)

Recommendations based on the above issues:

- Design the CPSEs to work with the available grid services. (For example, the PSEs in Rick Stevens' Corridor One Project are layered on top of the grid services.)
- Design the CPSEs to work with the new, more sophisticated computer-user interfaces.
- Design for "thick" clients with substantial computing and graphics rendering power. (It will probably be cheaper and more effective to have the client perform many tasks.)
- Do not design for use with the web browser (going through the http server) for all communications, but instead design highly efficient communication components that can handle highly interactive sessions involving very complex data.
- Design the data to be sent over the networks to be as compact as possible by using knowledge of the applications. (Rick Stevens is planning to study data formats for visualization, spanning the range from pixels to scene graphs. However, in many cases it may be beneficial to go beyond this range to even more application-aware data forms. An example is the use of application-specific scripts to drive a remote scientific visualization tool. This technology is used in NASA's FASTexpeditions software, described in http://www.nas.nasa.gov/Software/FAST/FASTexpeditions. Highly interactive visualization of dynamic, high-resolution 3D scenes is obtained with only 1 Kbps of bandwidth.)

4.2 Strategies for Designing Collaborative Problem-Solving Environments II

Participants

Ulrike Axen	Nenad Ivezic	Arthur Olson
Kevin Brunner	Farnam Jahanian	Bahram Parvin
Henri Casanova	Anita LoMonico	Megan Schlitt (Facilitator)
George Chin (Moderator)	Cheryl Marks	Christine Yang

Introduction

The breakout session investigated strategies for designing CPSEs. The emerging concept of CPSEs presents new challenges and requirements to current software development practices. The notion of a CPSE is centered on the human traits of collaboration and problem-solving. To support and enhance these fundamental traits using computer technology, we need to comprehend how scientists collaborate and problem-solve in their native environments, and then incorporate and integrate this understanding into software designs and computer systems.

The session participants represent a diverse cross-section of professions. The session had strong representation from all sectors including federal research laboratories, government agencies, academia, and computer industry. The perspective of the domain scientist was effectively enforced by participants from the National Cancer Institute as well as other participants who had strong scientific backgrounds and/or had developed focused science-based applications. The group generally believed that strong representation from and involvement by domain scientists was critical for any CPSE project.

Important Issues

We investigated and discussed specific design topics defined by the collective group of workshop attendees. Prior to the breakout session, all workshop attendees were asked to identify the three most important strategies for designing CPSEs. The responses were collected and compiled. From the set of topics, we were to address those topics that were most prevalent in the collection of responses. We addressed the following four questions:

- How do we capture science (e.g., domain problems, scientific processes) and its requirements for CPSEs?
- How do we effectively identify and involve users in the development of CPSEs?
- What software development approaches are most effective for developing CPSEs?
- How do we connect the scientific to the underlying system architecture?

For each topic addressed, we were directed to clarify the design topic or issue, examine the current state of the practice, identify barriers and challenges, and provide recommended actions for future CPSE development efforts.

Current State of the Art

The question of how to best capture science in the development of CPSEs is a multifaceted issue. From a product perspective, we may examine how the science is actually represented and realized in the CPSEs we develop. From a process perspective, we may examine the analysis and design approaches we use to understand the domain science and to support and integrate the science with computer technology. We explored and discussed science capture from these two different views.

As we examined existing scientific applications and CPSEs, we found that the domain science is typically implemented within specific codes and instrumentation. Codes are developed to run specific scientific calculations or to run and collect data from laboratory equipment. Domain scientists tie these scientific codes and instrumentation together in an ad-hoc, piecemeal fashion to construct theoretical scientific models. The domain scientists can then share the theoretical models with other scientists and use them as comparisons against other models. Typically, a scientific model solves a specific, narrowly focused scientific problem. The ability to transfer or migrate the model to other problems and domains is often severely limited.

As much as current scientific models restrict the scientific problems they address, they also define the specific process that is carried out to reach a solution. The process may be referred to as the scientific workflow. Because science is implicitly represented by codes and instrumentation, the scientific workflow is implicitly represented in the order and manner that scientists apply codes, instrumentation, and other resources. For most scientific applications and CPSEs, the domain scientist generally follows a fixed sequence of steps, and thus, the scientific workflow is also fixed.

Existing scientific models typically do not provide explicit representations of the scientific workflow. Thus, the workflow is rarely explicitly captured. Yet, the capture of scientific process is a usual part of conducting science as scientists document their experiments and experiences in laboratory notebooks and rerun and reproduce experiments based on past results. In this case, scientific workflow is a dynamic process that is inherently part of the problem-solving effort. In contrast, engineering workflow models tend to focus more on procedure. The objective is to streamline and standardize the process for the sake of efficiency. The scientific and engineering views of workflow are very different yet most research on scientific workflow seems to address the engineering view of relatively static process. Not surprisingly, many domain scientists resist any form of a scientific workflow representation.

From the process perspective, two CPSE development approaches are prevalent in the current practice. One approach focuses on the domain science. Development is concentrated on domain-specific functions. Little effort is expended on developing general and reusable architectures. Rather, the domain science takes center-stage and development is typically confined to what is required to support domain applications. The second development approach focuses on underlying infrastructure services. The goal is to develop underlying CPSE services that may support a wide range of domains. Unfortunately, the result of this approach is commonly an architecture that is completely disconnected from any particular domain. Today, the two development approaches have promoted a tremendous gap between the functionalities of infrastructure services and domain applications.

CPSEs have been developed on different kinds of architectures. Art Olson of the Scripps Research Institute has developed several CPSEs by developing a modular code and applying scripting languages. The modular code defined specific scientific functions while the scripting languages tie the individual functions into a higher-level model. CPSEs have also been developed on top of lower-level frameworks or technologies. For example, some CPSEs have been developed using CORBA and its associated services. Others have been developed to use the web's infrastructure. CPSE may also be implemented via commercial software packages or environments. These commercial packages tend to be monolithic and noninteroperable with other systems and environments.

The role and contribution of the domain scientist in the development process is also an important issue for CPSEs. In the current practice, participating domain scientists are identified and employed in particular ways. Participating scientists are typically those who desire and/or are willing to be involved in the development process. In many cases, the participating scientists are pulled from existing collaborations with other members of the development team. Overall, the development team is typically comprised of a small core group of domain scientists that serves as the domain expertise. The precise function of the domain scientist varies among individuals and projects. We identified three different kinds of users involved in the development of CPSEs. The most obvious class of users is the domain scientist who understands the scientific concepts and context. Another class of users is the application software developer who typically has programming skills and may have a science background. These users develop the scientific applications for specific domains but rely on other domain scientists to provide the science content. The third type of user is a hybrid of the domain scientist and the application software developer. Through introspection, the hybrid user transforms his own understanding and knowledge into specific computer-based tools and capabilities.

Domain scientists are also involved in the evaluation of CPSEs. Some development teams include sociologists who study the use of a CPSE after it has been developed. Sociologists may employ surveys and questionnaires to collect feedback from a wide range of potential users. In terms of software engineering models, CPSEs have been primarily developed following the traditional waterfall model or a functional prototyping model.

Beyond development processes and products, some discussion was directed toward identifying the characteristics of collaboration and problem-solving. These characteristics may determine or impact the development processes that CPSE projects employ. For example, different science activities have different collaboration requirements. In the case of computer-supported scientific analysis and visualization, scientists may or may not collaborate in the analysis and visualization of scientific data and results. Thus, analysis and visualization may occur synchronously or asynchronously. When constructing scientific models and deriving solutions, however, scientists typically conduct these activities individually. Thus, synthesis activities are typically asynchronous. We note the distinction between analysis and synthesis to illustrate how scientists currently conduct their research, but we are also fully cognizant that the properties of the research will change with the introduction of CPSEs.

Barriers and Challenges

Capturing science and its requirements requires in-depth participation by the domain scientists. A frequent barrier is the lack of scientist buy-in to the development process and the emerging product. Conviction and commitment by the scientists is needed for the scientists and computer scientists to engage in constructive dialogue. Furthermore, a product is more likely to disseminate among scientists of a specific field if a champion of the product actively supports and promotes the product. The lack of a champion in the field of the domain science may also reduce the impact of a CPSE. For a CPSE to be popular and diversely applied, a domain scientist must actively promote it in the context of the science. Otherwise, the impetus and motivation for domain scientists to apply the CPSE may be missing.

Another potential barrier to the development of CPSEs is poor working relationships between domain scientists and computer scientists. The relationships are often polluted by false perceptions. Computer scientists often view the domain scientist as simply a computer user while domain scientists may view the computer scientist as just a programmer. In reality, the computer scientist and the domain scientist need to come together to derive functionality and effective systems. Each party needs to play a significant role in development. Cultural barriers, however, often exist between computer and domain scientists. They have different scientific goals, vocabularies, and professional backgrounds, which may impede their ability to effectively interact.

Furthermore, in many cases, domain scientists may exhibit a general lack of interest in the development of computer-based tools and capabilities. As a result, users may not necessarily be the most knowledge-able of the domain, but simply the most willing participants. In some cases, a lack of interest may be attributed to a lack of funding. Projects that wish to build interdisciplinary teams often have problems attracting funding that will support an interdisciplinary effort. Projects are often careful about the "color of their money" and are wary of "mixing the color."

Other barriers to capturing science center around issues of usability. First, if the use of a particular scientific application or service will be diverse, the application must be intuitive to and usable by the scientist. Other barriers deal with longer-term usability issues. Maintenance and operational support for specific scientific applications are generally lacking. Furthermore, applications may quickly become obsolete if they rely on evolving computer technology. Thus, technology obsolescence may limit the lifetime of an application.

Other barriers exist in the software development process. CPSEs require the development of base-level infrastructure and high-level domain applications. Yet, the development of infrastructure and domain applications has largely been conducted separately. To a certain extent, domain scientists have focused primarily on applications while computer scientists have concentrated on infrastructure, and a general lack of coordination has persisted between these two parties. A barrier to the construction of CPSEs is the lack of domain scientist involvement in the development of CPSE architectures. Consequently, CPSE projects suffer from a general lack of domain understanding.

The lack of domain scientist participation may be attributed to a variety of factors. Domain scientists may simply lack the incentive to work on infrastructure projects and interdisciplinary funding may be limited.

Furthermore, usable programming tools and environments, such as those based on visual programming paradigms, may not be available to domain scientists. Thus, domain scientists may be extremely limited in their participation because they cannot effectively engage in the development work. The ability to capture domain knowledge and to incorporate this knowledge into CPSEs is a tremendous challenge. A general lack of experience with user-centered and participatory design models currently persists among the CPSE development community.

With respect to architecture, various barriers exist to developing CPSEs on underlying technologies such as the web and CORBA. CPSEs designed to operate over the web may not be accessible due to institutional restrictions on web usage. Companies may have institutional policies that effectively block web access. The lack of effective security and access control mechanisms pose another problem as we try to facilitate access to data, codes, and machines. Developing CPSEs on top of standards-based software components may also involve steep learning curves to effectively apply the standards. Furthermore, most of the underlying technologies generally lack formal and de facto industry standards.

With respect to software engineering models, old development approaches, such as the waterfall model, are still being applied even though iterative, user-centered development models are probably more relevant to the development of CPSEs because of their focus on and involvement by the user. One challenge is to derive effective strategies for combining the development models and processes employed by the domain scientist with those employed by the computer scientists to derive development models that may accommodate both fields.

Recommended Actions

CPSE development is truly an interdisciplinary effort yet CPSE projects have encountered numerous problems and constraints in acquiring the participation of domain scientists, and garnering interdisciplinary recognition and monies from funding sources. Domain scientists need to receive greater incentives to participate in CPSE infrastructure projects and have access to more intuitive and usable development tools. Funding sources need to be made aware of the benefits and value of interdisciplinary development teams and be encouraged to fund collaborative ventures such as in the development of CPSEs.

A primary objective in the development of CPSEs is to improve the interaction between computer scientists and domain scientists. The interaction must be particularly close if the requirements and research processes of the domain scientists are going to find their way into the CPSEs that are developed. Improved interaction may be facilitated in a variety of ways. First, computer and domain scientists should put forth greater effort in cross-training each other in their respective fields. The computer scientist would learn about the scientific content and context to be incorporated into a CPSE while the domain scientist would learn about computer technology and the design of computer systems. CPSE projects should also develop multidisciplinary peer roles where the different members of a project yield similar levels of power and control. Regardless of whether the project promotes symmetric or asymmetric authority and command, however, the roles of the different disciplines and individual contributors should be defined early on to establish expectations and working relationships. The development of a common vocabulary is also important for domain and computer scientists to effectively communicate in their collaborative endeavors.

Generating greater interest in CPSEs among domain scientists requires a greater emphasis on education and training. Domain scientists need to be made better aware of CPSE capabilities and benefits. In addition, domain scientists need to develop a certain level of comfort and familiarity with computer technology before they can put CPSEs to practical use. Those domain scientists with "computer-phobia" will likely spurn the general notion of a CPSE. Awarding domain scientists for technology transfer may also improve the distribution and use of CPSEs by providing the scientists with an incentive to promote the technology.

To promote greater dissemination and usage of CPSEs, projects need to identify inspirational, motivating champions of the technology. These champions should be domain scientists who recognize the real scientific applicability of CPSEs. In addition, CPSE success stories should be better publicized to generate greater interest among the domain science community. From a wider perspective, computer and domain scientists may find other more general forums for interactions and discussions on CPSEs by attending multidisciplinary workshops and conferences. To establish common ground for and facilitate greater distribution of CPSEs, the emergence of standards bodies or consortiums would also be beneficial to the dissemination and promotion of CPSEs.

The participation of domain scientists in CPSE development may also be improved by applying more "user-conscience" software engineering methods. Over the last decade, various software development approaches and strategies have emerged from the field of human-computer interaction (HCI) that emphasize the role and active participation of users. These development approaches fall mainly under the categories of "user-centered design" and "participatory design." The application of user-centered and participatory design methods seems logical for CPSE development because of the required participation of domain scientists. Furthermore, specific analysis and design techniques need to be intuitive to and usable by domain scientists. Thus, techniques such as paper prototyping and visual programming would be highly applicable and effective for CPSE development.

When developing CPSEs, layered architectural approaches seem most appropriate. The goal is to build high-level problem-solving capabilities and functionality on underlying layers of infrastructure and services. Thus, the development approach is to first determine and develop the required layers of infrastructure, and then develop design models on top of the infrastructure. Component-based architectures are also useful in the development of CPSEs to support reusability and plug-n-play integration. Such architectures will allow different CPSEs to reuse basic underlying services while supporting the incorporation and integration of domain-specific components.

Technologies for CPSEs do exist in various forms. Another objective in CPSE development should be to incorporate and apply existing technologies and solutions. We should not "reinvent the wheel." One reasonable development strategy might be to simply examine the characteristics of successful CPSE implementations and then follow similar development strategies and technology applications.

To lessen the gap between application and infrastructure development, a greater emphasis should be placed on developing closer working relationships between application and infrastructure developers; in most cases, between domain and computer scientists. Yet, other more technical strategies may be employed to better merge application and infrastructure development. One useful direction would be to elucidate the missing layers, services, and abstractions between the domain layer and the underlying infrastructure. The definitions could serve as a roadmap for developing future generations of CPSEs. Another strategy for merging applications with infrastructure is to apply software and information modeling tools. Such tools provide mechanisms and guidance that may support and enhance the CPSE development and integration process.

Insights and Revelations

Three basic directions emerged from the breakout session. First, the most emphasized direction was the view of the domain scientists to be directly and meaningfully involved in CPSE development, largely representing a social perspective to CPSEs. Increasing the participation of domain scientists requires changes in perceptions and culture. Domain and computer scientists must be encouraged, motivated, and committed to working with one another. Developing domain scientists that are champions of CPSEs is a critical step in the promotion, dissemination, and pervasive use of CPSEs. We also need to enlighten and educate the funding agencies on the benefits and value of cross-disciplinary projects that will bring domain and computer scientists together, and allow them to effectively interact and strive toward common goals.

As much as we would like domain scientists to be involved in CPSE development they also need the appropriate methods to actively participate. Thus, from a process perspective, the development process has to better accommodate and support the direct participation of domain scientists. We need to explore software development approaches that apply user-centered and participatory design principles and techniques. We also need to provide specific analysis and design tools that are accessible to, usable by, and effective for domain scientists. For example, visual programming tools may allow domain scientists to develop applications in an intuitive and natural way.

For the third direction, we explored CPSE development from an architecture perspective. The design and development of CPSEs is influenced and constrained by the features of the underlying infrastructure Different architectural strategies seem relevant to the construction of CPSEs. A layered architecture will delineate and focus our development efforts by providing different layers of abstraction for development A component-based approach provides greater flexibility for overall integration and configurability to different domains. Desirable attributes of a CPSE architecture include modularity, reusability, and conformance to industry standards.

5.1 Future System Architectures for Collaborative Problem-Solving Environments I

Participants

Shahrokh Daijavad	Fred Johnson	Conrad Mulligan (Facilitator)
Jim Garrett	Bill Johnston	Carmen Pancerella (Moderator)
Al Geist	Olle Larsson	Marek Podgorny
Deborah Gracio	Elena Mendoza	Karen Schuchardt
Mary Anne Scott		

Introduction

All participants in this breakout session were enthusiastic and active. The resulting session was stimulating and engaging. We were tasked with the following agenda, which was rather ambitious:

> Various collaboration and problem-solving environments have sprung up in academic and commercial research and development efforts. Our goal is to combine the theories and capabilities of such environments into a unified architecture for developing CPSEs. We no longer view collaboration and problem-solving as separate attributes, but as a synergistic account of how scientists solve problems together. We seek to extrapolate and evolve current collaborative and problem-solving system frameworks to support this new concoction.

While we did not come up with a unifying architecture for CPSEs, we discussed many important issues and had some good insights into CPSEs.

We began the breakout session by having each participant list what they believed were the most important issues in building future architectures for CPSEs. This activity was motivated by a homework assignment given the previous evening asking the workshop participants to list their top three architectural issues for CPSEs. These issues are what we believe researchers will face as they try to develop a unified architecture for developing CPSEs. The list created by the group is as follows:

- There are no requirements or specifications for PSEs. In fact, PSEs are vaguely defined.
- The architecture should be easy to use. The CPSE development will be an iterative process during design and implementation.
- There should be a seamless integration of both synchronous and asynchronous tools.
- The architecture itself must be enduring and have a property of longevity. The architecture must be designed for change because technology is constantly changing and the CPSE will evolve over time.
- It is important to determine the appropriate level of commonality for all CPSEs.

- The architecture must be customizable because the CPSE will be extendable to diverse users.
- The architecture should support knowledge capture.
- The architecture should allow for heterogeneity and scalability. The architecture must allow for different participants with different levels of computer capabilities and abilities so that all participants can make the most out of their capabilities. For instance, the CPSE may include everything from users with hand-held PDAs to high-end workstations and supercomputers.
- The architecture should contain useful, appropriate language for CPSEs and should have a coupling to resource management.
- The domain scientists should be involved in the architecture and CPSE design.
- Security (i.e., access control) should be part of the underlying architecture.

We then discussed the current state of the art in both problem-solving environments and collaboration. We separated the state-of-the-art discussion into these two components because we believed very few CPSEs existed, and it was important to discuss the two components of CPSEs to get a true understanding of the current state of the art. Given our state-of-the-art lists, we then discussed the limitations of current state of the art, and the barriers and challenges that exist. These discussions are summarized in later sections of this document.

Finally, we discussed the following questions posed by the workshop organizers:

- How do we combine collaborative and problem-solving technologies in a useful way?
- What are the primary functional components of a CPSE?
- How do we accommodate extensibility (i.e., adding new collaboration and problem-solving tools)?
- How do we accommodate system-level modifications (i.e., data formats, database interfaces, application interfaces, computer interfaces)?
- How do we accommodate legacy applications?
- How should system features be organized to best support collaborative problem-solving?
- What is the boundary between a CPSE and the application code?
- What should one expect from a common CPSE development toolkit?

We discuss specific details of this breakout session in the following sections.

Important Issues

One issue that was discussed early in the session was why we need architectures to build CPSEs". We concluded that architectures enable sharing of components, resources, and services and that they facilitate integration.

We spent a lot of time discussing the nature of CPSEs. It's important in CPSEs to be able to control when to collaborate and share results with others.

One interesting observation that we made was that a PSE is very structured; i.e., a laboratory protocol can be followed in a PSE, but a CPSE is not as structured—the nature of collaboration is free-form and unpredictable. Hence, when we were asked to address the question, "How do we combine collaborative and problem-solving technologies?, we recognized that the difference between the structured nature of PSEs and the unstructured nature of collaboration makes it difficult to achieve some of the CPSE goals, such as capturing/recording information for playback. In response to this same question, we discussed the need for metadata and indexing. We agreed that most of these issues are still open research questions, and they need to be addressed in the context of CPSEs. As a group, we agreed that we need a tighter integration between problem-solving environments and collaboration.

As a response to the question, What are the primary functional components of a CPSE?, we came up with a wish list; i.e., a list of the elements we believe are found in an architecture for CPSEs. This list is as follows:

- layered approach
- a component-based approach
- workflow layer to organize and invoke services, which will be a challenge to synchronous collaboration
- lightweight architecture, with a high degree of independence in components
- Globus/Grid Forum (an example of how resource management [discovery, scheduling of computational resources] can be handled by the CPSE architecture)
- integration of access control
- extension mechanisms; e.g., some scripting mechanism, wrappers, or a way to add new tools
- a tighter integration of PSE and collaboration
- promising technology: Common Component Architecture (CCA)
- preserving the process/history in CPSE, which is necessary in a CPSE for two reasons: verifying how the solution was reached and that it provides process integrity.

In response to the question, What should be expected from a common CPSE development toolkit?, we suggested a criterion: implement "AVS" on top of CPSE; i.e., the criterion is to use a toolkit to implement workflow. We also expect widgets to implement access control and other services. Finally, in response to this question, we expect that the development toolkit will allow users to specify the following about sharing and collaborating: With whom? To what extent? Scope of sharing? Under what circumstances?

In response to the question, How do we accommodate extensibility?, we believe two issues exist. First, the extensibility of the CPSE and the extensibility of the resources exists. We also believe that a technology such as CCA leads naturally to extensibility.

In response to the question, How do we accommodate legacy applications?, we suggest treating legacy application as another resource, source code or an API to integrate it into the CPSE.

In response to the question, How do we accommodate system-level modifications (i.e., data formats, database interfaces, application interfaces, computer interfaces)?, we believe that one key issue is dealing with data formats and the need for adequate metadata. We believe that this is a difficult open research problem at this time. The breakout session on data management for CPSEs addressed most of the other important issues regarding data and databases.

In response to the question, How should system features be organized to best support collaborative problem-solving?, we believe that these issues were covered by the breakout session on resource management and we referred to the breakout summary.

Finally, in response to the question, What is the boundary between CPSE and applications?, we listed a workflow management system as an example of this boundary.

State of the Art

We separate our discussion of state of the art into two lists—one for state-of-the-art problem-solving environments and the other for state-of-the-art collaboration. Some tools/programming environments/technologies can be considered in both lists.

Problem-Solving Environments

First, we discussed the state of the art in problem-solving environments in science and engineering. We listed the following problem-solving environments as successful systems that are enhancing the work of scientists and engineers. Our list includes both problem-solving environments and tools and technologies used to create problem-solving environments.

Our list is by no means complete because problem-solving environments have been around in some form for many years, even if they are not specifically called problem-solving environments. For example, we list several engineering problem-solving environments, yet many others are developed and deployed as either commercial products or in-house solutions.

Science

Extensible Computational Chemistry Environment (ECCE), developed at Pacific Northwest National Laboratory (http://www.emsl.pnl.gov:2080/docs/ecce/index.html), is a problem-solving environment focused on computational chemistry. It has won a R&D 100 award.

The Structural Molecular Biology problem-solving environment, presented by Art Olson (http://www.scripps.edu/pub/olson-web/) at this CPSE workshop, is a problem-solving environment for facilitating computational research in structural biology. This problem-solving environment uses Advanced Visual Systems (AVS) and Python as a "glue-layer." The breakout session believed that the strong points about this environment are the scripting capabilities and the visual dataflow.

Engineering

ANSYS (http://www.ansys.com) is a Finite Element Analysis company that has started to provide other functionalities with their products.

GT STRUDL (http://shell5.ba.best.com/~solvers/gtstrudl.html) is an integrated system for modeling and computer-aided engineering for structural engineers.

The STAAD environment (http://www2.ctceng.com/ctceng/staad1.html) is also used for structural design and analysis.

SAP2000 is another integrated structural engineering environment like STAAD (http://www.csiberkeley.com).

Windchill is a commercial integrated product data management system from Parametric Technology Corporation (http://www.ptc.com). It is an engineering problem-solving environment for data acquisition and visualization.

I-deas is an integrated design and analysis environment from SDRC (http://www.sdrc.com/).

Several CAD companies, like Autodesk (http://www.autodesk.com) and Bentley Systems (http://www.bentley.com/), are also providing a base CAD system on top of which many third-party developers are developing systems to support the design and analysis of systems. These systems can be classified as problem-solving environments.

Finally, several CASE tools are in use as a software development problem-solving environment. Many of these tools can be considered CPSEs because they support multiple developers collaborating on a software project.

Tools and Technologies

The spreadsheet paradigm has been used to show the functional relationships between elements in a problem-solving environment, one of the earliest technologies for building problem-solving environments.

The visual icon programming paradigm; i.e., graphical programming language development environments, includes several tools for creating problem-solving environments. These tools include Webflow, AVS (http://www.avs.com), LabVIEW developed by National Systems (http://www.natinst.com/labview/), and Ski Run.

IDL (Interactive Data Language) is a fourth-generation language for analysis, visualization, and custom science application. It is software distributed by Research Systems, Inc. (http://www.rsinc.com).

Several component-based distributed object technologies are currently used for problem-solving environment development; e.g., Enterprise Java Beans (EJB) (http://www.javasoft.com/products/ejb/index.html) and CORBA (http://www.omg.org or http://www.corba.org).

Next, we discussed state of the art in collaborative systems. In our discussion we distinguished between asynchronous collaboration, synchronous collaboration, and hybrid systems; i.e., those systems that support both synchronous and asynchronous collaboration. The March/April 1999 issue of *IEEE Internet Computing* is devoted to collaborative computing over the Internet. Several current state-of-the-art projects, collaboratories, and collaboration technologies (both synchronous and asynchronous) are listed in this issue, some of which are included in this writeup.

Asynchronous

Certainly the most common asynchronous collaborative tool is e-mail. Discussion groups and news groups are collaborations-based on the concept of e-mail.

Lotus (http://www.lotus.com) is perhaps the most well-known and most successful company developing asynchronous collaborative products or groupware. Lotus Notes is probably the most famous of these products, and it is primarily used for office collaboration; e.g., document sharing, shared calendars, group scheduling, and e-mail.

Livelink, a product from Open Text (http://www.opentext.com), is another tool for document-centric collaboration.

IpTeam, a commercial product from NexPrise (http://www.nexprise.com), is a collaborative tool that supports product team integration in supply chains. It facilitates project management across teams, team administration, RFP negotiations, and message and document management.

Microsoft (http://www.microsoft.com) has announced a product (Platinum?) that is based on the WebDAV standard.

Electronic notebooks, like those developed under the DOE 2000 project, are state-of-the-art collaborative tools. Furthermore, other asynchronous collaborative tools are being developed and deployed to pilot projects under the DOE 2000 program.

Telemedicine and remote help desks are examples of successful collaborative systems that are deployed today.

Synchronous

Collaborative tools based on the T.120 and similar standards for data conferencing, whiteboards, shared applications, chatting, and file transmission are current state of the art for synchronous collaboration. Microsoft NetMeeting, SunForum, SGImeeting, Intel ProShare, and PictureTel LiveShare are examples of such products.

Virtual Network Computing (VNC) (http://www.uk.research.att.com/vnc/) is a remote display system that allows viewing of a computing 'desktop' environment not only on the machine where it is running, but also from anywhere on the Internet.

A variety of Internet multicast conferencing tools for audio and video-conferencing are available for use on the Mbone (http://www.mbone.com), a multicast backbone built on top of the Internet to support multiparty communication. These tools run on a variety of platforms, including machines running different flavors of Unix and Microsoft Windows.

BitRoom, a commercial product developed by Persystant (http://www.persystant.com), is an electronic commerce application integrated with telephony, extensible to multiple participants.

Several examples of remote instrumentation exist that allow for real-time collaboration with instruments. The SPARC project at the University of Michigan (http://www.crew.umich.edu/UARC/) is an example of a collaboratory that has remote instrumentation for upper atmospheric and space physics. Several collaboratories for microscopy also exist.

Habanero from National Center for Supercomputing Applications (NCSA) (http://havefun.ncsa.uiuc.edu/habanero/) is a framework or API that gives developers the tools they need to create collaborative Java applications. Tango (http://www.npac.syr.edu/tango/), developed at the Northeast Parallel Architectures Center at Syracuse University, is a similar framework, except it is a web object integration framework for collaborative systems. These systems have the following features: event-driven, entirely open, extensible, well-defined APIs, and language independent.

Finally, distance-learning applications exist that are successful examples of synchronous collaboration.

Hybrid

Some products/projects attempt to deliver both synchronous and asynchronous collaboration to users. Both Sametime from Lotus (http://www.lotus.com) and TeamWave Workplace (http://www.teamwave.com) are real-time collaboration tools for team collaboration.

Finally, several projects in DOE 2000 exist that offer both synchronous and asynchronous collaboration.

Barriers and Challenges

Next, we discussed limitations, barriers, and challenges to state of the art in problem-solving environments and collaboration, listed below:

- No real CPSE development toolkit exists.
- Nothing addresses CPSE needs.
- Human factors, such as floor control and interruption management, are not addressed.
- Scalability (one person thinks this is a nonissue), not addressed.
- Security and security policy; e.g., firewalls, not addressed.
- Need platform heterogeneity.
- Need adaptivity or quality of service at all levels (including network, data, CPU).
- Need to address support for mobility of operations; not addressed, i.e., not all users will have desktop computers.
- Direct impact on protocols and the size of clients, among other things, not addressed.
- No standard interchange formats exist.
- Need to address registry and discovery.
- Only pieces of the puzzle (of PSE and collaborations) addressed; i.e., nothing addresses a CPSE beginning to end.
- No abstraction of resource management for using/finding resources exists.
- Need to address reliability (24x7) and fault tolerance, with a provision for redundancy.
- Need to address independent data views (customization).

Recommended Actions

We believe that the following technologies are promising in the development of future architectures for CPSEs: EJB, Extensible Markup Language (XML), CCA, collaboration standards (T.120 and other flavors), and event-driven technologies (Tango and Habanero). We believe that the specific technology paths that will be driven by CPSEs, rather than some other application, are group access control, advanced sharing, and advanced collaboration technologies.

Insights and Revelations

The question of future architectures for problem-solving environments was probably the most challenging issue discussed in this workshop. In fact, this issue is why the group never really brainstormed about a specific architecture design but rather discussed issues and recommendations.

Interoperability, seamless integration, and scalable infrastructures are still basic research questions in computer science. Hence, they are open issues for future architectures of CPSEs. It would be useful for DOE to work with other agencies (e.g., NSF, DARPA) to initiate a program in CPSEs.

Personally, I believe that at this time it is probably not feasible to develop "a unified architecture for developing CPSEs" for several reasons. First of all, it is a bit premature to design a single unifying architecture. We have not had enough experience in the relatively new area of CPSE design and development, and as we continue to develop, deploy, and maintain CPSEs for scientists and engineers, we will discover many of the architectural needs that may not have been captured in this breakout session or this workshop.

Second, most of the promising software infrastructure and technologies (e.g., Globus, EJB, CCA, XML) are neither robust nor fully developed. Furthermore, many promising technologies never get fully developed before they are replaced with the latest and greatest technology. This seems to be very true in the areas of middleware, infrastructure, and integration. In the recent past, CORBA, Java, web infrastructure, Enterprise Java Beans, DCOM, and many others have attempted to solve some of the architectural issues, and each of these have been praised for their strengths and criticized for their weaknesses. When the middeware layer is such a moving target, it is difficult to build enduring architectures.

Third, no real model exists for collaboration, and thus it is difficult to integrate the concept of collaboration into software architectures in general. To date, problem-solving environments and software in general have not been designed with collaboration as a requirement. Basic research issues need to be addressed in this area. In general, collaboration is going to be a critical component of next-generation software.

Finally, many issues in knowledge creation, knowledge capture, and knowledge discovery are broad and unsolved research issues. Likewise, open issues exist in workflow for collaborative scientific and engineering problem-solving. These research questions will impact future CPSE architectures.

Many of the emerging CPSEs will likely contain reasonable solutions that address only pieces of the requirements for architectural support for CPSEs. A strong need exists to develop and deliver CPSEs in many different areas—biology, engineering, medicine, chemistry, etc.—and the employed solutions will address a subset of needs to deliver solutions to users. Unfortunately, most of these CPSE development teams will have to design and develop a CPSE architecture. During this time, researchers and developers will learn more about CPSEs and architectures. It is critical that this group of researchers publish their results and have meetings, workshops, and conferences to discuss their findings. This workshop was a good first step. We need to continually ask some of these fundamental questions about CPSE architectures, and in time, we will move toward common, unifying architectures for CPSEs.

Several emerging software technologies exist that will have an impact on future architectures for CPSEs. Mobile computing, autonomous software (e.g., intelligent agents), and active and goal-directed software will be key software technologies for future architectures for CPSEs. Digital library technology, knowledge management, and large-scale distributed resource management and information grids will have an impact on the data and computational aspects of CPSEs.

5.2 Future System Architectures for Collaborative Problem-Solving Environments II

Participants

Gary Black	Garrett Luke (Facilitator)
Don Denbo	Jim Myers (Moderator)
Larry Jackson	Michel Sanner
Mike Leuze	Eswaran Subrahmanian

Introduction

The Architecture II breakout group began with introductions and a general discussion of our approaches to architectures for CPSEs. In general, we were fairly representative of the workshop as a whole, with experience in developing and deploying both end-user systems and components/toolkits, in designing architectures, and in analyzing user requirements and processes across a wide range of scientific disciplines. As part of the discussion, we reviewed the list of architecture components developed by the whole CPSE workshop group as part of the previous evening's "homework" assignment. Not surprisingly, we all agreed with the components listed (e.g., directory services, security, etc.).

However, when we sought to add a level of detail to the description of these components, differences in the immediate requirements from the various scientific disciplines represented became readily apparent. The size of communities needing to share resources ranged from two people to millions. The need for specific security services, e.g., authentication, ranged from minimal to critical. The amount of data shared by different communities varied by many orders of magnitude. One size would clearly not fit all. This direction led the group to question the goal that seems to be implicit in the phrase "CPSE architecture"—that one set of CPSE components and services could be defined and that one national/global infrastructure could/should be built to support CPSEs across science and engineering. We also asked the related question —If this should not be a goal, what should be?

Our discussion of this issue included technical, scientific/domain, economic, and political/organizational considerations. On the technical side, we noted the relative immaturity and rapid advancement of many of the basic technologies mentioned at the workshop, e.g., public key certificates, DCOM, XML, directories, etc. We also noted the active research within the community on higher level services, e.g., security, data/metadata management, and in scaling these services beyond the current state of the art to higher speeds, larger numbers of users, more data sets, larger data sets, etc. The necessity of working with legacy applications—monolithic programs that may include their own security, data management, and other services, with few if any ways to link to external services—was also noted, as was our relative lack of influence on their evolution (i.e., they are commercial products and/or science is not the major market for them). In terms of science, the group noted that while CPSEs are/would be seen as useful tools in nearly all disciplines, the near-term needs/overall drivers in various domains differ greatly (collaboration, access to high-end computing or large data stores, process automation, etc.). Additionally, while cross-disciplinary collaboration via CPSEs would be useful, the state of practice is still islands of interoperability. Both of these observations imply the possibility for net negative effects on researchers' productivity in attempting to move toward a common CPSE infrastructure. For example, productivity gains from integrating domain applications into a CPSE could easily be offset by losses from added complexity in processes (requiring users to obtain a Grid security certificate when only local resources are used), added infrastructure (using directory services and petabyte-capable data stores on small projects), and reduced flexibility in upgrading services, etc. The economic aspects of this possibility—that the cost/benefit ratio of standardizing a CPSE infrastructure, or standardizing too early, is lower than alternatives such as developing domain-specific frameworks or simply investing in new science—were also discussed and seen as a major reason for lack of user support for CPSE development. Finally, the political/organizational difficulties of supporting common infrastructures were noted. Specifically, the group discussed the difficulties in funding a transition from prototype to production software, the change in culture needed to make that transition, the concerns about relying on infrastructure whose direction is controlled by a third party (analogous to the discussions about Sun owning Java), and concerns about relying on infrastructure funded by short-term grants.

Together, these observations suggested that planning for and building a global CPSE infrastructure is not the most effective path toward advancing CPSEs and realizing their benefits in science and engineering programs. Instead, our group discussed a more evolutionary approach, as follows.

Continue:

- Developing general/evolving CPSE architecture concept and component/service interfaces
- Researching advanced CPSE concepts and components/services
- Developing prototype CPSE components/services
- Testing/evaluating prototype components/services and architecture concepts in collaborating domain CPSEs.

Start/Continue:

- Developing/buying production CPSE components/services (based on COTS technologies) in conjunction with specific CPSE projects
- Developing CPSE community resources—lessons learned, software and project descriptions, mentoring and training opportunities, etc.
- Promoting CPSE-ready science applications—applications that may be built to run stand-alone today, but are designed with CPSE components/services in mind, so that a minimum amount of effort will be required to swap an internal component for a shared CPSE component if/when the science benefit justifies it. The move towards components can be partially justified by increased code maintainability. Being CPSE-ready would be an added value if the CPSE component/ service "model" were followed. Both of these benefits could be promoted through outreach, training, publication of the CPSE concept and service interfaces, and sharing component code.

- Investigating translation/glue technologies such as scripting, schema translation, flexible data formats and schema languages, data translation services, etc.
- Understanding the communities we are working with and their research processes. User interfaces and high-level tools that allow researchers to make effective use of CPSE services is a critical but not yet well-understood task in creating effective CPSEs. Since technologies and processes co-evolve, we clearly need to work closely with domain researchers to make progress (and identifying ways we can make incremental, cost-effective improvements in their lives is a great way to do this!)

Don't:

- Try to harden/productize global CPSE components without domain support. If such work isn't being supported by the domains (after the component has been prototyped, its value demonstrated, and an evaluation/pilot implementation done to identify what work is needed to move to production), then it is probably not cost effective. (If we are truly working to support researchers doing science and engineering, we have to trust their judgement.)
- Force global standards across domains that do not currently need to interact. Integration/translation is becoming cheaper and easier all the time; implementing technology that is too advanced (again, beyond the proof-of-concept, proof-of-value stage) in anticipation of future needs has an opportunity cost that should be accounted for.

In the concluding moments of the session, the group pondered how the nature of CPSE infrastructure might differ from that of other national infrastructures (e.g., roads, railroads, telephones, and the Internet) and if this comparison could highlight why we felt a national CPSE infrastructure was premature and perhaps not necessary. One such analogy is explored in the "Insights" section below.

Important Issues

- CPSE services and standards are and will continue to evolve rapidly.
- The benefits of CPSEs are needed now to support the work of existing groups/disciplines.
- Agreement on CPSE architecture concepts and interfaces is more important for the community than a globally supported implementation.
- Opportunities to interact/exchange ideas with other CPSEs developers are very valuable in reaching/maintaining/evolving a common CPSE architecture concept.

State of the Art

The group did not create a list of current work.

Barriers and Challenges

- Cost of membership: the cost of moving a new application into a CPSE and deploying it to and maintaining it for existing users must be less than the cost of alternatives.
- The development and maintenance of a common architecture requires coordination; i.e., this effort needs to be managed by a permanent, funded organization accepted across the community. Such an organization does not currently exist.
- Current agreement about the specification and requirements for various architecture components ends with the following description: everyone wants a security service, but everyone wants different specs in terms of level of security, scalability, etc.
- CPSE architecture service implementations probably need to be commercial or open source; e.g., not controlled by small groups and subject to the vagaries of research funding, to be acceptable to the CPSE community as a whole. Multiple implementations would be best.
- Is the cost of a standard implementation of an architecture service higher than developing bridges/adaptors? (If we need standards now, why don't bridges and adaptors already exist?)

Recommended Actions

- Promote modular design of scientific software.
- Consider concepts of external security, events, data/metadata, methods as key!
- Promote scripting to link scientific components.
- Think in terms of multiple CPSE target groups: component/model developers, component/model integrators, users of integrated applications.
- Support discussion of CPSE approaches across disciplines through workshops, journals, web, etc.
- Promote development of basic security, resource discovery, and remote data access services as incentives to a scriptable component approach.
- Promote research end-user scripting/visual programming paradigms.
- Promote research workflow display and dynamic end-user workflow definition/modification mechanisms (i.e., reusable process templates, whole process capture, workflow pruning, and annotation).
- Develop component architecture test-beds WITHIN centrally managed projects/organizations.

Insights and Revelations

Consider a flawed analogy between CPSEs and the early days of railroads. In both cases, small incompatible solutions sprang up. In both cases, people with vision realized that interoperablility leads to vastly enhanced service. In the case of railroads, the solution was to create a national "railroad architecture" with standard track gauge. By analogy, we should build a common CPSE architecture. However, critical differences exist in these two cases.

Railroad cars and engines must be designed for a single-gauge track. CPSE applications can be developed to support multiple gauges (e.g., security services, directories) that require only slightly more than retooling for a single standard.

In railroads, long-haul traffic was the most lucrative. In CPSEs, we have already moved resources geographically and organizationally to make high-value local interactions. At least initially, the interactions between groups separated by large geographical, organizational, or disciplinary distances across a common CPSE architecture are by definition not cost-effective by other means.

In railroads, the choices for track gauge were known. In CPSEs, only some of the choices are known, and we expect many infrastructure standards to emerge from other communities; e.g., e-commerce. Further, we expect the standards to evolve rapidly.

If the costs to build and maintain a common CPSE architecture are significantly greater than to develop multiple local architectures, it may be wisest to simply develop applications that are compatible with a range of architectures. Second, because CPSE interactions within a discipline or organization have high value, making compromises in architecture to support cross-disciplinary CPSEs may cause an overall reduction in value. Third, if standards are still evolving, trying to build and maintain a common architecture may simply slow the rate at which individual communities can adopt and take advantage of new technologies. While the comparison of railroads and CPSEs certainly has limits, it is instructive to note that they differ at a fairly high level with regards to the strength of drivers toward common infrastructures. Thinking in terms of delivering the most value to users questions the need for a common CPSE architecture, and instead focuses discussion on the development of flexible CPSE applications and a range of conceptually compatible architecture implementations.

Homework Assignments

Workshop participants were given homework assignments at the end of each day. The homework was collected and compiled, and the results provided in this section.

Homework for June 29th

Identify three strategies for designing CPSEs.

- Focused, evaluated test-beds (with control groups?)
- Focus on science, not computer science
- Plug and play modules - modular environments that can be customized
- Hierarchical - build components first for general CPSEs that can be shared across disciplines
- Involve end users to develop requirements, design and refinement/feedback
- Plan for customization—unique interfaces for users, different workflow for different classes of users
- Consider deployment issues up front—24x7 (with little admin involvement), training, installation, productionization
- Collect requirements from domain scientists
- User interface usability
- Modular design
- Identify disciplines where other collaboratories or PSEs exist and add the other piece
- Keep big picture in mind
- Keep domain experts involved
- Prototype early
- Obtain a user group willing to work closely with the computer science group on an iterative process of defining requirements, designing, prototyping, evaluating, redefining requirements, redesigning
- Create components for commonly used services, but create a layer on top of these services that is tailored for the specific user domain
- Use adequate computing power at user sites to provide highly effective user interfaces. Currently being developed (voice and gesture recognition, dynamic 3D visualization)
- Involve an equal number (or nearly equal) of domain experts from the requirements phase to specs to design
- Plan on throwaway prototype systems early on to illicit feedback and gain experience to get it right (or much better) the second time
- Design to get components available to users incrementally rather than one big release that takes years
- Agent technologies
- Abstraction-based layer over distributed resource management layer
- Close cots—R&D interaction/bridge (e.g. CORBA - GLOBUS)
- Bottom-up, modular framework, using as much existing software to begin with as possible
- Stringent security policy right from the start
- Plan for a world with heterogeneous tools with often noncompatible features
- Techniques for observing, capturing, and representing expert problem solving strategies to build workflow representations
- Participatory design techniques tailored to the design of CPSEs
- Better shared understanding of capabilities and purpose of currently available "scientific production quality" software tools

- Gathering the requirements
- Defining/finding the right infrastructure
- Designing applications and tools on top of the infrastructure
- User-centered iterative design and development and deployment process
- Reliance on testbeds
- Web-based, low-entry SST
- Find successful collaboration and integrate in PSE environment
- Create portable testbed to demo environment
- Invest in university centers of excellence for training in collab systems
- CORBA and its enabling services, e.g., notification, trading, transaction, security
- Enterprise JAVA beans: relatively new with limited services and constrained to JAVA world
- The main limitation of above approach is that services are provided at the "host" level. To have a significant performance impact, enabling services need to be pushed down to the network layer; e.g., "active networks," MIT Simulation on TCP and reliable multicast over active networks, etc.
- Extensible
- Scalable
- Low entry requirement
- Listen - PROTOTYPE - Evaluate
- Derive CPSE from data/workflow/model
- "LEGO" toolkit model user creates own
- Traditional top-down/bottom-up: listen to user, give them what they want, phase in what they need
- Develop them around a high level, preferably platform independent and interpreted language that will serve as "glue" to connect components and facilitate their interoperation
- Develop modular components that can be reused across domains
- Provide programming capabilities to automate tasks and allow for rapid prototyping of variations of the available computational methods and/or new combinations of these methods.

Homework for June 29th

Identify three components for future CPSE architectures.

- Open Metadata Registry
- Global Event System (publish/subscribe)
- Global Scripting
- Ease of use - user interface and visualization
- Timely use - fast data delivery and intelligent delivery
- Communication - collaborative tools for interacting with other scientists
- Security - access control at fine grain, authentication of user
- Extensibility and ease of adding new tools and true tool integration
- Data management - archiving, locating, searching, querying, managing large data sets, seamless data interchange
- Distributed object middleware (e.g., CORBA, RMI)
- Dataserver/subsetting/data translation
- Metadata collection/indexing/searching
- Chat room - collaboration
- Visualizer/data browser
- User extensible interface to models
- Distributed architecture
- Componentized
- Language independent
- Component for high-performance 3D dynamic, shared visualization (both synch and asynch)
- Components for persistent virtual space
- Components for resource discovery, reservation, and utilization
- Baseline components independent of application domain for components such as data management, resource management, collaboration tools, experiment management, job launching, job monitoring.
- Free of low-cost nonproprietary run-time third party apps. No commercial dbs or at least an architecture that allows different db management systems for baseline support where one choice is no cost to user
- Must be web-based to allow for collaboration and cross platform support as well as all the typical benefits of web based architectures
- Java and JAVA Beans
- Abstraction/model-driven design (e.g., UML-based models/patterns design)
- Ontologies
- A "base module" with security and a negotiation protocol ("if I send you an AVI file, can you handle it?" "No, but I can handle MPEG")
- "plug-n-playable" converters ("oh, I can't handle AVI, but I have an AVI-to-MPEG converter, so I should accept this file")

- Data management filters ("I have a slow connection. Therefore, I shouldn't get the high-resolution image but a lower ...")
- Visual programming capabilities to support scientist-driven specification of workflow (like AUS) Extensible object library
- Workflow/problem solvers representation objects/data structures
- Remote job launching/monitoring capabilities
- Combined async/sync architecture
- Components addressing security
- Support for pervasive computing clients (e.g., PDAs)
- Data servers including search engines
- Computational clusters
- Visualization tools and services
- Notebook
- Shared windows
- Immersive telepresence
- Visual programming language for constrained object integration
- Automatic GUI generation from an advertised service and its corresponding properties; e.g., based on control parameters of an instrument, their interrelationship and functional use conditions generate a GUI and then let user edit their final placement
- Learning, prediction, and clustering of use conditions (content data in general) for real-time feedback to user. The intent is hint the user for abnormal access (illegible), error discovery, and how-to guided tour. For example, how is this protocol for imaging an inclusion different from those in data bank (data bank maintains a "model" system)? Is it possible to contrast a particular protocol with model system for consistency checking (reinforcement) or criteria of new access pattern?
- Integration of virtual reality; e.g., is it possible to represent the front panel of an instrument control in the virtual reality and thus bypass GUIs that are inherently difficult to use?
- Data discovery
- Access control
- Data management
- Support for data management (e.g., storage, retrieval, sampling, searching, caching) and data representation (structures to reflect hierarchical organization of data, relationships between objects, etc.)
- Visualization toolkit providing high-level visualization capabilities of 2D and 3D data. These tools should be reusable across domains and most importantly extensible and customizable by the user community
- Hooks allowing for the integration of legacy code as well as fast-evolving computational methods originating for research laboratories
- Group collaboration tools (dist mtgs, lectures, panels, site visits)
- Security - transparent
- Network Flow Management (cpus, (illegible) and mgmt, route mgmt)
- Security
- Self-describing data
- Resource management.

Homework for June 30th

What should be our next step following the CPSE workshop?

- A project involving the integration of many projects/lessons learned to date; funded and with a deployment community that cares
- We need to get some money
- Involve universities and other federal agencies
- Should be able to create interest groups along different architectural issues (e.g., visualization) to provide a means for community sharing and inference of standards (emergent)
- Each of the organizations may want to look into their own collaborative practices and share this information, which can be the basis from which to identify common problems and issues beyond disciplinary boundaries
- Need a conference/workshop where researchers can get together to discuss CPSE issues; this was a good first step-need to expand
- Possible BOF or meeting at another established conference (e.g., SC).
- Establish website to point to CPSEs worldwide
- Create e-mail list to discuss/announce CPSE topics/events
- Focus by some group on techniques for capturing, documenting, modeling scientific problem-solving processe
- Implementation and requirements elicitation of (1) collaboration that can benefit with the introduction of PSE tools, and (2) PSEs that could benefit with collaborative tools
- Use the workshop report as a basis to advocate support and coordination for CPSEs within organizations, disciplines, programs and across agencies
- Follow-up workshop(s) to discuss design strategies and architecture in more depth
  Post the first draft of the report on the web and open an e-mail list, Lamda Moo, or similar place (room) to discuss and/or modify the report
- Post success stories on the website.

Hanging Issues

- Some kind of discussion forum (threaded?), seeded with outlines from the conference
- What science opportunities are prime for CPSE implementation
- How to create interest groups and getting people to share information about their own collaborative processes
- User interface, human factors issues
- Importance of customizing CPSEs to support a range of uses/capabilities
- How to deploy/maintain CPSEs
- Should this workshop be an annual event and expand to include engineering applications
- Way to keep discussion going (e.g., mailing lists)

Next Step Discussion, July 1st

Notes from "What should be our next step following the CPSE workshop?" Discussion

- This is a timely area
- BOF meetings desirable, but many meetings are domain-specific
- Crucial for CPSE community to identify itself
- Needs to be an event that brings CPSEs to attention of people-multiagency
- Need place to go to learn about current work (tutorial being run on GLOBUS and TANGO)
- Need place to contribute information
- Where to publish
- Electronic journal
- Abstracting important
- Respect important
- IEEE internet camp
- Meetings
- Electronic index that points to journals
- Index sites, bring lists together
- Threaded discussion list for subtopics
- Who's missing from this group grow community
- Success stories
- Programs/funding for multidisciplinary research
- Other program managers/other agencies who are interested
- "Awards" for significant accomplishments
- See Smithsonian S&T awards
- Publicize awards
- Special issues - CPSEs - IEEE Computer
- Electronically publish new articles (refereed)
- Summarize "issue" for science
- CPSE Resource
- CPSE.org
- Place where you could enter some CPSEs
- Research issues may include devices/environments other than desktops
- Report to HPN Action Team
- Summary to interagency people.

Workshop Program

Tuesday, June 29, 1999

7:30	Continental Breakfast/Registration	12:45	Technical Presentation - Edward Chow, JPL NASA
8:30	Welcome to the Workshop - Ray Bair, PNNL Welcome from the DOE Sponsor - Mary Anne Scott, DOE OS-MICS Introduction - Debbie Gracio, PNNL Logistics - Suzanne Dunn, McNeil Technologies, Inc	1:30	Purpose of Breakouts - Jim Myers, PNNL
		1:45	Breakouts Distributed Resource Management Data Management Scientific Collaboration and Research
9:15	Questions and Answers	3:15	BREAK
9:30	Technical Presentation - Farnam Jahanian, University of Michigan	3:30	Resume Breakouts
10:30	BREAK	5:00	Reconvene - Wrap-up (Q/A, Receive Homework)
10:45	Technical Presentation - Bill Johnston, LBL	5:30	ADJOURN
11:45	LUNCH		

Wednesday, June 30, 1999

7:30	Continental Breakfast	1:00	Purpose of Breakouts - Jim Myers, PNNL
8:30	Breakout Summaries (25 minutes each - 15 presentations, 10 questions) Distributed Resource Management Data Management Scientific Collaboration and Research	1:15	Breakout Strategies for Designing CPSEs Future System Architectures for CPSEs
		3:00	BREAK
9:45	BREAK	3:15	Resume Breakouts
10:00	Technical Presentation - Tom Furness, University of Washington	5:00	Reconvene - Wrap-up (Q/A, Receive Homework)
11:00	Technical Presentation - Art Olson, Scripps	5:30	ADJOURN
12:00	LUNCH		

Thursday, July 1, 1999

7:30	Continental Breakfast	10:25	Technical Presentation - Shahrokh Daijavad, IBM Research
8:30	Breakout Summaries (25 minutes each - 15 presentations, 10 questions) Strategies for Designing CPSEs Future System Architectures for CPSEs	11:25	Workshop Summary, Open Discussion - Jim Myers, PNNL
		11:50	Next Steps
		12:00	ADJOURN
10:10	BREAK		

Workshop Participants

Ulrike Axen
Washington State University
School of EECS
PO Box 642752
Pullman, WA 99164-2752
Phone: 509-335-4961
Fax: 509-335-3818
E-mail: axen@eecs.wsu.edu

Ray Bair
Pacific Northwest National Laboratory
PO Box 999, Mail Stop K8-91
Richland, WA 99352
Phone: 509-376-7939
Fax: 509-376-0420
E-mail: raybair@pnl.gov

Arindam Banerji
Hewlett-Packard
10450 Ridgeview Court
Cupertino, CA 95015
Phone: 408-343-6106
Fax: 408-343-6777
E-mail: AXB@HPL.HP.com

Chaitan Baru
San Diego Supercomputer Center
9500 Gilman Drive
La Jolla, CA 92093-0505
Phone: 858-534-5035
Fax: 858-822-0906
E-mail: baru@sdsc.edu

Gary Black
Pacific Northwest National Laboratory
PO Box 999, Mail Stop K1-96
Richland, WA 99352
Phone: 509-375-2316
Fax: 509-375-6631
E-mail: gary.black@pnl.gov

Kevin Brunner
Silicon Graphics Incorporated
2011 N. Shoreline Blvd.
Mt. View, CA 95023
Phone: 650-933-6874
Fax: 650-932-6874
E-mail: brunner@sgi.com

Henri Casanova
University of California, San Diego
9500 Gilman Drive, Dept. 0114
La Jolla, CA 92093-0114
Phone: 858-534-5913
Fax: 858-534-7029
E-mail: casanova@cs.ucsd.edu

George Chin
Pacific Northwest National Laboratory
PO Box 999, Mail Stop K1-96
Richland, WA 99352
Phone: 509-375-2663
Fax: 509-375-3641
E-mail: george.chin@pnl.gov

Edward Chow
NASA/Jet Propulsion Laboratory
4800 Oak Grove Drive, Mail Stop 126-201
Pasadena, CA 91109
Phone: 818-393-3854
Fax: 818-393-3003
E-mail: edward.chow@jpl.nasa.gov

Shahrokh Daijavad
IBM
T.J. Watson Research Center
30 Sawmill River Road
Hawthorne, NY 10532
Phone: 914-784-7663
Fax: 914-784-6498
E-mail: shahrokh@us.ibm.com

Donald Denbo
NOAA/PMEL
NOAA/PMEL/OCRD
7600 Sand Point Way, NE
Seattle, WA 98115
Phone: 206-526-4487
Fax: 206-526-6744
E-mail: dwd@pmel.noaa.gov

Terrence Disz
Argonne National Laboratory
9700 South Case Avenue
Argonne, IL 60439
Phone: 630-252-7831
Fax: 630-252-5676
E-mail: disz@mcs.anl.gov

Lori Freeland
Pacific Northwest National Laboratory
PO Box 999, Mail Stop K1-96
Richland, WA 99352
Phone: 509-375-2412
Fax: 509-375-6631
E-mail: lori.freeland@pnl.gov

Thomas A. Furness III
Human Interface Technology Laboratory (HIT Lab)
215 Fluke Hall, University of Washington, Box 352142
Seattle, WA 98195-2142
Phone: 206-685-8626
Fax: 206-543-5380
E-mail: tfurness@u.washington.edu

Jim Garrett
Carnegie Mellon University
Department of Civil and Environmental Engineering
ICES
Pittsburgh, PA 15213-3890
Phone: 412-268-5674
Fax: 412-268-7813
E-mail: garrett@cmu.edu

Al Geist
Oak Ridge National Laboratory
PO Box 2008
Oak Ridge, TN 37831-6367
Phone: 423-574-3153
Fax: 423-574-0680
E-mail: gst@ornl.gov

Debbie Gracio
Pacific Northwest National Laboratory
PO Box 999, Mail Stop K1-96
Richland, WA 99352
Phone: 509-375-6362
Fax: 509-375-6631
E-mail: gracio@pnl.gov

Nenad D. Ivezic
Oak Ridge National Laboratory
PO Box 2008
Bldg. 6010, Mail Stop 6414
Oak Ridge, TN 37831-6414
Phone: 423-241-4351
Fax: 423-241-6211
E-mail: z5I@ornl.gov

Larry Jackson
University of Illinois/NCSA
152 CAB
605 E. Springfield Avenue
Champaign, Illinois 61820
Phone: 217-333-0949
Fax: 217-333-5973
E-mail: jackson@ncsa.uiuc.edu

Farnam Jahanian
University of Michigan
Dept. of EECS
Ann Arbor, MI 48109
Phone: 734-936-2974
Fax: 734-747-9482
E-mail: farnam@eecs.umich.edu

Fred Johnson
U.S. Department of Energy
SC-31
19901 Germantown Road
Germantown, MD 20874-1290
Phone: 301-903-3611
Fax: 301-903-7774
E-mail: fjohnson@er.doe.gov

Bill Johnston
NASA Ames Research Center/LBNL
Mail Stop 258-5
Moffett Field, CA 94035
Phone: 650-604-4365
Fax: 510-486-6363
E-mail: wejohnston@lbl.gov

Alan Karp
Hewlett-Packard
49EL-FR
10450 Ridgeview Court
Cupertino, CA 95015
Phone: 408-343-6109
Fax: 408-343-6777
E-mail: alan_karp@hp.com

Carl Kesselman
USC Information Sciences Institute
4676 Admiratty Way, Suite 1001
Marina Del Rey, CA 90292
Phone: 310-822-1511 x338
Fax: 310-823-6714
E-mail: carl@isi.edu

Richard Kouzes
West Virginia University
866 Chestnut Ridge Road, Room 211
Morgantown, WV 26506-6216
Phone: 304-293-8281
Fax: 304-293-7498
E-mail: rkouzes@wvu.edu

Olle Larsson
Argonne National Laboratory
9700 S. Cass Avenue, Bldg. 221
Argonne, IL 60439-4844
Phone: 630-252-1151
Fax: 630-252-5986
E-mail: larsson@mcs.anl.gov

Mike R. Leuze
Oak Ridge National Laboratory
PO Box 2008
Bldg. 6010, Mail Stop 6414
Oak Ridge, TN 37831-6414
Phone: 423-574-5200
Fax: 423-241-6211
E-mail: rxr@ornl.gov

Stu Loken
Lawrence Berkeley National Laboratory
1 Cyclotron Road
Mail Stop 50B-4230
Berkeley, CA 94720
Phone: 510-486-7474
Fax: 510-486-4300
E-mail: loken@lbl.gov

Anita LoMonico
National Cancer Institute - Informatics
6130 Executive Blvd. EPN 211
Bethesda, MD 20892
Phone: 301-435-6134
Fax: 301-402-1037
E-mail: lomonica@exchange.nih.gov

Cheryl Marks
NIH/National Cancer Institute
6130 Executive Blvd. EPN 501
Bethesda, MD 20892-7381
Phone: 301-435-5226
Fax: 301-496-8656
E-mail: cm74v@nih.gov

Richard May
Battelle
4337 15th Avenue, NE #114
Seattle, WA 98105
Phone: 206-616-1481
Fax: 206-543-5380
E-mail: richard.may@pnl.gov

Richard McGinnis
NASA Langley Research Center
2 South Wright Street
Hampton, VA 23681-2199
Phone: 757-864-6893
Fax: 757-864-9882
E-mail: r.s.mcginnis@larc.nasa.gov

Elena Mendoza
Pacific Northwest National Laboratory
PO Box 999
Richland, WA 99352
Phone: 509-376-9252
Fax: 509-376-0420
E-mail: elena.mendoza@pnl.gov

Jim Myers
Pacific Northwest National Laboratory
Mail Stop K8-91
3335 Q Avenue
Richland, WA 99352
Phone: 509-376-9558
Fax: 509-376-0420
E-mail: jim.myers@pnl.gov

Arthur Olson
The Scripps Research Institute
10550 N. Torrey Pines Road
Mail Drop MB5
La Jolla, CA 92037
Phone: 858-784-9702
Fax: 858-784-2860
E-mail: olson@scripps.edu

Carmen Pancerella
Sandia National Laboratories
PO Box 969, Mail Stop 9012
Livermore, CA 94551
Phone: 617-630-0316
Fax: 617-294-1230
E-mail: carmen@ca.sandia.gov

Bahram Parvin
Lawrence Berkeley National Laboratory
Mail Stop 50B-2239
Berkeley, CA 94720
Phone: 510-486-6203
Fax: 510-486-6363
E-mail: b_parvin@lbl.gov

Marek Podgorny
NPAC @ Syracuse University
3-206 CST 111 College Place
Syracuse, NY 13244
Phone: 315-443-1722
Fax: 315-443-1973
E-mail: marek@npac.syr.edu

Kimberly D. Rasar
Oak Ridge National Laboratory
PO Box 2008
Bldg. 6010, Mail Stop 6414
Oak Ridge, TN 37831-6414
Phone: 423-574-5200
Fax: 423-241-6211
E-mail: kb2@ornl.gov

Michel F. Sanner
The Scripps Research Institute
10550 N. Torrey Pines Road, MB5
La Jolla, CA 92037
Phone: 858-784-2341
Fax: 858-784-2860
E-mail: sanner@scripps.edu

Paul Saxe
SciCo Inc.
5031 Palermo Drive
Oceanside, CA 92057
Phone: 760-724-4563
Fax: 760-724-5526
E-mail: psaxe@scicousa.com

Karen Schuchardt
Pacific Northwest National Laboratory
PO Box 999, Mail Stop K1-96
Richland, WA 99352
Phone: 509-375-6525
Fax: 509-375-6631
E-mail: karen.schuchardt@pnl.gov

Mary Anne Scott
U.S. Department of Energy
19901 Germantown Road
Germantown, MD 20874-1290
Phone: 301-903-6368
Fax: 301-903-7774
E-mail: scott@er.doe.gov

Eswaran Subrahmanian
Carnegie Mellon University
HBH 1209, Institute for Complex
Engineered Systems
5000 Forbes Avenue
Pittsburgh, PA 15217
Phone: 412-268-5221
Fax: 412-268-5229
E-mail: sub@cs.cmu.edu

Ravi Subramanya
Pittsburgh Super Computer Center
4400 5th Avenue
Pittsburgh, PA 15213
E-mail: ravi@psc.edu

Vaidy Sunderam
Emory University
Department of Math and Computer Science
Emory University
Atlanta, GA 30322
Phone: 404-727-5926
Fax: 404-727-5611
E-mail: vss@mathcs.emory.edu

Dave Thurman
Pacific Northwest National Laboratory
PO Box 5395
Seattle, WA 98105-0395
Phone: 206-528-3552
Fax: 206-528-3552
E-mail: dave.thurman@pnl.gov

Val Watson
NASA Ames Research Center
Mail Stop 258-2
Moffett Field, CA 94035-1000
Phone: 650-604-6421
Fax: 650-604-4377
E-mail: vwatson@mail.arc.nasa.gov

Christine Yang
Sandia National Laboratories
PO Box 969, Mail Stop 9012
Livermore, CA 94551-0969
Phone: 925-294-2016
Fax: 925-294-1230
E-mail: clyang@ca.sandia.gov