National Science Foundation

NSF Workshop on Planning for Cyberinfrastructure Software

October 5 & 6, Washington, DC

Summary of Workshop Presentations

Organizers and moderators:
Alan Blatecky
David Messerschmitt

National Science Foundation

Workshop Charge

By

Alan Blatecky

Context

- Cyberinfrastructure (CI) technologies will transform how science and collaboration will be conducted in the coming decades
- Science and research is becoming increasingly dependent upon CI tools
- And, while there are some federal programs which support the development of new CI tools…

…there is little attention or funding focused on making CI tools robust, reliable, pervasive or persistent

Goals

Regarding NSF supported cyberinfrastructure software development:

- Utilize participant expertise and experience to identify the best processes and performers
- Where a consensus, provide concrete recommendations forming the basis for program announcements
- Where controversy, identify the who and how of more study

Final outcome is a written report & recommendations

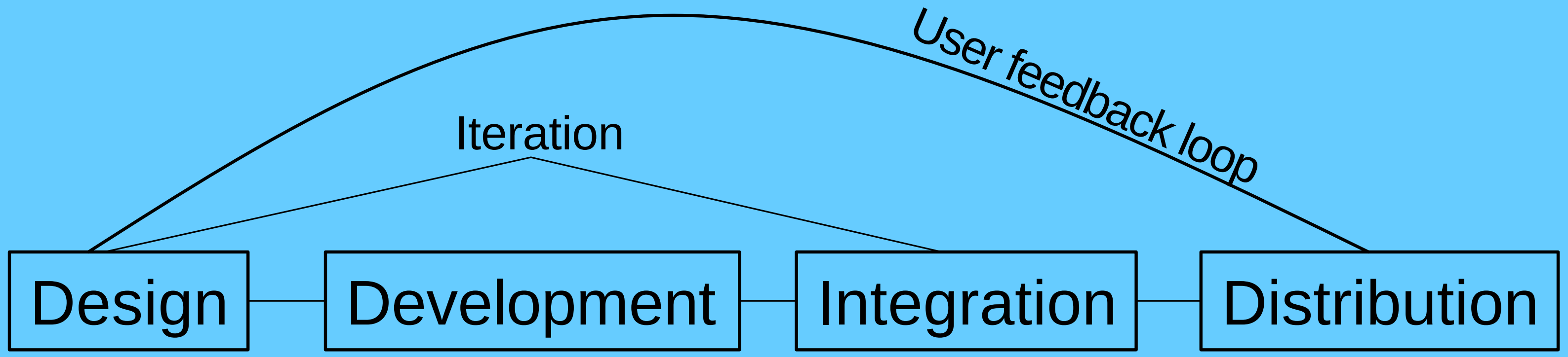
National Science Foundation

Workshop Architecture

By

David Messerschmitt

Cyberinfrastructure software lifecycle model



Features
Architecture
Development plan

Implementation
Testing
Maintenance

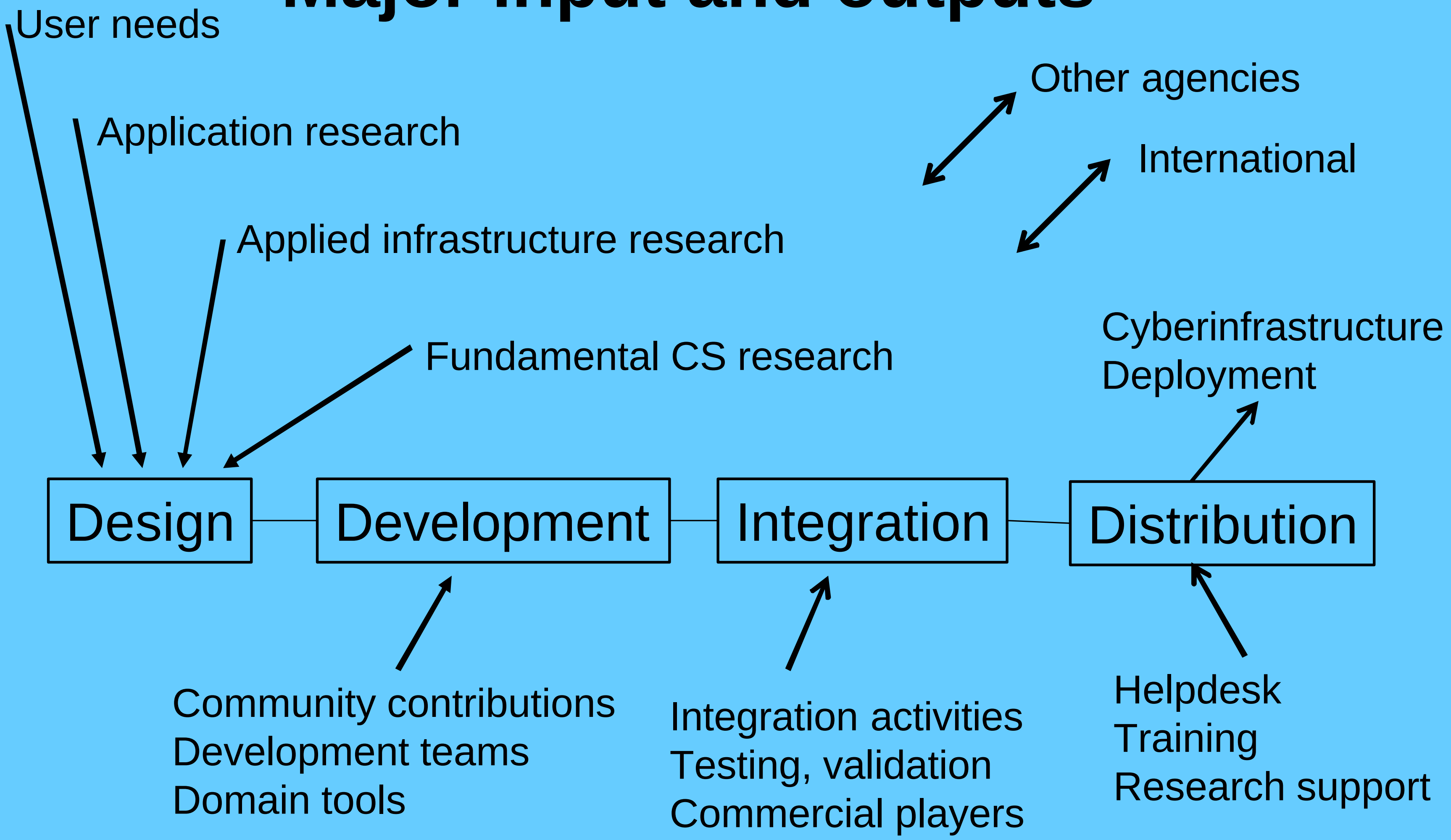
Testing
Porting

Service provision
Support
Training

Deployment

National Science Foundation

Major input and outputs



Workshop sessions

1. Success: definition
2. Inputs: goals and processes

Processes associated with:

3. Design
4. Development
5. Integration
6. Distribution
7. Commercial software
8. Cross-cutting issues

Session Guidance

By

Alan Blatecky

Roles

- **Moderator: (Heavy handed) focus, consensus, results**
- **Leader: Presentation on issues and recommendations**
- **Respondents: Confirm, criticize, alternatives**
- **NSF/NIH/DOE: Reality check, what matters**

National Science Foundation

Guide to leader

Establish a base for discussion:

- Primary issues
- Alternative approaches
- Recommendations

National Science Foundation

Remember

- Focus on processes
- Example:
  - We won’t try to determine which disciplines should be served first by cyberinfrastructure
  - We will recommend how NSF should determine that

Session 1
Definition of success

By what measures should we judge a body of cyberinfrastructure software developed and distributed by NSF?

National Science Foundation

Session 1: success
Issues

- 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?

National Science Foundation

Session 2: related communities

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 cyberinfrastructure

National Science Foundation

Session 2: related communities Issues

End-user communities

- Generic vs. discipline-specific needs: ordering, identifying
- What disciplines to serve

Cyberinfrastructure researchers

- Research prototypes vs. from scratch
- Stimulate research into cyberinfrastructure vs implementation now
- Tracking latest research vs. stability and reliability

National Science Foundation

Session 3: Design

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

National Science Foundation

Session 3: Design Issues

- 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

National Science Foundation

Session 4: Development

What processes should NSF employ to develop cyberinfrastructure software?

National Science Foundation

Session 4: Development Issues

- 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)

National Science Foundation

Session 5: Integration

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

National Science Foundation

Session 5: Integration Issues

- Performers: commercial, academic, Federal labs, outsourcing
- Choices
  - Platform
  - Components
- Deployment feedback $\rightarrow$ integration
- Test and validation methodologies
- Interoperability standards
- NSF program manager role (SCI and other directorates)?

National Science Foundation

Session 6: Distribution

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

Session 6: Distribution Issues

- Choices: what and when
- Service provision vs. software download
- Repositories
  - “Certified” vs informal
  - Coordination
- Monolithic vs. fragmented distributions
- Licensing terms and conditions
  - User modifications, contributions
  - Fees
- Support and training

National Science Foundation

Session 7: Commercial software

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

Session 7: Commercial software Issues

- **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**

National Science Foundation

Session 8: Cross-cutting issues

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

End

Cyberinfrastructure Software Initiative: Metrics for Success

Rick Stevens
The University of Chicago
Argonne National Laboratory

Cyberinfrastructure Software

- Cyberinfrastructure software as infrastructure
  - Must support the NSF raison d'etre
  - Accelerating important science programs
  - Sweeping in all of science infrastructure
- Not all software is CI SW
  - It should be the hard “core” on which many other tools and applications can be built
  - It should be the common currency for building Cyber infrastructures for science
  - Common tech that has broad benefit and leverage
  - Empower many applications and disciplines

Cyberinfrastructure Software as Infrastructure

Any Infrastructure that benefits Society and improves Quality-of-Life is by definition only successful if users are not or (no longer) aware of its presence and take it for granted, like the freeways we drive on, the drinking water supply, electricity, or the Internet and e-mail. The key attributes are that the infrastructure is (1) readily available, (2) reliable, (3) everywhere, (4) easy to use, and (5) directly supports or benefits other societal functions. - Frieder Seible, UCSD

National Science Foundation

Metrics - Easy to Measure Quantities that help determine if the program is on track towards success

- Some assumptions
  - Success is not highly non-linear
  - Real Success and Program Success are similar
  - CI Software is similar to other types of infrastructure
  - Our intuition can be useful in discussing the topic
- Program success metrics are proxies for more important measures
  - Improved scientific productivity
  - Reduced cost of large-scale facilities
  - Improved scientific decisions

Readily Available

- CI Software will be infrastructure when:
  - The user has an assumption of availability
  - It is a commodity with many suppliers
  - Adapting the infrastructure to the applications in trivial
- Progress metrics:
  - Number of discrete sources of the CI SW and services
  - Ease of requesting service or SW (time to availability)!
  - Degree of commodity with many many suppliers
- Program metrics:
  - Number of university IT shops willing to support it
  - Multiple sources to insure availability

Reliable

- CI Software will be infrastructure when:
  - It just works
  - It might fail but we don’t notice the failures
  - Whole enterprises exist to keep it working
- Progress metrics:
  - Approaching 4 or 5 Nines of uptime (99.999%)
  - Increasing MTBF, decreasing MTTR?
  - Number of code installs without a reported failure
- Program metrics:
  - Reliability goals should be explicitly stated and tracked
  - Fraction of resources focused on reliability instead of features

Everywhere

- CI Software will be infrastructure when:
  - CI SW is ubiquitous in science and engineering
  - All major universities have internalized the services
  - Major internet providers have internalized it
- Progress metrics:
  - Number of CI projects using CI core SW
  - Number of groups supporting the core SW
  - Degree of commoditization
  - Decreasing number of Internet providers without it
- Program metrics:
  - Number of partner directorates committed to deployment

National Science Foundation

Easy to Use

- CI Software will be infrastructure when:
  - You are not aware you are using it
  - There is no learning curve
  - It is completely embedded in existing applications
- Progress metrics:
  - Keystrokes for install, uninstall, update, etc.
  - For developers, the fraction of CI tools that just work
  - Number of embedded applications using it
  - Fraction of people who stop talking about it
- Program metrics:
  - Existence of market forces in the program to encourage investment in ease of use

CI Software Directly Supports or Benefits other "societal" Functions

- CI Software will be infrastructure when:
  - It is accelerating important science programs
  - It is adopted by many science programs that don't have to use it
- Progress metrics:
  - Number (fraction) of significant large-scale NSF (and other agency) projects building on CI SW
  - Number of groups spontaneously adopting the technology
- Program metrics:
  - Number of NSF directorates and programs jointly supporting the CI SW program

Summary

- Metrics == things we can count
- CI SW == software for enabling e-Science
- Five principle measures
  - Availability
  - Reliability
  - Ubiquity
  - Ease of Use
  - Clear Benefits to Broad "society"

National Science Foundation

NSF-SCI Workshop on Cyberinfrastructure Software Program Planning

Session 2 - Goals in Gathering Inputs

Lead: Dan Atkins

Responders: Tom Jordan, Vicky White

Agency People: Sang Kim, Debbie Crawford

Sheraton Crystal City, 5-6 Oct. 2004

Goal and Session Outline

- Goal **- The goal of the workshop is to provide NSF (specifically SCI) with several concrete recommendations and ideas to help develop a new program announcement next year to address the software requirements for Cyberinfrastructure.**
- Session Outline (15 min. lead/responses + 30 min. discussion)
  - **Primary issues that need to be addressed.**
  - **Alternate solutions/approaches**
  - **Recommended approach**

National Science Foundation

Session Topics

- 1. Metrics of success
- 2. Goals in gathering input - *What are cyberinfrastructure goals in dealing with related communities?*
- 3. Design Process
- 4. Development Process
- 5. Integration Process
- 6. Distribution Process
- 7. Relationship to Commercial Software
- 8. Cross-cutting Issues

Cyberinfrastructure Goals*

- More applications, capabilities, efficiency
- Reuse and multiple-use of designs; capture of commonality
- Spread of best practice
- Achieving interoperability
- Provision of tools and services
- Shared facilities
- Assistance and expertise

* From Appendix A of RSETC

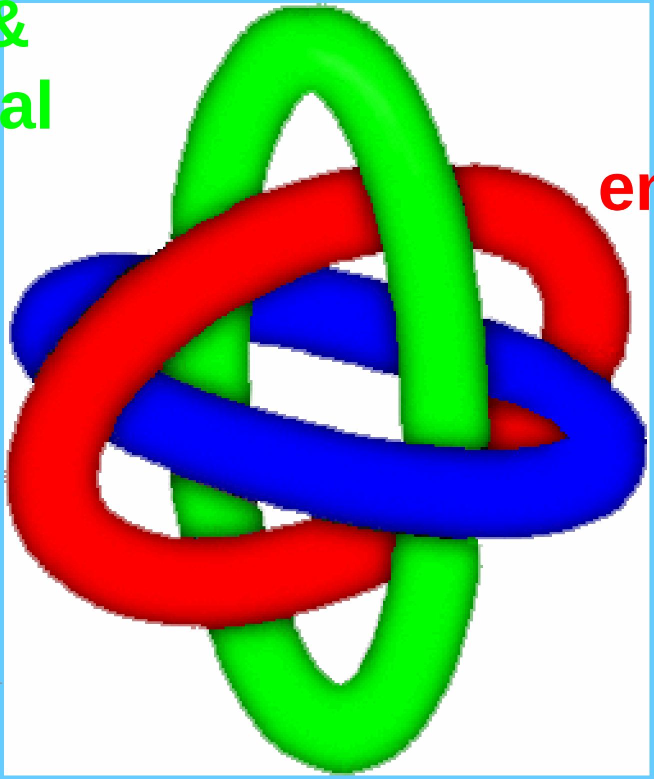
National Science Foundation

Relevant Communities (Stakeholders)

- CISE basic research
- Applied CISE research and development including software (middleware).
- Non-CISE science and engineering research communities. (application of CI to research).
- Organizations for provisioning CI and related services.
- Research universities and laboratories (users and co-funders)
- Commercial software companies
- Other U.S. Federal Agencies
- Global science communities -CI/e-science programs and institutions outside the U.S. (users, performers and co-funders)

Borromean Ring Design Teams

Behavioral & organizational sciences; policy



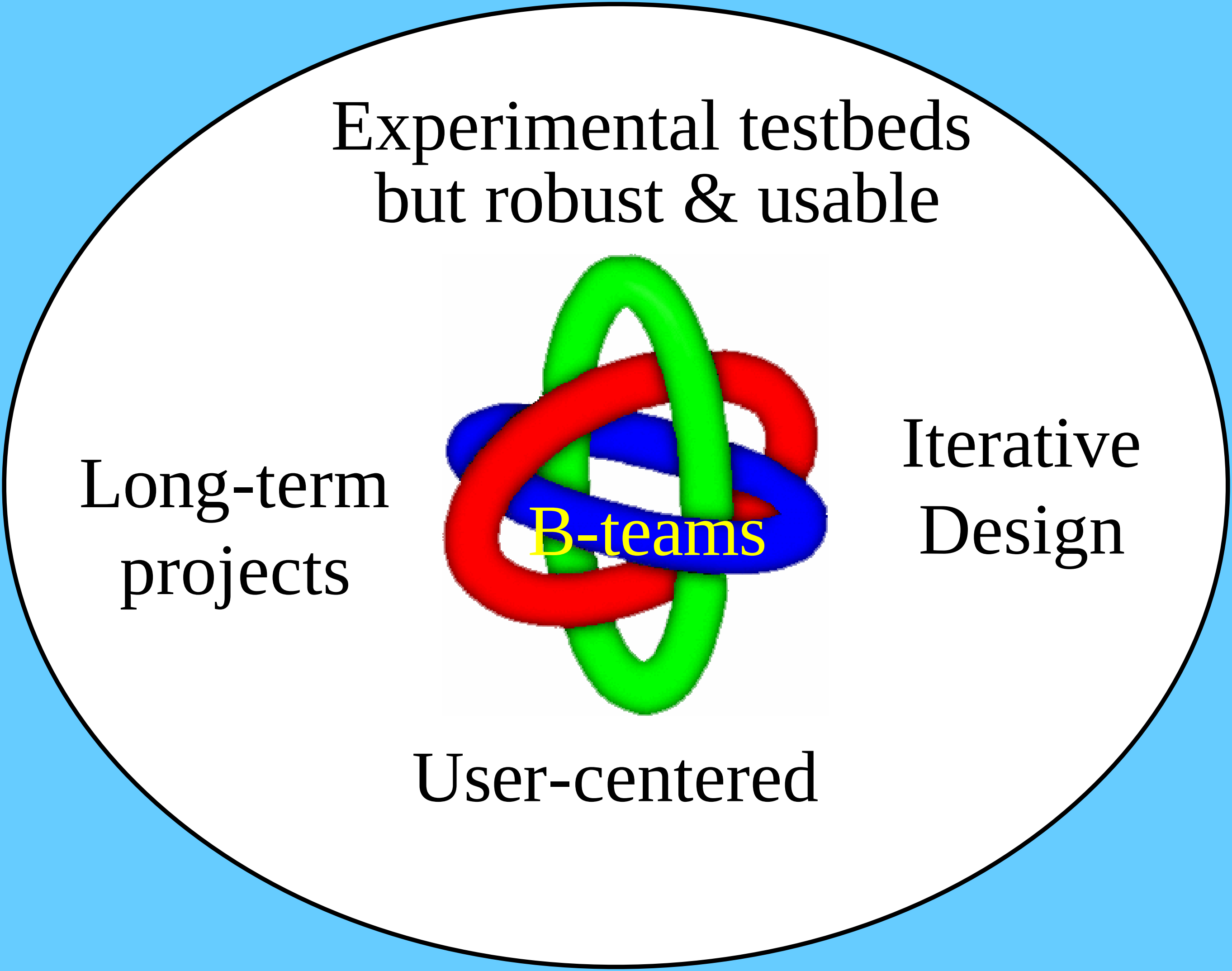
Computer & information, science, engineering R&D

Research (and education) Communities (disciplinary, multi-disciplinary)

mutual respect,
alignment of mutual self-interest,
collateral learning

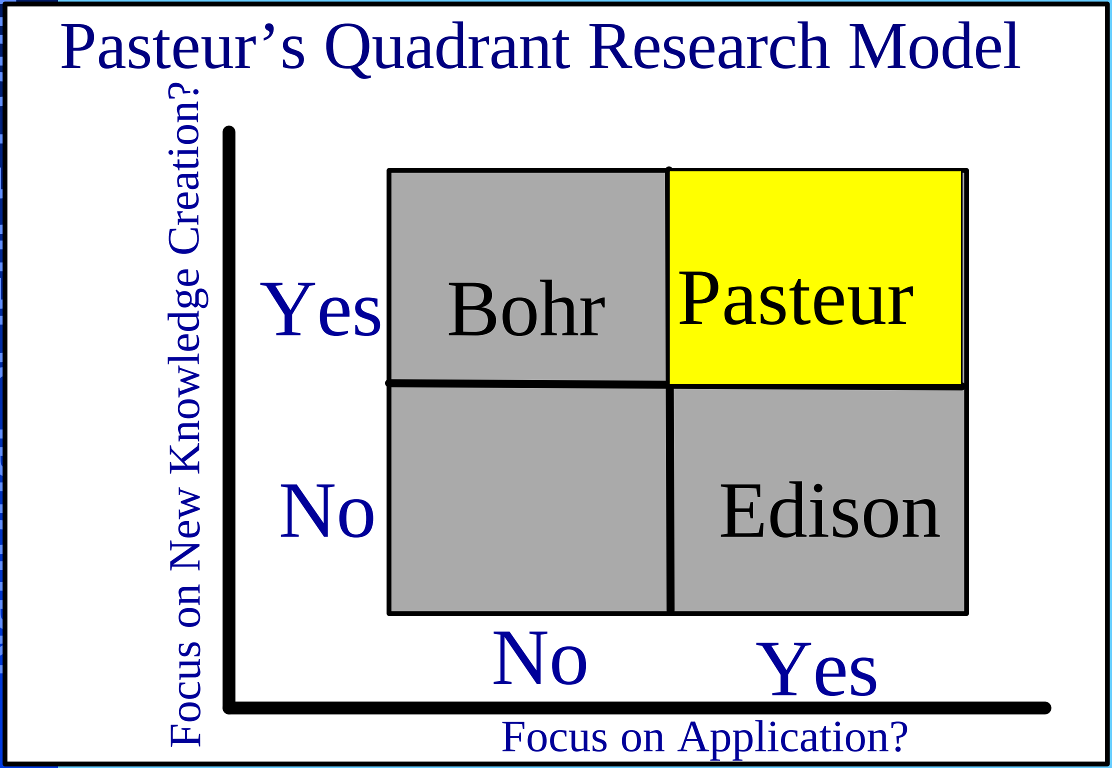
Three symmetric, interlocking rings, no two of which are interlinked. Removing one destroys the construct.

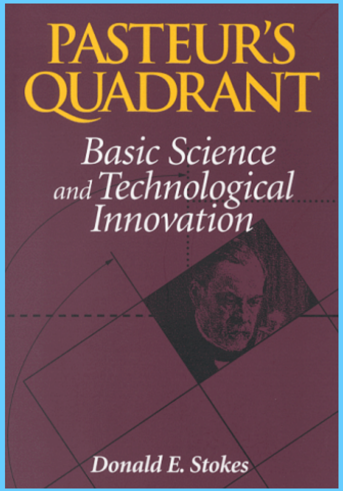
CKC* R&D Approach

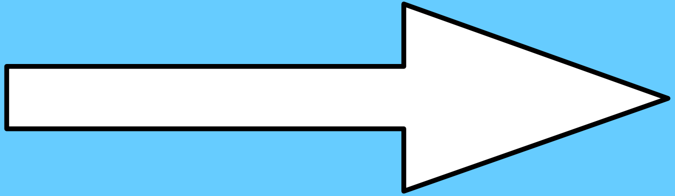


*CKC = Cyberinfrastructure-enabled Knowledge Communities

Need multi-disciplinary PQ research models


Pasteur’s Quadrant Research Model
Focus on New Knowledge Creation?
Yes
No
Bohr
Pasteur
Edison
No
Yes
Focus on Application?





Creation of knowledge: basic, curiosity-driven research

Application of knowledge

Classic Linear Research Model

Session 2
Issues to consider in gathering inputs to design a CI software program

As NSF organizes a cyberinfrastructure software program, what are some goals/issues in dealing with the relevant (NSF) communities:

1. **What is the balance between generic and discipline/project-specific needs?**
2. **What communities are selected first, and how?**
3. **Leveraging (building upon) working research prototypes vs. development from scratch? Extent of encouraging cooperation/merger between project, e.g. middleware.**
4. **Stimulating research into applications and infrastructure vs. leveraging existing research?**
5. **Tracking latest research vs. stability and reliability?**
6. **Early definition and implementation vs. identifying needed research and waiting for results?**

How to manage (set priorities) in the balancing act between….

- 1); 1) CI R&D; 2) (transformative) use of CI in science research; 3) creation/provisioning of CI.
  - Creating conditions for synergy between the above.
  - Holistic Integration of technical and social issues.
- Balance between 1) generic and 2) discipline/project specific needs?
- Building on existing projects (or mergers of projects) vs. active seeking of newer (clear and reset) approaches?
- Relative investment in various communities. Who goes first?

Balancing Act - continued

- Investment 1) in curiosity driven basic research and waiting for possible application outcomes; 2) in implementation based on what is known now; 3) virtuous cycles between the research and application in “Pasteur’s Quadrant”.
- 1)Leading edge/bleeding edge; and 2) stability or controlled evolution for users.
- 1)Higher technological performance; 2) more ubiquity of use. (top vs. lower parts of the pyramid).

Vicky White - FNAL

- Learn from existing specific CKCs - e.g. HEP and programs, e.g. ITR.
- Comments on issues in framing slides.

Tom Jordan - Geoscience “case study”

1. Complex, system-level problems spanning basic-science & mission goals.
2. Requires organizing large interdisciplinary, multi-institutional communities
3. Approach: Broad-based (virtual, functionally complete) collaboratories….
4. Supported by multi-layer CI
5. Must sustain long-term interactions between information scientists and domain scientists
6. Some metrics of success
7. Inadequate programmatic structure; can’t wait for lots of new money.

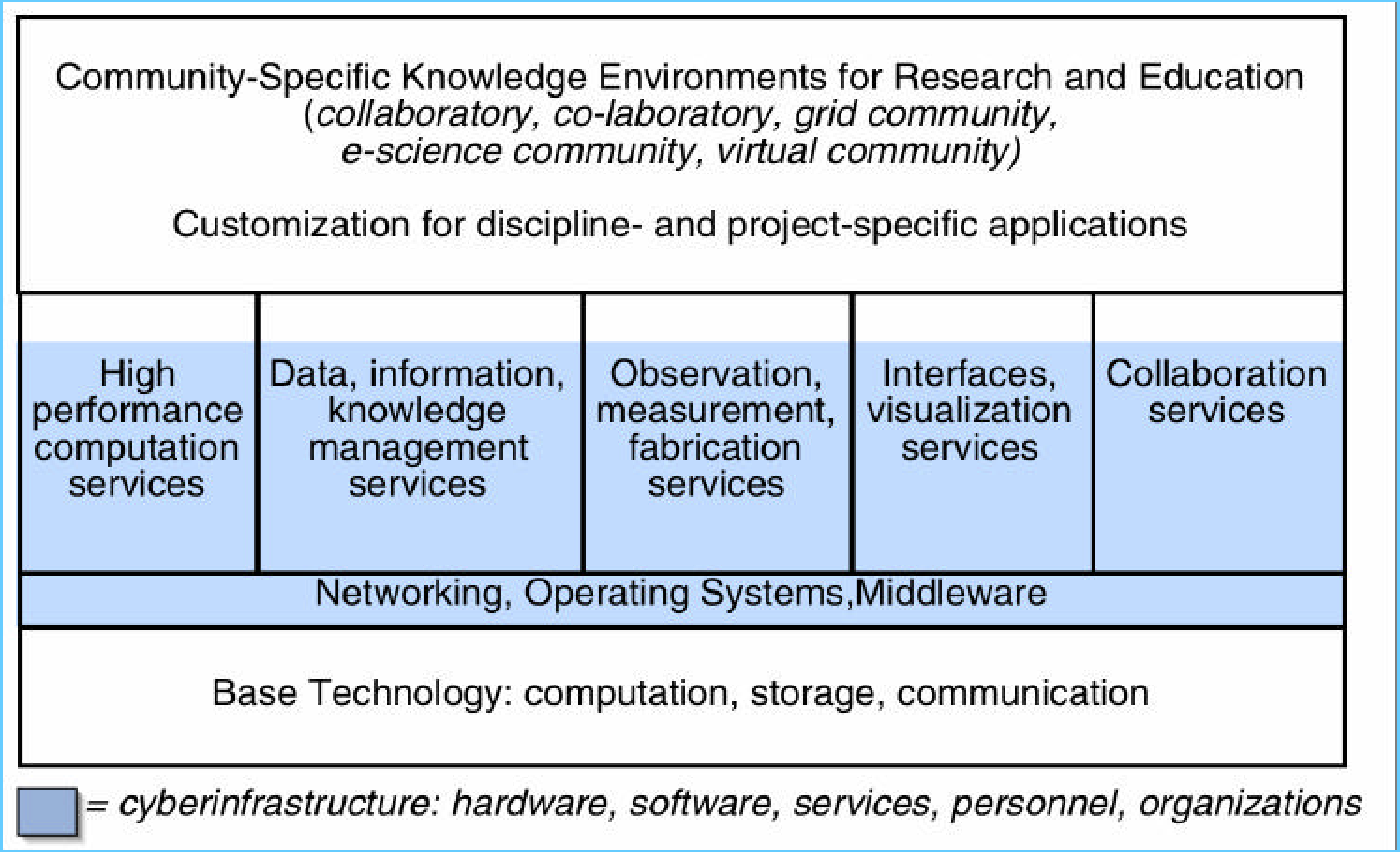
National Science Foundation

Open Discussion

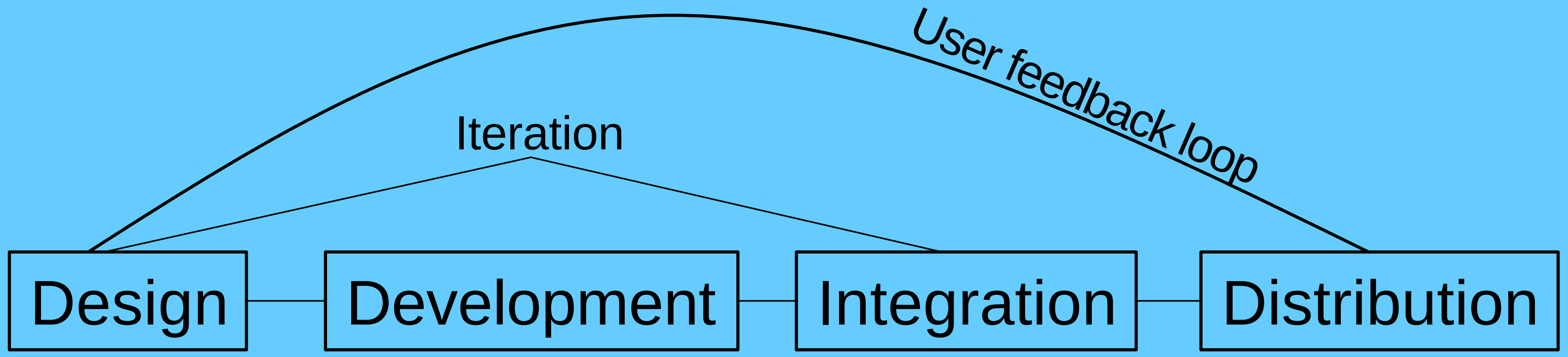
National Science Foundation

BUFFER

Functional stack


Community-Specific Knowledge Environments for Research and Education
(*collaboratory, co-laboratory, grid community, e-science community, virtual community)*
Customization for discipline- and project-specific applications
High performance computation services
Data, information, knowledge management services
Observation, measurement, fabrication services
Interfaces, visualization services
Collaboration services
Networking, Operating Systems,Middleware
Base Technology: computation, storage, communication
= *cyberinfrastructure: hardware, software, services, personnel, organizations*

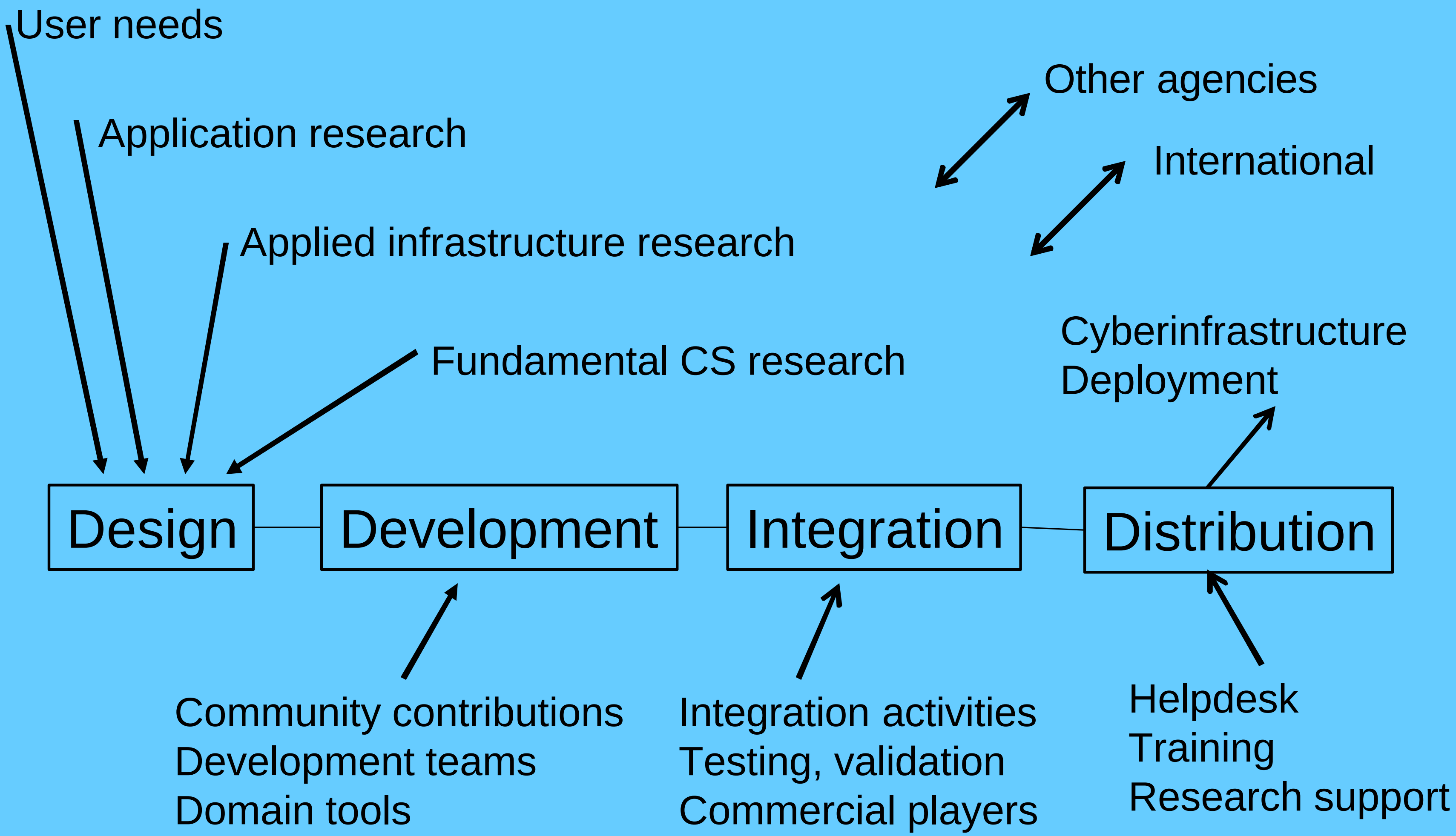
Cyberinfrastructure software lifecycle model



Features	Implementation	Testing	Service provision
Architecture	Testing	Porting	Support
Development plan	Maintenance		Training

Deployment

Major input and outputs



National Science Foundation

Session 2
Goals in gathering inputs from related communities

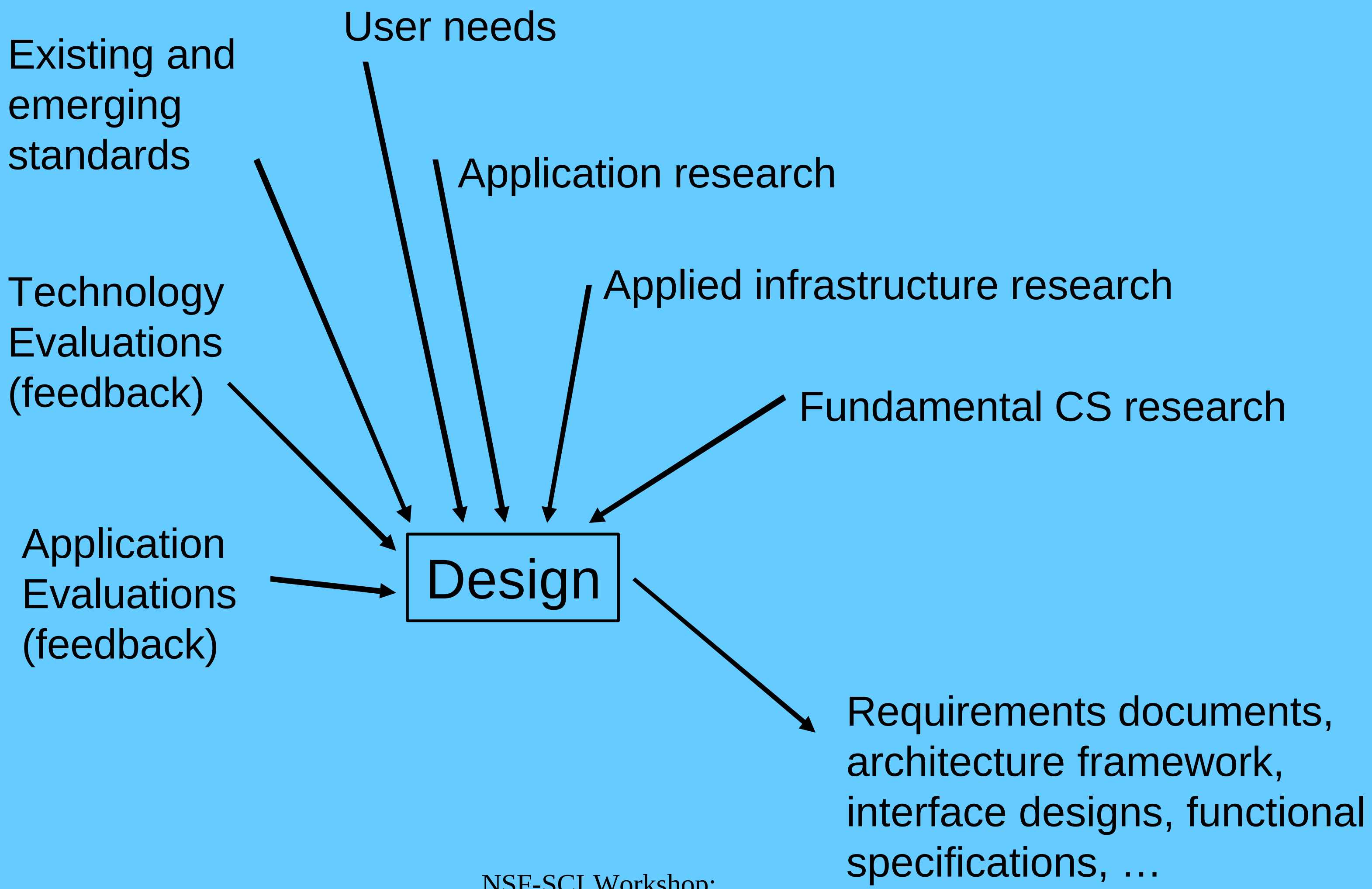
As NSF organizes a cyberinfrastructure development, what are some goals in dealing with related communities:

- Separating generic from discipline-specific needs
- What communities are selected first, and how?
- Leveraging working research prototypes vs. development from scratch?
- Stimulating research into applications and infrastructure vs. leveraging existing research?
- Tracking latest research vs. stability and reliability?
- Early definition and implementation vs. identifying needed research and waiting for results?

Design

Carl Kesselman, USC/ISI

Design Major input and outputs


User needs
Existing and emerging standards
Application research
Applied infrastructure research
Technology Evaluations (feedback)
Fundamental CS research
Application Evaluations (feedback)
Design
Requirements documents, architecture framework, interface designs, functional specifications, …

High-level Criteria

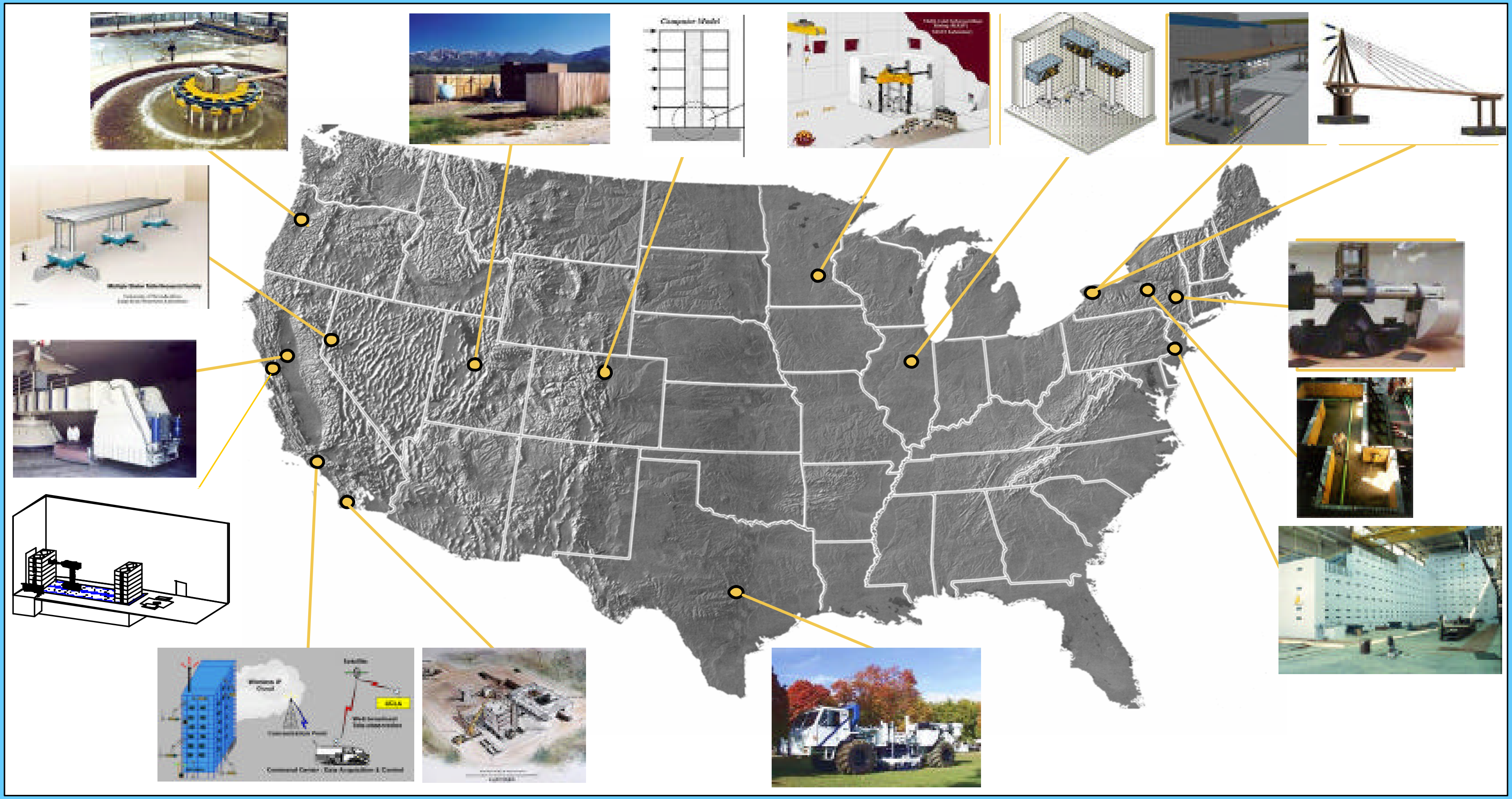
- Must make “sense” from a system perspective
  - Nothing replaces good systems engineering
- Must provide function to “end users”
- Must consider lifecycle/cost issues
  - Development, testing, operations, maintenance, update, …
- Must acknowledge rapidly changing landscape
  - Must design for evolution

What do we mean by CI design

- What are we designing?
  - A system, a set of systems, a set of components…
- Design outputs:
  - Architectural framework
    - Web services architecture, OGSA
  - Component function specification
    - Protocol or interface definition
  - Interchange format specifications
    - E.g. ontologies, schema's…
  - Deployment scenarios

Case Study: NSF Network for Earthquake Engineering Simulation (NEES)

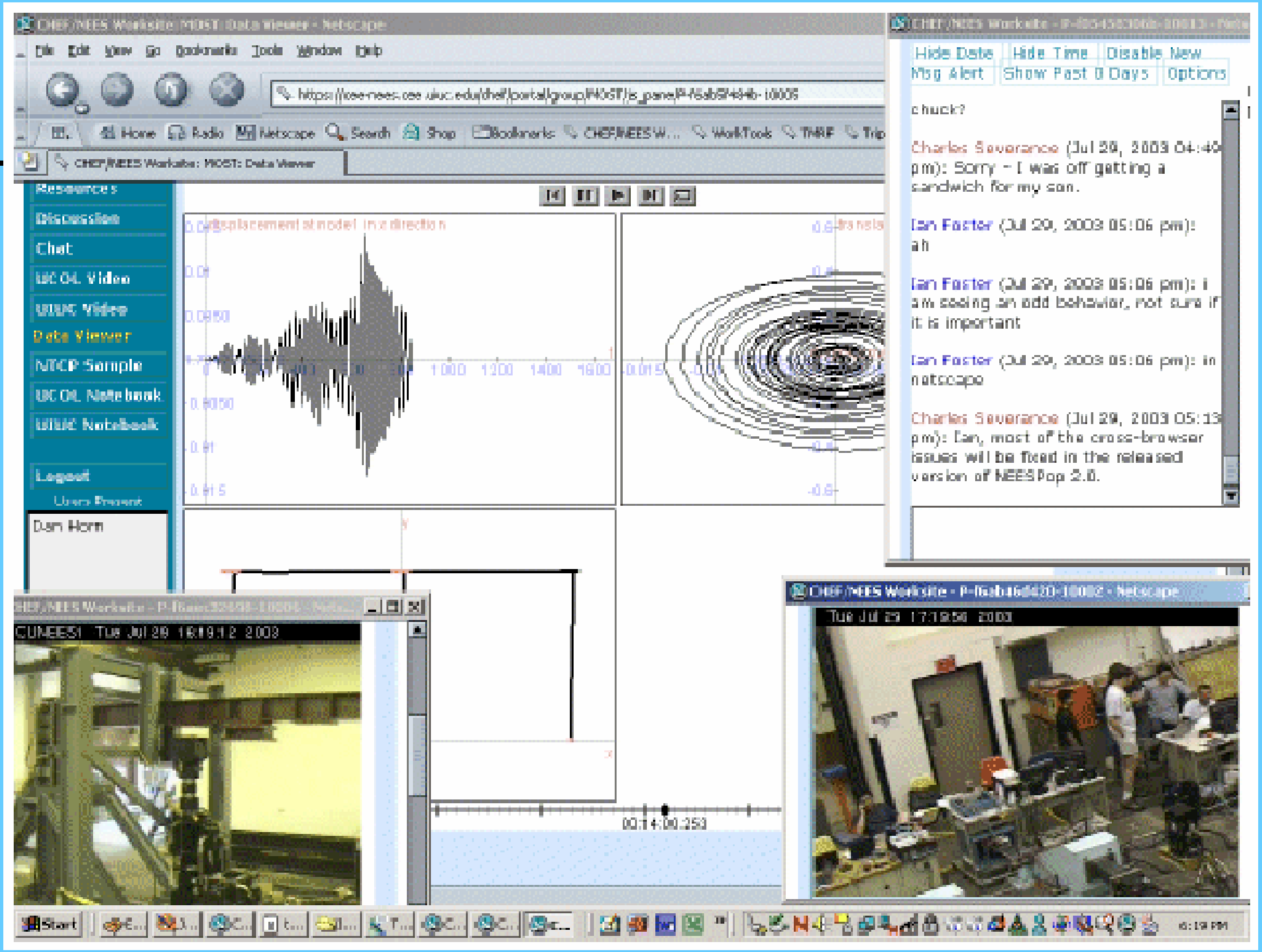
Transform our ability to carry out research vital to reducing vulnerability to catastrophic earthquakes



National Science Foundation

NEESgrid: User Perspective

Secure, reliable, on-demand access to data, software, people, & other resources (all via a Web Browser)



CHEF/NEES Worksite: MOST: Data Viewer - Netscape
File Edit View Go Bookmarks Tools Window Help
Home Radio Netscape Search Shop Bookmarks CHEF/NEES W... WalkTools THRF Trip
CHEF/NEES Worksite: MOST: Data Viewer
Resources
Discussion
Chat
UC OL Video
UIUC Video
Data Viewer
NTCP Sample
UC OL Notebook
UIUC Notebook
Logout
Users Present
Dan Horn
displacement at model in x direction
1000
1200
1400
1600
00:14:00:253
CUNEES! Tue Jul 29 16:19:12 2003
CHEF/NEES Worksite - Netscape
Tue Jul 29 17:19:56 2003
Start
Hide Date
Hide Time
Disable New Msg Alert
Show Past 0 Days
Options

chuck?

Charles Severance (Jul 29, 2003 04:40 pm): Sorry - I was off getting a sandwich for my son.

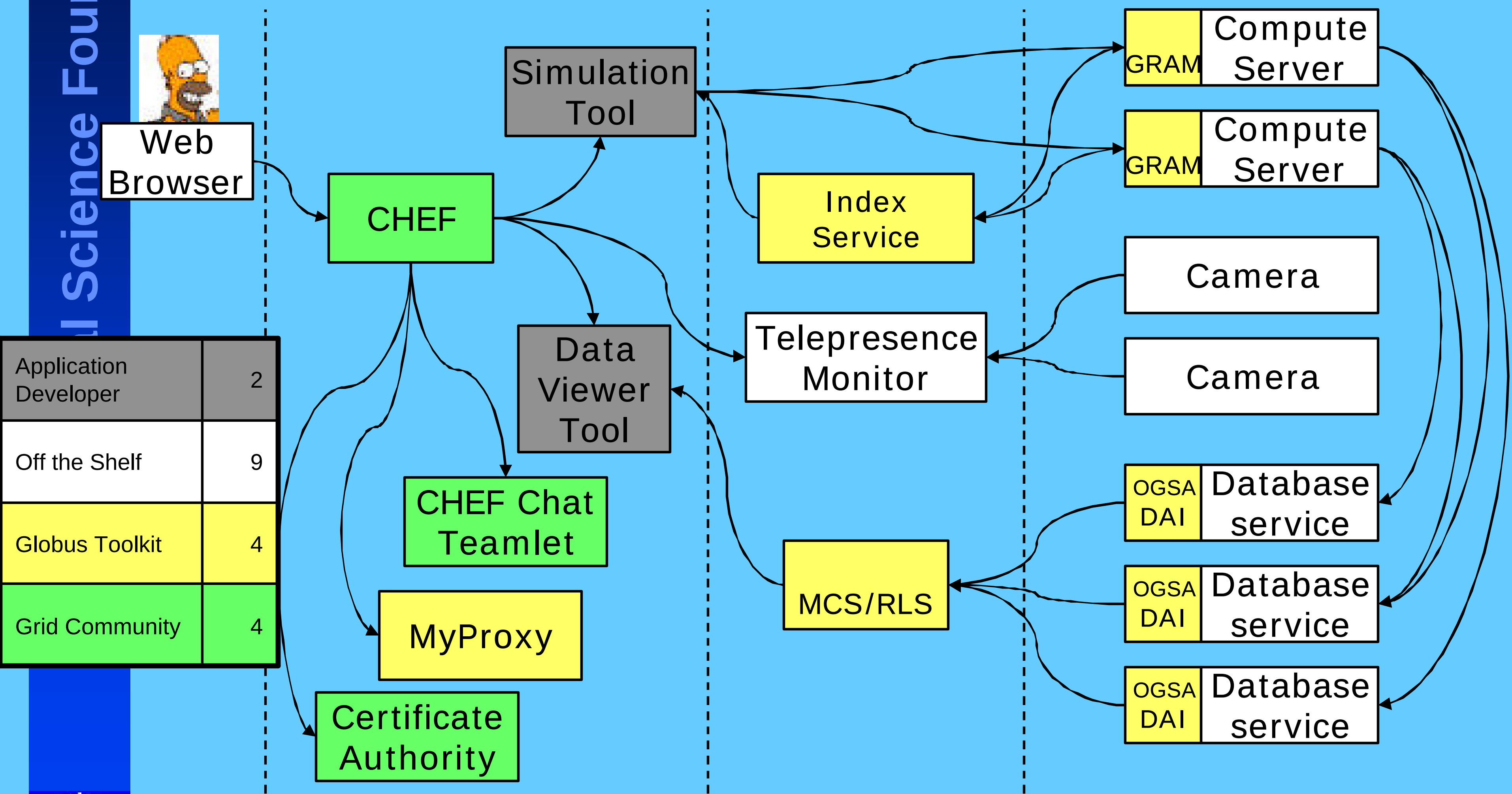
Ian Foster (Jul 29, 2003 05:06 pm): ah

Ian Foster (Jul 29, 2003 05:06 pm): I am seeing an odd behavior, not sure if it is important

Ian Foster (Jul 29, 2003 05:06 pm): in netscape

Charles Severance (Jul 29, 2003 05:13 pm): Ian, most of the cross-browser issues will be fixed in the released version of NEESPop 2.0.

How it Really Happens (with Cyberinfrastructure Software)



Application Developer	2
Off the Shelf	9
Globus Toolkit	4
Grid Community	4

Users work with client applications

Application services organize VOs & enable access to other services

Collective services aggregate &/or virtualize resources

Resources implement standard access & management interfaces

National Science Foundation

What shapes design

- User requirements
  - What do we want the infrastructure to do
  - May result in short-sighted or non-reusable
- Technology requirements
  - How to build a system with the right properties
  - How to determine right properties, may result in systems that cannot be used
- Experience in both the above
  - It may not be a good idea to separate design from development, e.g. Internet routing collapse

Role of Standards in Design?

- Standard is not equivalent to "widely used"
  - Adobe PDF (not a standard, widely used)
  - Many RFC are standard, but rarely used

That Old Internet Example

- A few basic design principles
  - Layered hourglass design (IP at neck)
  - End-to-end principle
- Technology driven at core "application/requirements" driven at upper levels
- Captures best practice as standard
  - Implementation required for RFC approval
- Enables reuse of expensive infrastructure
- Design for extensibility, evolution

The Open Science Grid Design Process

- Many pieces already exist and are being used by application communities
  - Globus, Condor, SRM, …
- Create roadmap that identifies sets of pieces to be “part of the system”
  - Committee drawn from OSG community
  - Applications may deploy additional components
- Specify configuration/deployment topology
  - Security, monitoring, storage element, …

BIRN and TG

- Build on variety of available technologies
- Constructed some missing pieces
- System design primarily defined by individual projects
  - i.e. not a committee

Relevant Observations

- Substantial capability, on time and on budget
  - Thanks to substantial body of existing application-independent building blocks
- "Experiment-driven deployment" a key strategy
  - Engagement with apps → sustained feedback
  - "Design-build-hand off" not right strategy
- Work in turn contributes to larger CI software
  - E.g., teleinstrumentation control
- Unclear model as to how design moves forward

Top Down/Bottom Up/Middle Out

- Have high-level plan and architecture/design for all components
  - Requires agreement across CI
- Select components that provide needed functions, worry about integration later
- Middle-out: create architectural framework
  - Layered, hourglass design
  - Facilitate reuse
- None at all?

National Science Foundation

Session 3
Design processes

What processes should NSF employ to design cyberinfrastructure software?

- Explicit top-down design vs. responding to community contributions
- Is there some sort of roadmap or does anything go; how are dependencies handled, how will deployment experiences feed back into design?
- How should design activities be linked to development?
- Delegated design responsibility vs. proposal-driven vs. community-driven?
- How are priorities set as to what gets done and by who/when?
- How can domain and community input be assured?
- Developer-driven vs. end-user-driven?
- What is the appropriate time scale for planning and design?

Cyberinfrastructure Software Initiative: Development Process

Ian Foster

The University of Chicago

Argonne National Laboratory

Respondents:

Ben Domenico, Phil Papadopoulos

What processes should NSF employ to develop cyberinfrastructure software?

- What are the goals and opportunities?
- What are the major challenges?
- What are some alternative approaches? and
- Provide some recommendations and/or be an advocate for specific ideas and suggestions

The Opportunity: Software as Time Machine

- Recall "supercomputer as time machine" story
  - x1000 CPU power = 15 years of Moore's law
  - Lets us do important things now, not later
  - And helps us understand how to shape the future

- Similar argument applies to CI software
  - Commodity software evolves to meet mass market needs (15 years = 5 Windows & Linus versions)
  - NSF spending on CI software accelerates access to advanced capabilities now
  - Also helps us understand those capabilities:

National Science Foundation

Using the Time Machine: In 2019, Scientists Routinely …

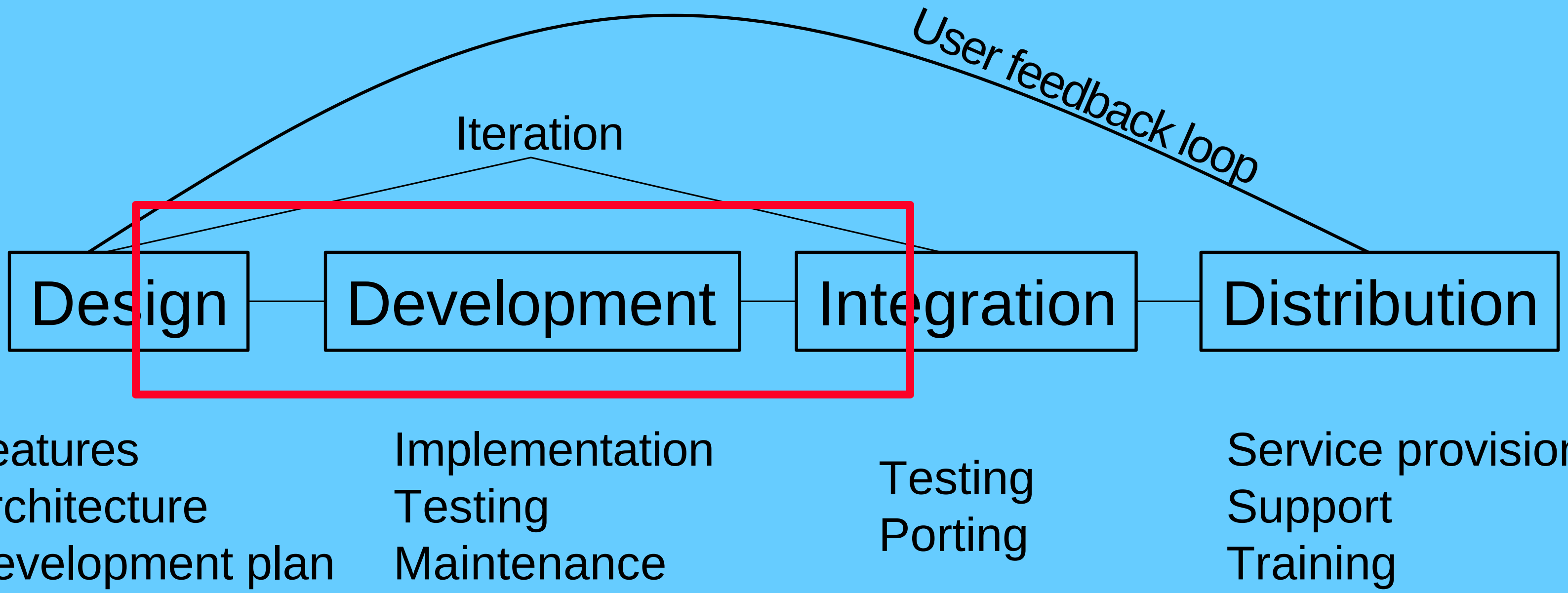
- Integrate & extract information from data at thousands of locations (millions in the case of sensors) worldwide
- Access petascale datasets & computational services, specialized software, remote instrumentation as part of their desktop environment
- Harness computers at hundreds of sites for ultra-scale computation and data analysis
- Work within small & large distributed teams of many sorts, many disciplines, in many places

Building the Time Machine: Challenges are Profound

- Clearly lots of hard technical issues relating to scale (in multiple dimensions), diversity, etc.
- But, in addition, we don't really know what people will do with these new capabilities
  - Software & applications must co-evolve
- And, we don't understand emergent properties that relate to how capabilities are used
  - Organizational structures, security, deployment, monitoring, operations, failure modes

→ We need to build systems to learn

Cyberinfrastructure software lifecycle model



Deployment

Four Alternative Approaches to Developing CI Software/Services

- Focus all-to-scarce funding on science!
  - Commodity software meets many needs
  - Grad students will write extra software if needed, as they have always done
- Fund software engineers, but put under direct control of scientists to meet application needs
  - They know best what needs to be done
- Modest funding to "harden" existing software
  - There's lots of good software out there
- Allocate significant funds to specialist teams, to design and build specialized CI software
  - Provide resources & develop the expertise needed to build the time machine

my view

The Need for CI Software Teams

- Software is a fundamental CI component …
  - Should be tackled as a cross-cutting issue
  - Commonality is critical in many dimensions
  - Should not, in general, be application-specific

- … but engagement with applications is critical to success
  - Cannot "throw it over the fence"
  - Need prolonged give and take

→ CI software must be application driven but should not, in general, be application directed
  - CI software teams as equal partner with users

National Science Foundation

What CI Software Teams Do

- They develop software
  - And also, on occasion, buy/borrow/harden
- They also research, design, integrate software
  - Spiral development model, not waterfall
  - Can’t separate research, design, development—but research is highly applied: 1-3 year timeframe
- They develop & sustain human expertise
  - The most important element!
  - Has significant implications for size & duration of funding
  - 2-3 year grants aren’t the way to do this!!

How Many Teams? What Profile?

- Size
  - Expertise is scarce
  - Critical mass issues due to diverse expertise needed
  - Need to provide career paths for excellent people
  - Coordination is expensive
  - →Large “center(s)” required for the bulk of the work
  - →Component-specific projects also important
- →Profile
  - Need to provide opportunities for student

Other Issues:
Open Source & Security

- CI software must be open source
  - Government is paying for it
  - Enable community contributions to testing, functionality
  - Enable international cooperation
  - Non-viral license to enable commercial use
  - Does not preclude compatibility with/integration of commercial software
- Security is a vital emergent property
  - CI software teams must include security expertise
  - Well-defined security review process

Defining a Program

1) CI software center(s): large, 5-10 years
   - Define, build, integrate, evolve key software as defined in “National CI Software Roadmap”
   - Provide training, integration, support to applications
   - Integrate contributions from other providers
   - Provide testing facilities for the community
2) CI software projects: small, 1-3 years
   - Build specific components identified within National CI Software Roadmap
3) Science integration projects: medium, 3-5 years
   - Deliver application successes
4) CI Software Coordination Group: some admin
   - Representatives from applications and CI

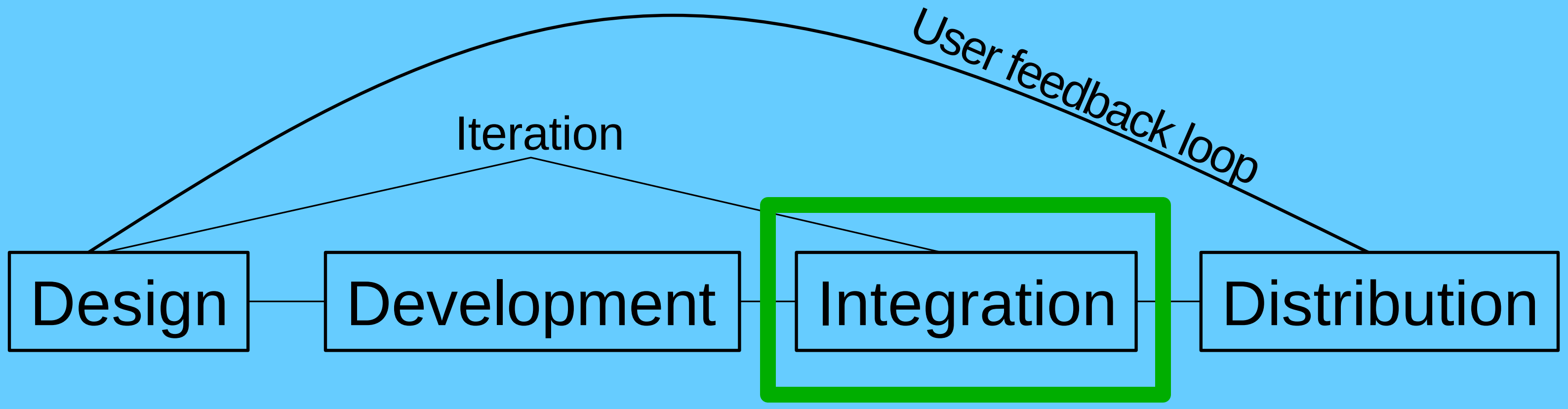
Integration

LEAD: Miron Livny

RESPONDENTS: Larry Peterson,
Bob Wilhemson

National Science Foundation

Cyberinfrastructure software lifecycle model



Features	Implementation	Testing	Service provision
Architecture	Testing	Porting	Support
Development plan	Maintenance		Training

Deployment

Integration is NOT

boring,
embarrassing,
easy,
straightforward,
or,
…

and therefore should be

OUT SOURCED

National Science Foundation

Observation

- Middleware integration is unavoidable as in a distributed environment end-to-end functionality is delivered by complex, multi-vendor software stacks. The only question is who does the integration, "the community" or the "end user"?
  - Understand functionality
  - Understand interdependencies
  - Understand development environment of components
  - "source" and/or "destination" can be integrators.

Basics

- What does it mean to be an “Integrator?”
  - Package a collection of components provided by one or more “vendor(s)” in to “distributable” middleware.
  - Act as an “middle person” between the “vendor(s)” and the “distributor(s).
  - Provide Q/A information to “management”
- What does an “integrator” do?
  - Build the components
  - Package components as a distribution
  - Verify package
  - Test components and package

National Science Foundation

Services Provided

- Advanced build, verification and test capabilities and know how.
  - Generic software infrastructure
  - Hardware resources
- “neutral” build and test environment
  - First time a component has to build and be tested away from “home”
- Inter component testing
- Inter-component troubleshooting
- Inter-component coordination

National Science Foundation

Operational Challenges

- Maintain good relationships with ALL parties.
- How is the integration “budget” determined?
- Who decides what should be integrated?
- Who decides if a component is ready for integration?
- Who decides whether the integrator is capable of dealing with a component?
- Who manages the release cycle of the packaged software?

Issues

- How is the integration activity evaluated?
- Generic or customized integration tools?
- Relationship between component and package build and test activities.
- Can/should the integrator patch a component?
- Should commercial components be included?
- Is porting the responsibility of the integrator?
- Who picks up “defunct” components?

Straw man

- National (multi-agency) integration facility
  - Distributed infrastructure
  - Centralized management
  - Strong ties to other (national and international) integration facilities
  - “Center of excellence” for build and test technologies, software, resources, education and outreach
  - Leverages cyberinfrastructure
  - Provide build and test services to “small” vendors
  - Interface with software engineering community

PlanetLab Experience

- Shared / Global Infrastructure
  - design / deploy / evaluate broad-coverage network services
  - 440 machines / 201 sites / 26 countries
  - 1228 users (225 want to define/build the infrastructure)
- Tensions
  - stable platform vs place to innovate
  - design principles: http://www.planet-lab.org/PDN/PDN-04-021/
- Experience
  - bootstrap: get running quickly / deploy now/ evolve later
  - steady state: engineered for continual evolution
  - design model: organic, with just-in-time central planning
  - users: transparent opt-in usage model
  - leverage: exploit off-the-shelf (preferably open source)
  - integration: no “pitch it over the fence” allowed
  - incentives: prototypes --> long-running services (make it real)

National Science Foundation

Session 5
Integration processes

What processes should NSF employ to integrate software from multiple sources:

- How are decisions made as to what should be included in integration activities?
- What can be learned and applied from deployment activities?
- How should integration be facilitated, accomplished?
- What is required to test and validate integrated software capabilities?
- How should interoperability be accomplished?
- Should commercial software development firms or government labs be contracted?
- What is the process to determine supported platforms?
- Role of NSF program managers (SCI and other directorates)?

PlanetLab Experience (L. Peterson)

- Shared / Global Infrastructure
  - design / deploy / evaluate broad-coverage network services
  - 440 machines / 201 sites / 26 countries
  - 1228 users (225 want to define/build the infrastructure)
- Tensions
  - stable platform vs place to innovate
  - design principles: http://www.planet-lab.org/PDN/PDN-04-021/
- Experience
  - bootstrap: get running quickly / deploy now/ evolve later
  - steady state: engineered for continual evolution
  - design model: organic, with just-in-time central planning
  - users: transparent opt-in usage model
  - leverage: off-the-shelf software (preferably open source)
  - integration: no “pitch it over the fence” allowed
  - incentives: prototypes --> long-running services (make it real)

LEAD

A Driver

Linked Environment for Atmospheric Discovery

A Cyberinfrastructure for Mesoscale Meteorology Research, Forecasting, and Education

http://lead.ou.edu

NSF Large ITR Project (funded FY04)

NSF-SCI Workshop:
Planning for Cyberinfrastructure Software

LEAD Requirements

- On-demand
- Real time
- Automated/intelligent workflow tasking
- Resource prediction/scheduling
- Fault tolerance within and across hardware
- Response to increased or decreased resources
- Dynamic interaction
- Data caching
- Data management
- Interoperability
- Grid and Web services
- Personal virtual spaces (MyLEAD)
- Distributed and local computation
- High performance networking between sites
- Collective analysis
- Desktop to petaflop
- Authentication
- Data publishing
- Security

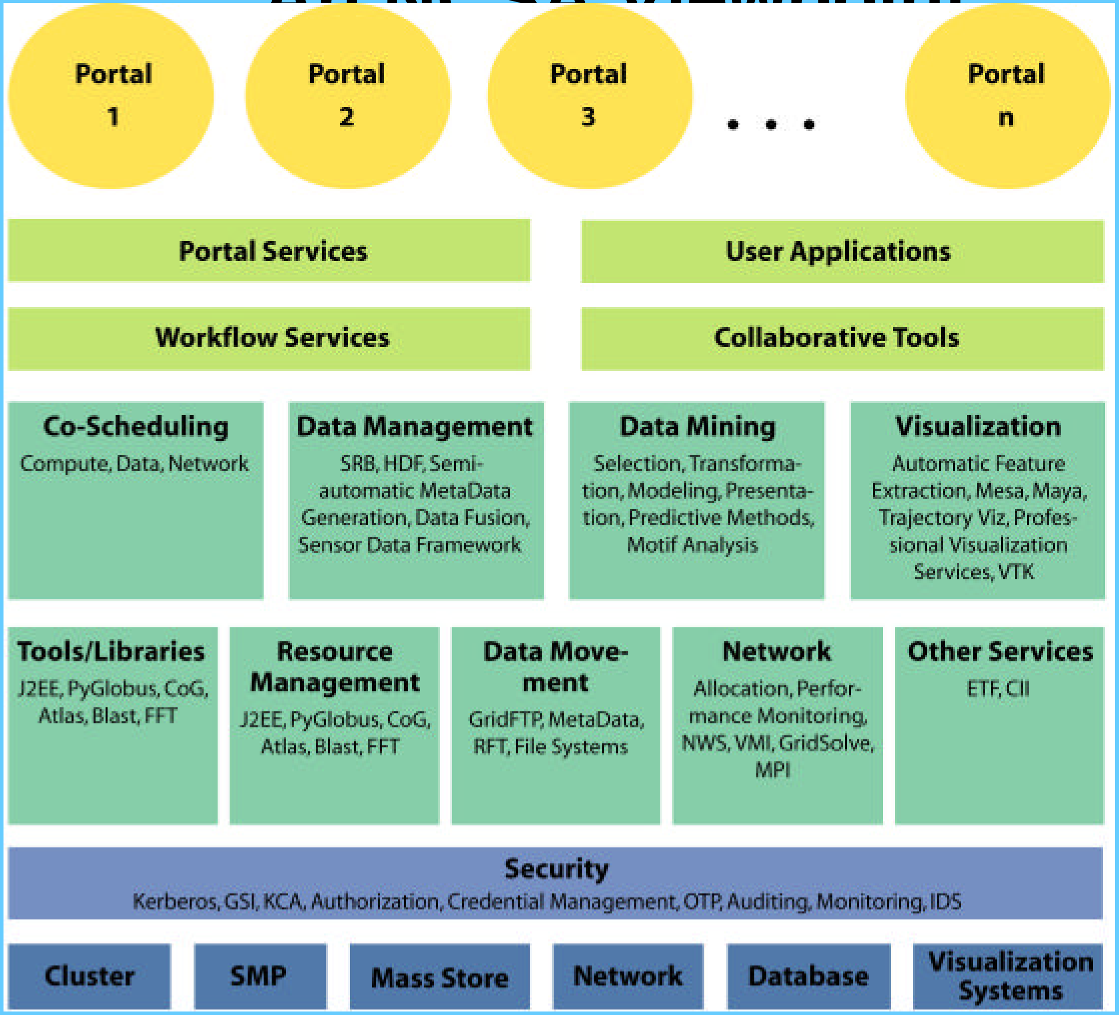
National Science Foundation

LEAD Requirements

- On-demand
- Real time
- Automated/intelligent workflow tasking
- Resource prediction/scheduling
- Fault tolerance within and across hardware
- Response to increased or dec[illegible]ased resources
- Dynamic i[illegible]on
- Data cach[illegible]
- Data mana[illegible]m[illegible]
- Interoperab[illegible]y
- Grid and W[illegible] s[illegible]s
- Personal vi[illegible]a[illegible] (L[illegible]
- Distributed [illegible]
- High perfor[illegible]ar[illegible]ng between sites
- Collective a[illegible]alysis
- Desktop to petaflop
- Authentication
- Data publishing
- Security

This is not easy!

Integrated Cyberservices and Tools An NCSA Viewpoint


Portal 1
Portal 2
Portal 3
. . .
Portal n
Portal Services
User Applications
Workflow Services
Collaborative Tools
Co-Scheduling
Compute, Data, Network
Data Management
SRB, HDF, Semi-automatic MetaData Generation, Data Fusion, Sensor Data Framework
Data Mining
Selection, Transformation, Modeling, Presentation, Predictive Methods, Motif Analysis
Visualization
Automatic Feature Extraction, Mesa, Maya, Trajectory Viz, Professional Visualization Services, VTK
Tools/Libraries
J2EE, PyGlobus, CoG, Atlas, Blast, FFT
Resource Management
J2EE, PyGlobus, CoG, Atlas, Blast, FFT
Data Movement
GridFTP, MetaData, RFT, File Systems
Network
Allocation, Performance Monitoring, NWS, VMI, GridSolve, MPI
Other Services
ETF, CII
Security
Kerberos, GSI, KCA, Authorization, Credential Management, OTP, Auditing, Monitoring, IDS
Cluster
SMP
Mass Store
Network
Database
Visualization Systems

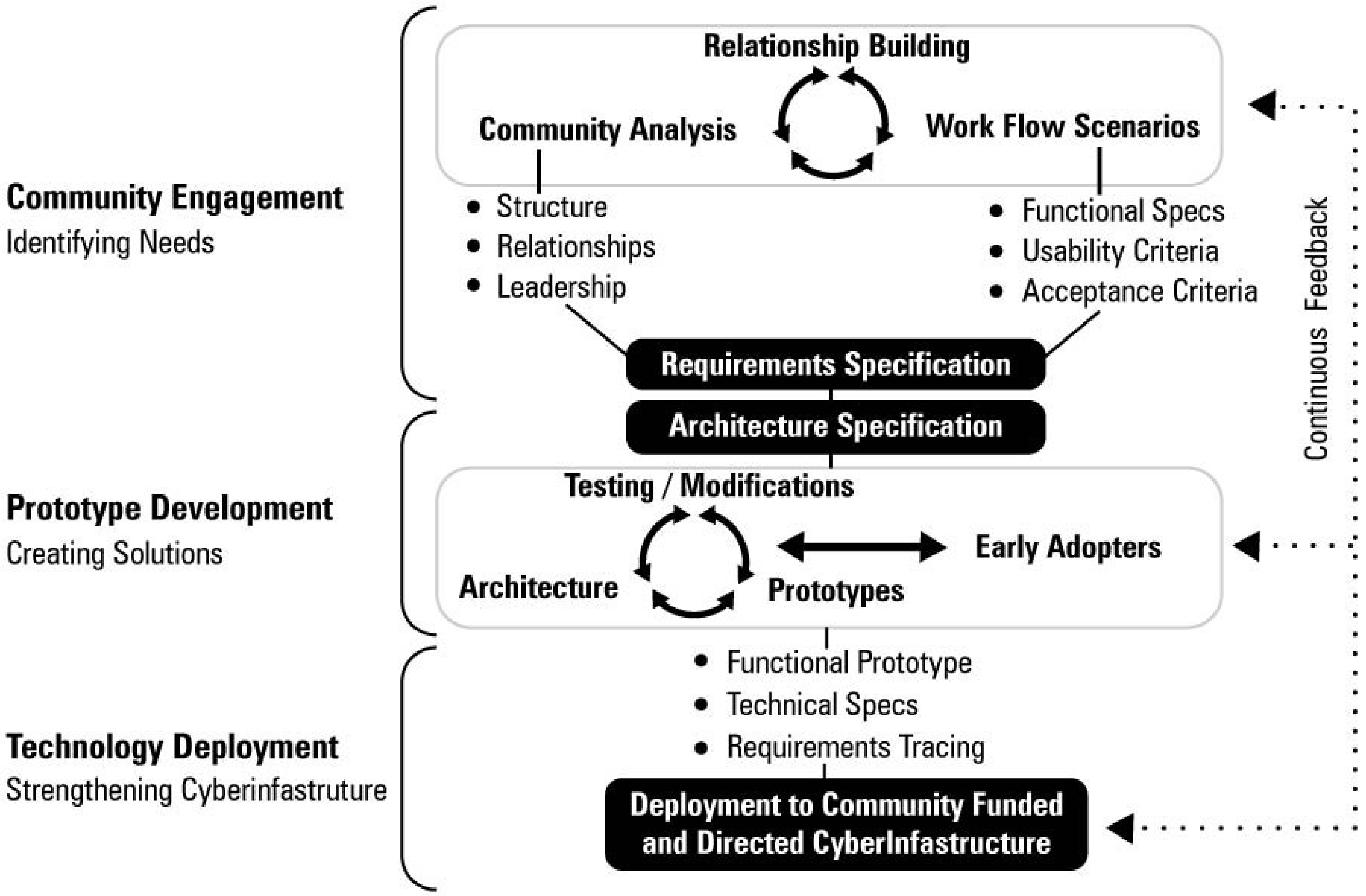
National Science Foundation

Workflow Component Sizes in Integration?

- Data sensing
- Data access
- Data management
- Data assimilation
- Simulation
- Data mining/analysis
- Data visualization

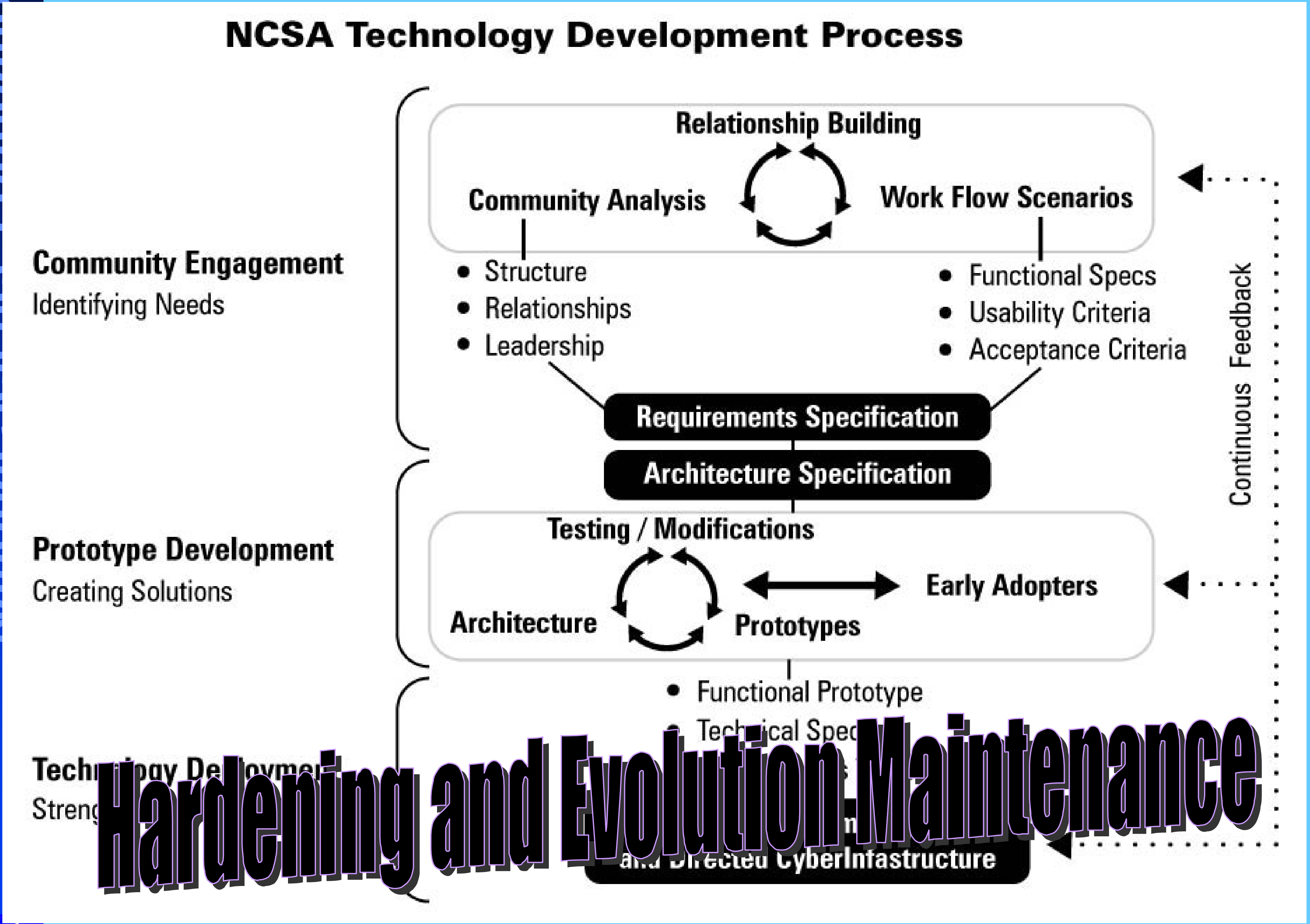
National Science Foundation

NCSA Technology Development Process



National Science Foundation

Hardening and Evolution Maintenance



Distribution

Randy Butler, NCSA
Stu Feldman, IBM
Miron Livny, UW-Madison
Matt Mathis, PSC

Cyberinfrastructure is more than software

- The Community
  - education
  - requirements
  - leadership
- Resource providers
- Simple services and schemas
- Policies; security, scheduling, etc
- Portals; that organize a set of community services
- Collaborative Environments/Virtual Organizations/Federations

Determination of when & what software to be distributed

- Has to be driven by “user” requirements where “users” = researchers & resource providers
- Successfully deployed and utilized in multiple instances
  - Instances within large CI projects
  - Bottom up adoption
- Clearly defined integration path with related software
- Clear model for support

Distribution Mechanisms

- Service providers
  - **Service providers for long term well integrated support**
  - **Rigorous testing and certified quality control**
  - **Integration and Dependency testing and documentation**
  - **Tightly packaged & straight-forward to install**
  - **Certified Repositories for integrated well supported and documented products that provide a roadmap through offerings**
- Software download
  - **Common Repositories for simple tools & services,**
  - **“new” unproven offerings with a best effort support model**

National Science Foundation

Licensing & Fees

- Licensing terms and conditions (e.g. user modifications)?
  - Open Source model is a must
  - Dependence already on commercial software
  - Key is how well components are supported & how exposed are the APIs

- Software is never free

Difficult Deployment Issues

- **Major policy issues with resource providers**
- **Security is a fundamental issue we still have not resolved**
  - **Identity management**
  - **Authorization services**
  - **Risks introduced by**
    - **Software**
    - **Relationships**
- **Legacy investments**

Deployment

- Standardization of policies, schema, and services at universities and labs
- Strong engagement process with resource providers and users
- Incentives for researchers and resource providers
- Novel site deployment & integration strategies
- “Small” number of supported operating systems
- Centralized services to lower the cost of entry

National Science Foundation

Centralized Services

- To help communities accept the software we need to lower the cost of entry barriers
- Lack of skills, time, resources, commitment
- Some suggestions
  - Certificate authorities - *should we fund DOE to do this?*
  - Attribute services & Directory services
  - Policy bridging
  - Software Repositories
  - Support Forums

Support

- Number of "general" solutions we can support
- Educating local support staff is critical
- Tools for customized configurations
  - Configuration management tools required
  - Support for this requires experts early on
- Integration group(s) provide clearing houses, testing, installation, configuration, and other types of support

"it's not good enough to provide bits. You have to provide rationally configured bits that ask the minimum of information from users to get a functioning system." *Phil Papadapolous*

Core Support

- Hard to develop training and support for a moving target – backward compatibility
- Clearly define “Core” set of tools and services (shared cyberinfrastructure), not an exhaustive list of what’s available.
- Generate uniform documentation and training materials
- Service providers to support local support teams
- Consider funding local support staff

Community Support

- Find community path finders and engage them as a guide and interpreter
- Understand community needs exploiting commonality & recognize uniqueness
- Work hard to deploy & demonstrate successes
- Applications drivers are a key, however there is a lack of understanding for the potential and therefore a lack of commitment
- Incentives for both users and resource providers

Summary

- CI does not have clear black and white boundaries
- We need to define, distribute and support a core set of services and tools
- Need to aggressively engage communities and coordinate these efforts
- We need to lower adoption barriers
  - Repositories
  - CAs
  - Directories
  - Policy standardization
  - Support & training centers

National Science Foundation

What approaches and processes should NSF employ to distribute cyberinfrastructure software?

- Determination of when and what software is to be distributed
- Service providers vs. software download?
- Should there be repositories (“certified” or common); should they be coordinated?
- Monolithic vs. fragmented distributions?
- Licensing terms and conditions (e.g. user modifications)?
- Free or charging?
- Mechanisms for and organization of support and training?
- How should support for deployment be handled?

National Science Foundation

NSF Workshop on Cyberinfrastructure Software

Session 7: Commercial Software

National Science Foundation

Session 7: Commercial Software

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

Commercial Software - Issues

- 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

National Science Foundation

Why Go Commercial?

- Why involve a commercial entity?
  - Existing products
  - Established processes, project management, staff
  - Infrastructure is a development and support activity, not classic research
  - Ability to support lifecycle
  - Record of success, competence
  - Paths to market

Why Go Commercial?

- Why would a commercial entity want to be involved?
  - Body shop (support for staff, with profit)
  - Fees for service
  - Product ownership and marketing value
  - Extending a product line to new market or scale, multiple uses
  - Service sale value
  - Larger marketing opportunity
  - Extend experience base
  - Scientific or speculative interest
- Basics
  - What is the value proposition?
  - Who takes the risks, who guarantees, who decides, who owns?
  - Think about Outsourcing and Offshoring

Why Go Academic?

- NSF always does it this way
- We are already very good at it (examples exist)
- We like doing it
- We want to extend the research model: learn to do development and teach our students
- We want to keep control of our child
- We are smarter and/or wiser
- The CyberInfrastructure domain is too complex, ill-defined, and uncertain

Commercialization Approaches

- Using or adapting existing software (COTS, GOTS, …)
  - Choosing, modifying
  - Lifecycle costs and requirements
  - Open source vs proprietary
    - Escrow, control, long-term promises
    - Site/national licenses, distribution, etc.

National Science Foundation

Commercialization Approaches

- Creating and managing new software commercially
  - Outsourcing, participation, other models
  - Arms-length development vs collaboration
  - Early vs Late stage
- Project management
  - Who owns what?
  - Guarantees of support (both sides)

Commercialization Approaches

- Moving research/experimental software into COTS
  - Encouragement, ownership, incentives
  - Ownership, certification, licensing, long-term usage rights
  - Lifecycle support, adaptation, replacement
- Moving services to commercial provider
  - Long term operations and support
  - Service Level Agreements etc.
- Outsourcing and Offshoring????

National Science Foundation

Commercial Software - Issues

- 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

National Science Foundation

Cross-Cutting Issues

Dan Atkins

Ian Foster

Rick Stevens

National Science Foundation

Session 8
Cross-cutting issues

What factors should determine the needed budget in transition and in the steady state?

How does NSF determine the relative budgetary needs of research, design, development, integration, and distribution?

How should NSF build up, how should it establish priorities, and what should it do first, and when?

Periodic evaluation of components and plan

International input and coordination

Overall process coordination

How does Cyberinfrastructure software feed into and enable deployment?

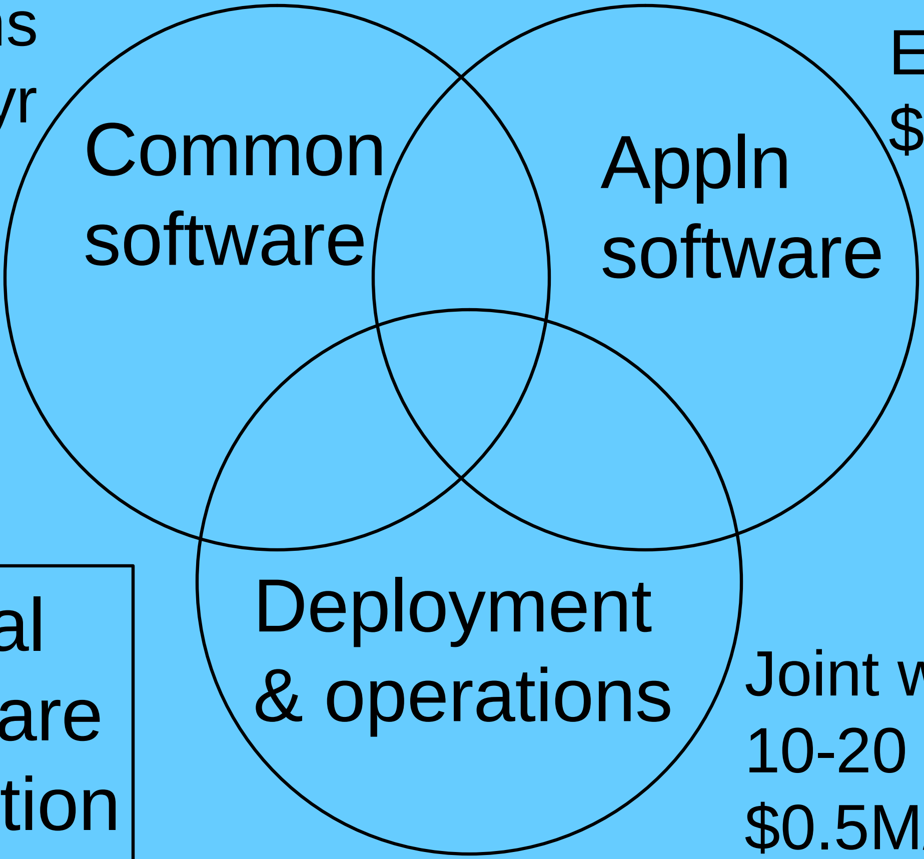
Role of NSF program managers (SCI and other directorates)?

More Specifically …

- The challenge: Achieving critical mass for software, by enabling two unnatural transitions
  - Software in small → software in large
  - Software → infrastructure
- What should a program look like to address this challenge, and to produce, deploy, and apply CI software?
- How do we make such a program attractive to other directorates?
- We'll make a concrete proposal

National Science Foundation

Program Structure



2-3 teams
$5-10M/yr

4-6 teams
Engage disciplines
$2-4M/yr

Joint with campus IT
10-20 campuses
$0.5M/yr per campus
Scale over time

National
CI Software
Coordination
Organization

“Waist” Software Teams (2-3, $5-10M each)

- A major source of core CI software
  - Focus on cross-cutting issues
  - Design—development—integration
  - Long-term support to develop expertise
  - Different teams address different subsets of required functionality, and/or competing approaches
- Requires strong feedback mechanisms from application and deployment groups

Application Software Teams (4-6 teams, $2-4M each)

- Purpose: engage major application communities
  - 4-6 enables cross-directorate engagement
  - Hopefully supplemented by other programs
- Specific purposes
  - Achieve adoption of advanced CI concepts and application successes
  - Encourage use of common infrastructure
  - Major source of input to “waist software”
  - Leverages substantial application R&D

National Science Foundation

Deployment & Support

- The NSFnet model: seed funding to universities to spur deployment of services
  - Local users &/or larger communities
  - Competition for participation & cost sharing
  - LHC "Tier 2" centers as contemporary example
  - Service based model presumably important
  - Integration: commercial, open source, NSF s/w
- Scaled approach
  - First, a few institutions with much expertise
  - Scale to larger number of sites that can clone recipes developed by early sites
  - Phase out once part of campus infrastructure
- Major source of input to "waist software" teams

National Science Foundation

National CI Software Coordination Organization

- Create new organization to support the work of the program
  - C.f. Advanced Networks & Services for VBNS
- Functions
  - Convene quarterly 3-day all-hands meetings for intense technical exchange
  - Coordinate requirements gathering
  - Provide administrative support
  - Reporting to NSF
- Modest budget needed for administrative support

Budget: Inputs

- Atkins report: $200M/yr for software
  - 10 centers at $5M = $100M
  - 50 projects at $2M = $100M
  - Includes applications, compilers, etc.: not just shared cyberinfrastructure
  - Estimate SCI is 1/4 of total, i.e., $50M
- Or, analysis based on total NSF budget: $6B
  - 10% for IT, i.e., $600M
  - 20% of that for SCI, i.e., $120M
  - 50% of that for SCI software, i.e., $60M

Budget Profiles

- Temporal
  - Ramp up is required, but otherwise need to think of CI software as long-term commitment
- Spatial (R, Design, Dev, Integration, Deploy?)
  - Research already covered by CISE
  - DDI best viewed as integrated activities and clearly the concern of CI Software effort
  - Deployment a combination of national CI (e.g., TeraGrid), university based (as above), and discipline specific (out of scope?)

Priorities

- Commit to a long-term activity, recognizing <u>human expertise</u> as the most critical resource
- Most pressing short-term issues are
  – Stabilize existing software development efforts to preserve & apply existing human expertise
  – Expand set of users to obtain broader input on requirements
- Then can look to recruit new expertise and address new needs
  – Expand set of software providers
  – Grow existing software development efforts
  – Start deployment activities

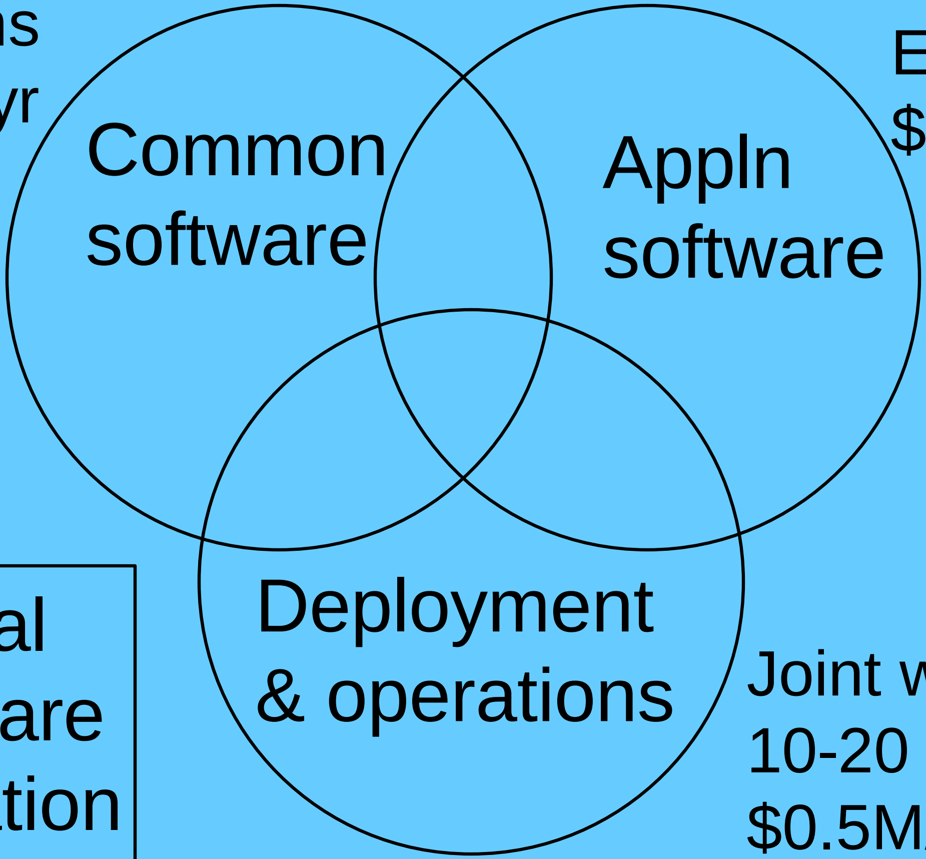
International Input & Coordination

- International coordination is critically important
  - Share effort of creating CI software
  - Enable interoperability for science projects
- Only one way to make it happen: joint projects
  - Real budget & joint deliverables
  - Political support from NSF & EU
- Challenge: double jeopardy
- Recommend:
  - 10% of overall budget allocated to joint projects with international partners

Coordination & Evaluation

- CI software coordination & communication organization can play a major role
  - Tight loop, intense technical interchange
  - If people know what others are doing at a deep level, divergence is at least deliberate
- Peer review is not the right approach to ongoing evaluation of CI software activities
  - Rather, provide a strong voice to the users: e.g., Unidata experience
- A vital role of NSF program managers is to facilitate cross-cutting inter-directorate work

Program Structure



2-3 teams
$5-10M/yr

4-6 teams
Engage disciplines
$2-4M/yr

Joint with campus IT
10-20 campuses
$0.5M/yr per campus
Scale over time

National
CI Software
Coordination
Organization