National Science Foundation
Shared Cyberinfrastructure Directorate

Workshop Report

Planning for Cyberinfrastructure Software

www.nsf.gov/cise/sci/ci_workshop/index.jsp

February 2005

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

The NSF Workshop on Planning for Cyberinfrastructure Software was conducted on October 5-6, 2004 in Arlington, Virginia. The workshop was timely both in terms of gaining a better understanding the relationship of software and cyberinfrastructure, and the importance of addressing critical issues and programs to support the software required for successful deployment of cyberinfrastructure.

The workshop organizers want to especially thank all the attendees for enthusiastically participating in the development of white papers and formal presentations, leading sessions, participating in panels and presenting alternate views and materials. Since the attendees had thought or written responses to the series of workshop questions before arrival, most of the workshop time was focused on discussion and consensus building which resulted in a series of recommendations and guiding principles which will be most useful to NSF and other Federal Agencies.

A special thanks to the Session Leaders who also served as editors for the Final Report: Dan Atkins ,Randy Butler, Stu Feldman, Ian Foster, Carl Kesselman, Miron Livny, Rick Stevens. Their input and suggestions were most helpful and appreciated. Also thanks to Mary Spada for workshop logistics and editing the Final Report.

Alan Blatecky, Dave Messerschmitt

Executive Summary

It is widely recognized that cyberinfrastructure (CI) technologies will transform how scientific investigation and collaboration are conducted in coming decades.[1] NSF has significant efforts underway both to advance those technologies and to help researchers use them. Yet one critically important CI component, software, has received relatively little attention. Historically, NSF has typically assumed that software will emerge either from external sources or as a side effect of research projects. This assumption is no longer tenable: such "found" software often does not provide needed functionality, robustness, and/or interoperability, and moreover, is not maintained. Well-engineered and useful software is essential to the creation and application of CI. In other words, CI software is infrastructure that must be acquired or constructed, stabilized and deployed to meet specific CI objectives if the promise of CI is to be realized.

This recognition that software is infrastructure leads to many questions. It is not clear what software is required, how robust that software should be, how it should be developed and supported, or how software can most effectively flow between the research community and commercial companies. In addition, there has not been adequate recognition that CI changes the rules and foundations of the research endeavor across much of NSF. CI software is a new class of artifact that should be the target of explicit design, construction, study, and evolution. NSF must develop new approaches and programs aimed at producing required CI technologies and software in support of science.

To address these CI software issues, and to begin to formulate recommendations, a two-day invitation-only workshop was conducted in Washington, DC on October 5-6, 2004. Twenty three leading national CI software developers and architects and 15 Federal Agency program officers (from NSF and other agencies) attended and participated.

Principal Recommendations

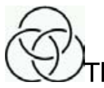
- NSF should establish new programs to support the *development* of CI software. These programs must provide for the development and sustenance of a critical mass of the necessary software engineering expertise.
- NSF should establish new programs to support the *deployment* of CI software. These programs must support multidisciplinary projects of substantial size and duration, so as to engage major scientific communities in the application and adoption of CI software within the context of CI goals.
- To accomplish these new programs in development and deployment, NSF should specifically:
    - Establish 2-3 large efforts, each with a 5-10 year duration, to develop, integrate, and support core CI software in key technology areas.
    - Establish 4-5 application software teams, each with a 5-year duration, to address the requirements of specific scientific domains, in a multi-disciplinary and cross-directorate approach, with shared funding, responsibility and review.

[1] As well-documented in the Report of the National Science Foundation Blue-Ribbon Advisory Panel on Cyberinfrastructure, Revolutionizing Science and Engineering throrugh Cyber-infrastructure (aka "The Atkins Report"). See http://www.communitytechnology.org/nsf_ci_report/

    - Continue the current NMI program and its efforts focused on spurring innovative software development.
- NSF should establish an umbrella effort to provide *coordination* of software development activities across domains and directorates, addressing in particular the vital topics of international and inter-agency coordination.
- NSF should take steps to encourage, empower and reward a software deployment sub-culture and a supporting applied research and engineering activity in the computer science community.
- CI software development and deployment should follow an open source model, build on open standards wherever possible, welcome community involvement, and encourage and accommodate industry.
- NSF should establish a formal external review process for all development and deployment projects, to encompass both specific deliverables and surveys of community usage and adoption.

General Guiding Principles for Cyberinfrastructure Projects and Activities

- **CI requires substantial software. CI software research is not the same as CI software development. The production of high quality CI software requires explicit design and development. These design and development tasks are expensive and require specialized expertise.**
- There is tremendous value to finding commonality in CI architecture so as to maximize use/reuse of software across diverse communities. However, finding commonality is hard, and requires focused and continued effort.
- The tasks of creating and deploying CI software are challenging tasks worthy of study in their own right. To this end, NSF should empower and reward a software development and deployment sub-culture and a supporting applied research activity in the computer science community.
- NSF must address the need to retain expertise and provide a career growth path for CI software developers and providers within the academic community.
- NSF SCI needs to develop programs that explicitly support CI software deployment and adoption, with the main drive coming from domain scientists from other directorates.
- Successful deployment requires not just research and development but also integration, testing, and interoperability. Successful adoption requires packaging, distribution, customization, and support services for end users.
- CI projects and programs should be multidisciplinary (e.g. Borromean Rings)[2] whenever possible, and CI research and development needs to be linked to deployment to ensure that research and development work is driven by real end-user requirements (e.g., Pasteur Quadrant Research Model -see page 8: **Goals in Gathering Inputs Session**).

[2] This design of three interlinked circles is known as the Borromean Rings. The three rings taken together are inseparable, but remove any one ring and the other two fall apart. Because of this property, they have been used in many fields as a symbol of strength in unity.

- CI software projects need to adopt and follow an open source model to encourage community participation. The focus of software work should be on wide usage and impact rather than blind adherence to standards and reference models
- CI projects require significant coordination at multiple levels: across projects, disciplines, directorates, and agencies, and with comparable international efforts.
- CI software development projects should engage commercial software providers to help identify value propositions, to help target commercial software to meet NSF community needs, and to facilitate technology transfer to industry where appropriate. To this end, NSF needs to encourage explicit discussion of what requirements can and cannot be met by commercial software.
- Because developing and applying CI is new and difficult, more formal frequent review is desirable for purposes of guidance, information gathering, and evaluation throughout CI project execution, including decision points for further support, not just at project end.

Workshop Overview

The NSF Workshop on Planning for Cyberinfrastructure Software was conducted in Arlington, VA on October 5-6, 2004. Twenty three (23) leading national CI software developers and architects and fifteen (15) Federal Agency program officers (from NSF and other agencies) were invited to attend and participated in the workshop. For a list of participants and the workshop agenda, see pages 17-19.

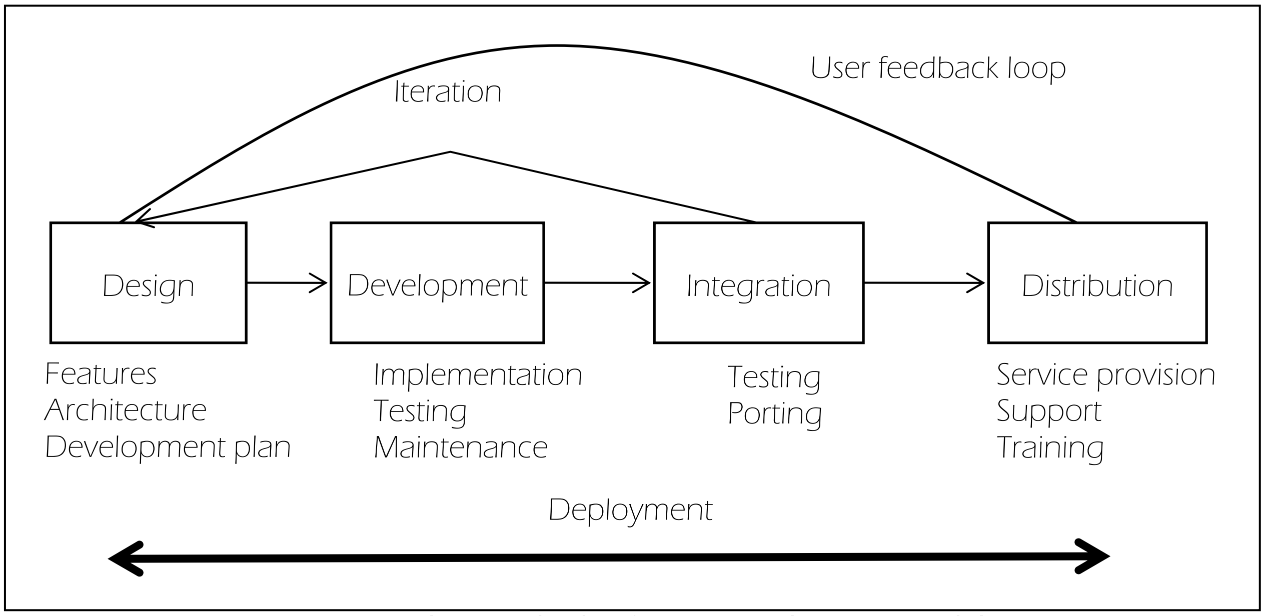


Figure 1: The Standard Software Development Model

The workshop agenda and discussion was based on the standard software model starting with design and ending with distribution (see Figure 1). It is important to point out that although the model looks linear, it is recursive and has several feedback loops as software development and deployment requires feedback from users and undergoes multiple iterations throughout its life cycle. The interactions will be particularly common for experimental software such as we envision for much of CI, and the feedback especially important. That said, the model provided a good basis and focus for workshop discussions to make sure all pertinent components and salient issues were addressed.

Session leaders were chosen for each of the eight sessions; each Lead had the responsibility to convene the session team and put together a presentation to address a series of questions. Each presentation also explored various alternative solutions and concluded with a specific approach and recommendations. Two Respondents were also assigned to each session team to play a formal "counterpoint" role by highlighting certain aspects of the CI component and providing additional information and alternative solutions. The remainder of the time was set aside for discussion leading to a consensus and summary.

SUCCESS METRICS SESSION

LEAD: Rick Stevens
RESPONDENTS: Carl Kesselman, Marty Humphrey
NSF: Sangtae Kim, Debbie Crawford

What measures should we use to judge a body of CI software?

- Impact on individual research communities vs. widespread adoption and usage?
- Usability and satisfaction vs. new features and performance?
- Evolvability and flexibility vs. stability and availability?
- Domain impact vs. wide adoption?
- Ease application development vs. operational capability?
- Unique needs of NSF community vs. eventual commercial success?

Five principle software infrastructure success metrics and measures were identified (see Appendix C: Summary of Workshop Presentations). Generally, any infrastructure that benefits society and improves Quality-of-Life is by definition successful if users are not aware of its presence and take it for granted (like freeways, water supply, electricity, the Internet and e-mail). This same definition can be applied to CI software.

1. Readily available - general availability; commodity item with many suppliers; adapting infrastructure to application is trivial
   Measures: many discrete sources of software; many suppliers; number of IT shops willing to support it.
2. Reliability – it just works; if it does fail, we don't notice it; whole enterprises exist to keep it working
   Measures: approaches 99.999% of uptime; increasing MTBF, decreasing MTTR; number of code installs without a reported failure; fraction of resources focused on reliability instead of features.
3. It is everywhere – software is ubiquitous in science and engineering; all major universities have internalized the services; internet providers have internalized it
   Measures: Number of CI projects use it; number of groups provide core support; a degree of commoditization; number of NSF directorates committed to deployment
4. It is easy to use – not aware you are using it; there is no learning curve; it is completely embedded in existing applications
   Measures: keystrokes for install, uninstall, update; fraction of CI tools that work; number of embedded applications use it

5. It directly supports other societal functions – it is accelerating important science programs and is adopted by programs that don't have to use it.
   Measures: Number of significant NSF (and other Agency) projects are using CI software; number of groups spontaneously adopt the technology; number of NSF directorates and programs jointly support deployment.

Discussion: It was noted that ease-of-use is often not considered relevant in design and development of research software, and this needs to change if software is to become adopted. Likewise, applications are also often not usually considered part of CI, but it is clear that applications must be integrated into software development and deployment projects from the outset. Other comments identified the need to establish specific funding efforts to focus on engaging users and small enterprises and expand the base of users and virtual organizations. The need to encourage provision of CI services, conducting user surveys and tracking actual usage and deployment was also highlighted.

GOALS IN GATHERING INPUTS SESSION

LEAD: Dan Atkins
RESPONDENTS: Tom Jordan, Vicky White
NSF: Sangtae Kim, Debbie Crawford

What are the goals and processes appropriate for prioritizing and dealing with related communities?

- End-users (NSF supported scientific communities, NIH research, etc)
- Applied researchers in CI
- End-user communities
  - Generic vs. discipline-specific needs: ordering, identifying
  - What disciplines to serve
- CI researchers
  - Research prototypes vs. from scratch
  - Stimulate research into CI vs. implementation now
  - Tracking latest research vs. stability and reliability

Advanced CI R&D must work closely with science and engineering research disciplines; the key is integral teaming and interdisciplinary groups (see Appendix C: Summary of Workshop Presentations). Just asking about requirements is a tiny part of the problem – computer science developers must learn to "walk in the shoes" of the scientists. Social scientists also need to be involved to help with human-centered design and to also design and conduct longitudinal evaluation of what impact the technology is having on the practice of the research community. CI software development and deployment must be based on Borromean Design Ring Teams where a notion of mutual self-interest and effort ensures that relevant communities and stakeholders are involved from the outset. CI software development and deployment efforts need to adopt the Pasteur Quadrant Research model; a dual focus on creation of new knowledge and a specific domain application. In general, development and deployment efforts need to start small and be domain specific before they are expanded and generalized.

Discussion: NSF needs to take into account the broader picture of other agencies working in this area (e.g. NIH, DOE) as well as international efforts which have already done significant groundwork. NSF also needs to reward forward-looking domain scientists who are willing to set aside near-term requirements to focus on using and developing CI. New models of balances between cooperation and competition need to be encouraged and supported. Building and using CI is an organic process – till, plant, grow, nourish, reap and disseminate. It needs to be realized that much of infrastructure (including CI software) is a "public good" and needs to be funded that way. Without new money earmarked for it, infrastructure investment by most research disciplines will always be a second choice behind "more money for my project."

DESIGN PROCESS SESSION

LEAD: Carl Kesselman
RESPONDENTS: Deb Agrawal, Mark Ellisman
NSF: Guy Almes, Kevin Thompson; PSC: Matt Mathis

What processes should NSF employ in the design phase of CI software?

- Explicit delegated top-down roadmap and design vs. responding to proposals
- Roadmap, lifecycle, timing
- Dependencies
- Priorities
- Deployment feedback ➔ refinements
- Design ➔ development
- Developer-driven vs. end-user and domain-science driven

The CI design process has many inputs and outputs and follows a spiral model (see Appendix C for the set of sessions slides). The design must make sense from a systems perspective, must provide functionality to end users, must consider life cycle costs and issues (including development, testing, operations, maintenance) and must acknowledge the rapidly changing landscape of technology by designing for evolution. In brief, CI software design should follow a “top down/bottom up/middle out” approach.

What is meant by CI design? A system, set of systems, set of components

Design inputs

- user requirements (what should the infrastructure do)
- technology requirements (what are the right properties)

Design outputs

- architecture framework (e.g. Web services, OGSA)
- component function specification (protocol definition)
- interchange format specifications (e.g. ontologies, schemas)
- deployment scenarios (hard to “tease out” of design)

What is Top Down/Bottom Up/Middle Out?

- Need high-level plan and architecture design for all components
- Select components that provide needed functions; worry about integration later
- Create layered, hourglass design which facilitates reuse

Discussion: BIRN[3] adopted a top-down approach which was necessary to get something up and running quickly as opposed to NEESgrid which used more of a bottom-up approach. In both cases, the spiral model of development was utilized. Other comments included the need to work through the dependencies on resources up front. Two types of service support needs to distinguished; applications that are enabled by the software, and the services needed to support the software. Design efforts need to plan for a rolling set of experiments and criteria. Lastly, NSF needs to more closely monitor and measure progress and specific milestones in the design and deployment of CI, perhaps even include a "bidder's conference" prior to submission of proposals.

DEVELOPMENT PROCESSES SESSION

LEAD: Ian Foster
RESPONDENTS: Ben Domenico, Phil Papadapolous
NSF: Jose Munoz; Redhat, Matt OKeefe

What processes should NSF employ to develop CI software?

- Performers: commercial, academic, Federal lab, outsourced
- Specification vs. proposal driven
- Small vs. large teams, distributed vs. centralized
- Community development (technologists, end-users)
- Coordination of complements
- Deployment ➔ refinements
- Deadline driven vs. open-ended
- NSF program manager role (SCI and other directorates)

There are four approaches to developing and deploying CI software: business as usual, an emphasis upon software engineering, "harden" existing software, use of specialized teams (see Appendix C for the set of session slides). Under "business as usual", funding continues to support science, software is a by-product and becomes ad hoc and uncoordinated. A focus on software engineering runs the danger of not addressing cross-cutting issues across multiple domains. An emphasis upon hardening existing software assumes the "right" software has already been developed and is available. The recommended approach is to support specialized software teams to design, develop, and deploy CI software.

CI software must be application driven, but should not in general, be application directed. Application developers and users must be embedded into CI software teams and be equal partners; software developers can not "throw their software over the fence" to users and expect the software to be useful. Deep partnerships and mutual respect among users and developers is essential for successful CI software teams.

Software teams:
- expertise is scarce, and a critical mass is required
- need to provide career paths for software team members

[3] The Biomedical Informatics Research Network (BIRN) is a National Institutes of Health initiative that fosters distributed collaborations in biomedical science by utilizing information technology innovations. See http://www.nbirn.net/ .

- time frames need to be longer than usual 3-yr NSF programs; 5 years for development, 10 years for deployment and support
- coordination is expensive; takes time, effort and sustained funding
- need to provide opportunities for students and education

Software issues:

- CI software must be open source
- community contributions encouraged and facilitated
- international cooperation needs to be encouraged and enabled
- security needs to be vital emergent property with a well-defined review process
- exit strategies, including commercial integration and adoption, must be part of the process from the outset

Defining a CI software program:

- create 2-3 large CI software centers; define, build, integrate and evolve key software
    - training, integration, applications support
    - integrate contributions from other developers
    - provide testing facilities for community
- create small specific CI software projects
    - build specific components identified by National Software Roadmap
    - projects funded for 1-3 years
- establish science integration projects
    - medium size for 3-5 years
    - focus on specific domains and applications (not general)
- establish National CI software coordination group

Discussion: NSF needs to be more flexible in addressing and supporting CI software grants, cooperative agreements and contracts. NSF needs to coordinate with other agencies (NIH, DOE, etc) and encourage more interagency activities. The larger CI projects require more experienced project managers to oversee and monitor the project including having enough vertical integration to ensure that complexities and cross-cutting issues are tracked and managed. CI software projects need more focus on science and applications; projects need a vision and a roadmap to help identify complementary thrusts and knowledge gaps to keep the project and people on track. Lastly, NSF needs to be more involved in the monitoring and tracking CI software development and deployment projects.

INTEGRATION PROCESS SESSION

LEAD: Miron Livny
RESPONDENTS: Larry Peterson, Bob Wilhemson
NSF: Kevin Thompson, DOE: Mary Anne Scott, NIH: Peter Lyster

What processes should NSF employ to integrate software from multiple sources, commercial and non-commercial?

- Performers: commercial, academic, Federal labs, outsourcing
- Choices
- Platform

- Components
- Deployment feedback → integration
- Test and validation methodologies
- Interoperability standards
- NSF program manager role (SCI and other directorates)?

CI integration is unavoidable in a distributed environment where end-to-end functionality is delivered by complex, multi-vendor software stacks (see Appendix C for the full set of slides). Integration is challenging and difficult and is a possible target for applied research on its own. It is separate from development and feeds distribution efforts. An integrator packages and tests a collection of software components provided by one or more vendors and acts as a middle person between the vendors and the distributors. The integrator also provides a range of services from build and test to inter-component testing, troubleshooting and coordination.

Integration of independently developed software tools requires a well managed and coordinated effort that can offer a national build and test infrastructure as well as act as a source of expertise in software engineering technologies related to software integration tasks. Given the international nature of today's software tools, such an effort must have strong ties with integration activities in other countries. Integration is a real-world effort (revisions, bug-fixes, etc) which requires dedicated resources and attention. Heterogeneity is likely to grow as CI software deals with a dynamic set of applications, tools, operating systems and hardware. This puts a significant burden on the distribution tasks as they need to deal with an ever changing landscape of requirements, solutions and players. This also requires a flexible funding structure that can quickly adapt to such changes.

Discussion: Although integration components were identified, a number of operational challenges were noted facing the larger picture of CI.

- Who determines what should be integrated? What happens to the components not chosen?
- Who decides if a component is ready for integration (methodology, Quality Assurance)?
- Who decides if an integrator is capable of dealing with a component?
- How should the integration activity be evaluated?
- Who manages the release cycle of the packaged software

DISTRIBUTION/DEPLOYMENT PROCESS SESSION

LEAD: Randy Butler
RESPONDENTS: Miron Livny; Stu Feldman
NSF: Kevin Thompson, NIH: Mike Marron, Redhat: Matt OKeefe, PSC: Matt Mathis

What modalities and processes should NSF employ to distribute CI software?

- Choices: what and when
- Service provision vs. software download
- Repositories
    - "Certified" vs. informal
    - Coordination

- Monolithic vs. fragmented distributions
- Testing (how many configurations and environments, how detailed the validation?)
- Licensing terms and conditions
    - User modifications, contributions
    - Fees
- Support and training

The complexity of the distribution system spans hardware and operating systems support, software version support, distribution mechanisms, licensing, and costs recovery methods (see Appendix C for the set of slides). CI support has the difficult challenge to ensure that successful deployments of CI are useful for a range of application developers and end users, and are easily administered by resource providers. Distribution, deployment, and support are all heavily defined by the platforms destinations. The greater the heterogeneity of the destination the more costly issues surrounding distribution, deployment and support. Limited resources will likely necessitate the need to define a specific set (possibly changing over time) of platforms that will be supported. Finally incentives are needed to help lower the cost of entry barriers for researchers who need to learn how to use CI tools, developers that need to understand how to build tools and services that rely on CI, and resource providers so that they can successfully administer their resources.

Deployments should ideally be driven by a real application, experiment or process. Initially this drives the successful deployment demonstrating full functionality, and enabling a set of early users. Thereafter a frequent application is needed to stress the system and give the site or resource owner a reason to maintain the middleware software. Without the identification of a researcher or application that frequently runs, the configuration will likely slip resulting in non-functional middleware services.

Effective and efficient distribution requires efforts in three areas; lower adoption barriers, deployment activities and core support.

1. Lower adoption barriers
    - establish repositories and linked directories
    - establish CI training centers
    - encourage certificate authorities and common approaches to security
2. Deployment
    - work toward standardization of policies, schemas and services at universities and at national labs
    - encourage strong engagement with resource providers and users
    - provide incentives for researchers and service providers
    - fund a “small” number of supported operating systems
3. Core support
    - need to define a set of “core” tools and services
    - generate uniform documentation and training materials
    - service providers to support local users

Discussion: Distribution is complex and it is essential to draw boundaries around what it does and should do or it will end up ineffectual. For example, distribution should not be responsible for debugging or determine what should be selected for distribution. Other questions raised included how to handle or manage and coordinate multiple distributors; who sets policies; how will legal issues be resolved; and what type of interfaces and relationships should the distributor have with commercial players? The final discussion item focused on the need for NSF to develop programs which support distribution efforts including the expectation of a specific set of deliverables and services.

RELATIONSHIP TO COMMERCIAL SOFTWARE SESSION

LEAD: Stu Feldman
RESPONDENTS: Ken Klingenstein; Redhat: Matt OKeefe
NSF: Priscilla Nelson, Kevin Thompson

What processes NSF should employ for early cooperation with commercial firms and for the eventual commercialization of its software?

- Enhancing commercial potential
- Commercial firms: cooperation and participation in design, development, integration, use
- Phasing in commercialization
- Licensing terms and conditions
- Insuring continuity of availability to the NSF community

Cooperation and coordination with the commercial software sector has been largely ignored or is absent in NSF programs and projects. However, cooperation and coordination with the commercial sector is essential to ensure availability and continuity of CI software to the NSF community (see Appendix C for full set of slides). Currently, commercial participation, if it occurs, is an after thought or serendipitous; it needs to be factored into the software cycle from the outset, and commercial support or licensing needs to be viewed as a desired outcome. The commercial sector brings experience, process rigor, and a mindset for continued support. CI should leverage this expertise and background at all stages. The scientific community faces the classic "make or buy" decision in software and opts for the make decision without considering the long term obligations of make and the impacts on the future of scientific research. Furthermore, there are currently few, if any, incentives to encourage the involvement of commercial participation either for the researchers or the commercial entities themselves.

Why should a commercial entity be involved?
- software can be linked/added to existing products
- established processes and project management expertise for effective development and support
- infrastructure is a development and support activity, not research
- ability to support lifecycle
- record of success and competence
- paths to market and access to the software value net-

Why should infrastructure software stay academic?
- NSF always does it this way
- the community is (occasionally) good at it and enjoys doing it
- desire to extend research model; teach students to do development
- desire to keep control
- CI domain is still too complex, ill-defined and uncertain yet to be effectively developed

Potential commercial approaches
- moving research and experimental software into COTS (licensing, transfer)
- moving software into services (hosting, remote delivery)
- moving services and support to commercial providers
- direct involvement in outsourcing and offshoring
- Phasing in commercialization

Discussion: European Union Commission proposals often include a requirement for commercial partners; they tend to be cooperative and can lead to future service and/or consulting opportunities. Commercial interest will naturally occur as the market grows and it becomes a viable economic product, but that may take too long. Early planning, incentives, and collaboration are appropriate to ensure long-term quality and commitment, as well as mutual interoperability of different subsystems. Consideration of long term support of software needs to become mandatory, not simply an option. This can be accomplished by commercialization or BSD approaches. In any event, it is essential that NSF make an effort and a commitment to support CI software for the long haul, and not leave duration and long term support to chance. Program announcements and proposals should address not only continued development, but maintenance and support of CI software.

CROSS CUTTING ISSUES SESSION

LEAD: Ian Foster
RESPONDENTS: Dan Atkins, Rick Stevens
NSF: Steve Meacham, Sangtae Kim

- How to determine appropriate budget in build-up and steady state?
- How to determine relative budgetary needs of research, design, development, integration, and distribution?
- What should NSF do first, and when?
- Post evaluation of plan and outcomes and feedback
- International input and coordination

CI software involves a vast array of cross cutting issues because it not only serves and supports the research enterprise, it is also the subject of research itself and is in process of being developed and established (see Appendix C: Summary of Workshop Presentations for the full set of session slides). The organizational structure of software projects need to be iterative and recursive. Cross-fertilization between applications and infrastructure deployment is essential, and although deployment has to start with a specific domain, it also to be concerned and supportive of common requirements and software. CI is multi-disciplinary by definition and requires significant coordination at multiple levels (e.g. software, application, inter-project, international). Software projects need peer and merit review during and post project for useful evaluation; NSF program managers need to provide more coordination across projects, directorates as well as involving resources outside NSF.

Software faces several major challenges:
- moving software from small projects to large scale use and deployment
- helping software transition from development to infrastructure

- what should a program look like to address these challenges to produce, deploy, and apply CI software?
- how can the program and projects be made attractive to other directorates?

A new program structure with four components was proposed as a model for CI software development and deployment.

1. Establish 2-3 software teams at $5-10M/yr to design and develop core CI software
2. Establish 4-6 Application Software teams to engage major application communities and encourage use of common infrastructure
3. Provide "seed funding" to 10-20 universities to spur deployment of services, scale over time
4. Establish a National CI Coordination organization to support the work of the program and provide coordination support

Discussion: Need to involve international and interagency roles in the model perhaps through joint projects and/or matching funds. Coordination committees and activities have to be stronger and more "heavy weight" in form and function. Cross-cutting issues in knowledge, information and data management issues need to be addressed at the same time.

APPENDIX A
Workshop Agenda

October 5

8:30	Breakfast *(Buffet provided in Workshop Room)*	
9:00	Welcome and NSF Perspective	Sangtae Kim
9:15	Workshop charge/Agenda	Blatecky/Messerschmitt
9:45	**Success Metrics Session** **LEAD: Rick Stevens** **RESPONDENTS: Carl Kesselman, Marty Humphrey** **NSF: Sangtae Kim, Debbie Crawford** *What measures should we use to judge a body of cyberinfrastructure software?*	
10:30	Break	
10:45	**Goals in Gathering Inputs Session** **LEAD: Dan Atkins** **RESPONDENTS: Tom Jordan, Vicky White** **NSF: Sangtae Kim, Debbie Crawford; NIH: Eric Jakobsson** *What are cyberinfrastructure goals in dealing with related communities?*	
11:30	**Design Process Session** **LEAD: Carl Kesselman** **RESPONDENTS: Deb Agrawal, Mark Ellisman** **NSF: Guy Almes, Kevin Thompson; PSC: Matt Mathis** *What processes should NSF employ to design Cyberinfrastructure software?*	
12:30	Lunch *(Buffet provided in Foyer)*	
1:30	**Development Processes Session** **LEAD: Ian Foster** **RESPONDENTS: Ben Domenico, Phil Papadapolous** **NSF: Jose Munoz; Redhat, Matt OKeefe** *What processes should NSF employ to develop Cyberinfrastructure software?*	
3:00	Break *(Refreshments and snacks provided)*	
3:30	**Integration Process Session** **LEAD: Miron Livny** **RESPONDENTS: Larry Peterson, Bob Wilhemson** **NSF: Kevin Thompson, DOE: Mary Anne Scott: NIH Peter Lyster** *What processes should NSF employ to integrate software from multiple sources?*	

4:30	**Distribution Process:** **LEAD: Randy Butler** **RESPONDENTS: Miron Livny; Stu Feldman** **NSF: KevinThompson, NIH: Mike Marron; Redhat: Matt OKeefe;** **PSC: Matt Mathis** *What approaches and processes should NSF employ to distribute CI software?*
5:45	Break for dinner (everyone on their own)

October 6

8:30	Breakfast *(Buffet provided in Workshop Room)*
9:00	**Relationship to commercial software:** **LEAD: Stu Feldman** **RESPONDENTS: Ken Klingenstein; Redhat: Matt OKeefe** **NSF: Priscilla Nelson, Kevin Thompson** *What processes should NSF employ for commercialization of software?*
10:15	Break
10:30	**Cross-cutting issues:** **LEAD: Ian Foster** **RESPONDENTS: Dan Atkins, Rick Stevens** **NSF: Steve Meacham, Sangtae Kim** *What factors should determine the need budget in transition and steady state?*
12:00	Lunch – Adjourn *(Sandwich/Salad Buffet provided in Foyer)*
1:00	Workshop Editors meeting (less than 60 minutes) Report organization, assignments, timelines

APPENDIX B
Workshop Attendees

Name	Affiliation
Deb Agarwal	LBL
Dan Atkins	University of Michigan
Alan Blatecky, Workshop Co-Chair	RENCI
Randy Butler	NCSA
Ben Domenico	UCAR
Mark Ellisman	UCSD
Stuart Feldman	IBM
Ian Foster	Argonne/University of Chicago
Fabrizio Gagliardi	CERN
Marty Humphrey	UVA
Tom Jordan	USC
Carl Kesselman	USC/ISI
Miron Livny	University of Wisconsin
Ken Klingenstein	Internet2
Matt Mathis	PSC
David Messerschmitt, Workshop Co-Chair	UC Berkeley
Matthew O'Keefe	ReHat
Phil Papadopolous	SDSC
Michael Papka	Argonne/University of Chicago
Larry Peterson	Princeton
Mary Spada, Workshop Editor	Argonne/University of Chicago
Rick Stevens	Argonne/University of Chicago
Bob Wilhemson	NCSA
Vicky White	Fermi National Lab

Federal Agency Attendees

Guy Almes	NSF
Vicki Booker	NSF
Debbie Crawford	NSF
Peter Freeman	NSF
Miriam Heller	NSF
Sangtae Kim	NSF
Peter Lyster	NIH
Mike Marron	NIH
Steve Meacham	NSF
Jose Munoz	NSF
Priscilla Nelson	NSF
Barry Schneider	NSF
Mary Ann Scott	DOE
Kevin Thompson	NSF
Michael Willig	NSF