IMPROVING SCIENTIFIC SOFTWARE PRODUCTIVITY AND SUSTAINABILITY - THE IDEAS APPROACH

JUNE 23, 2017
ANSHU DUBEY
MATHEMATICS AND COMPUTER SCIENCE DIVISION
ARGONNE NATIONAL LABORATORY

OUTLINE

- **IDEAS Project Motivation**
- Productivity Challenges in Scientific Computing
- Community Best Practices
- How the IDEAS Project Helps

"... it seems likely that significant software contributions to existing scientific software projects are not likely to be rewarded through the traditional reputation economy of science. Together these factors provide a reason to expect the over-production of independent scientific software packages, and the underproduction of collaborative projects in which later academics build on the work of earlier ones."

Howison & Herbsleb (2011)

GENERAL STATE OF SCIENTIFIC CODES

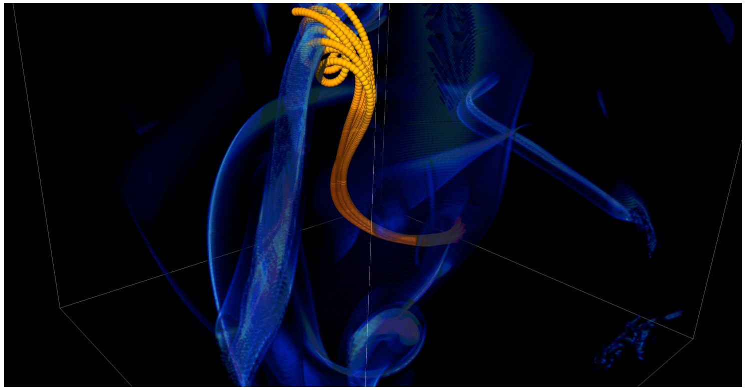
- Start in small groups
- Accretion leads to unmanageable software
- Parts of software may become unusable over time
- Inadequately verified software produces questionable results
- Increases ramp-on time for new developers
- In some cases process resets and starts over

Reduces software and science productivity due to technical debt

Technical debt – implementation without design and plan (quick and dirty) collects interest => more effort required to add features later.

Debt can compound

EXAMPLE



- In 2005 BG/L was made available at short notice
- Quick and dirty development of particles
- Many in-flight corrections of defects
- One was adding tags to track individual particles
  - **Got many duplicated tags due to round-off**
- Had to develop post-processing tools to correctly identify trajectories

We had a software process in place. The code was tested regularly. This was one instance when the full process could not be applied because of time constraints. We got ready for the run in less than a month, the run went for 1.5 weeks, and it took over 6 months before we could trust the processed results.

HEROIC PROGRAMMING

Usually a pejorative term, is used to describe the expenditure of huge amounts of (coding) effort by talented people to overcome shortcomings in process, project management, scheduling, architecture or any other shortfalls in the execution of a software development project in order to complete it. Heroic Programming is often the only course of action left when poor planning, insufficient funds, and impractical schedules leave a project stranded and unlikely to complete successfully.

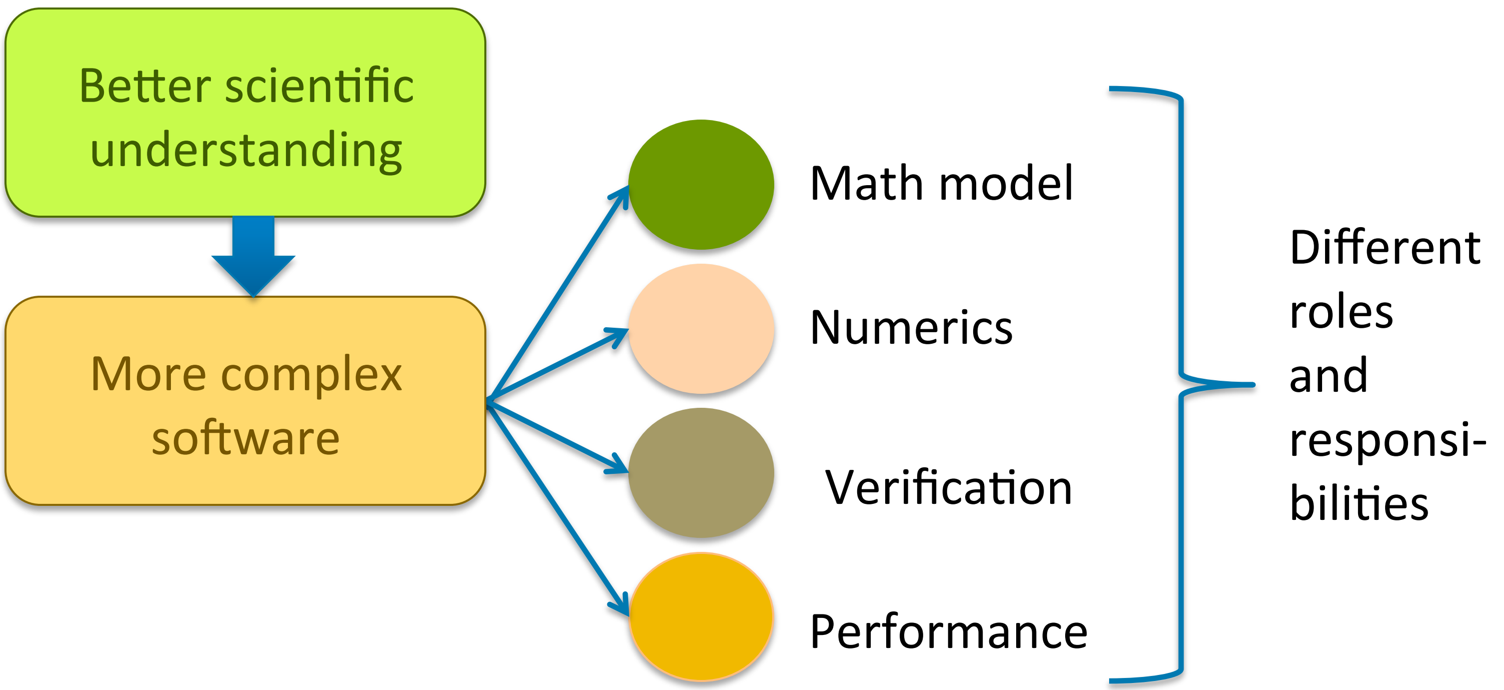
From http://c2.com/cgi/wiki?HeroicProgramming

Science teams often resemble heroic programming

Many do not see anything wrong with that approach

WHAT IS WRONG WITH HEROIC PROGRAMMING

Scientific results that could be obtained with heroic programming have run their course, because:

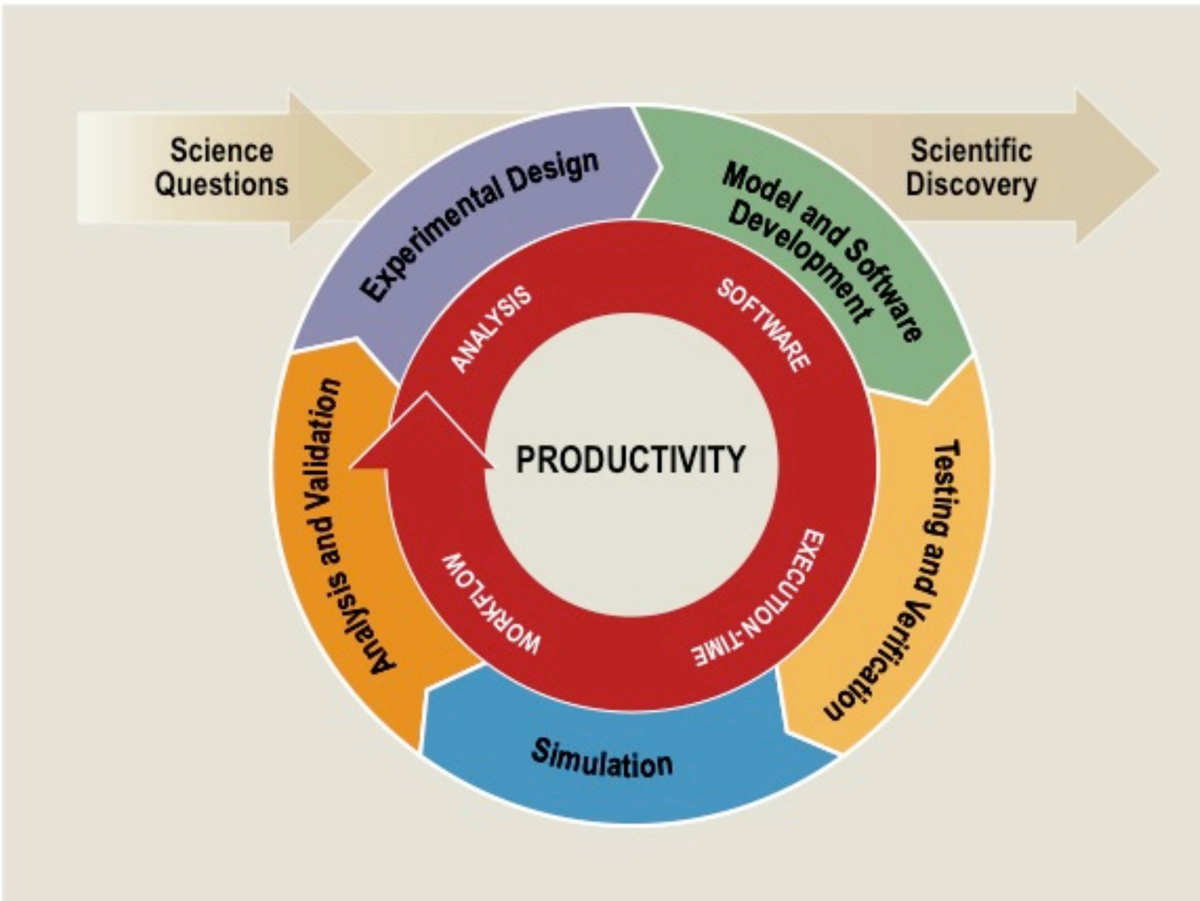


It is not possible for a single person to take on all these roles

LOOKING TOWARD FUTURE

- Codes aiming for higher fidelity modeling
  - More complex codes, simulations and analysis
  - Numerous models, more moving parts that need to interoperate
  - Variety of expertise needed – the only tractable development model is through separation of concerns
  - **It is more difficult to work on the same software in different roles without a software engineering process**
- Onset of higher platform heterogeneity
  - Requirements are unfolding, not known apriori
  - **The only safeguard is investing in flexible design and robust software engineering process**

THE SCIENTIFIC PROCESS



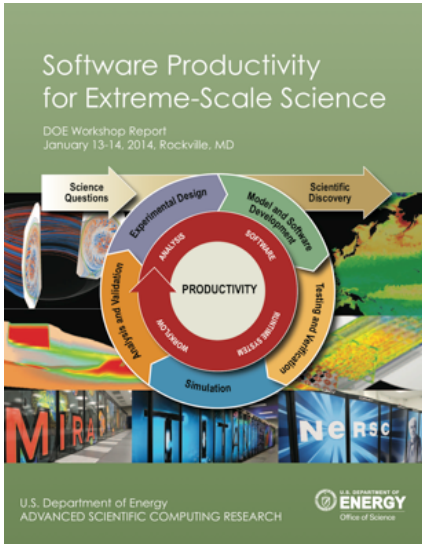
INTEROPERABLE DESIGN OF EXTREME-SCALE APPLICATION SOFTWARE (IDEAS)

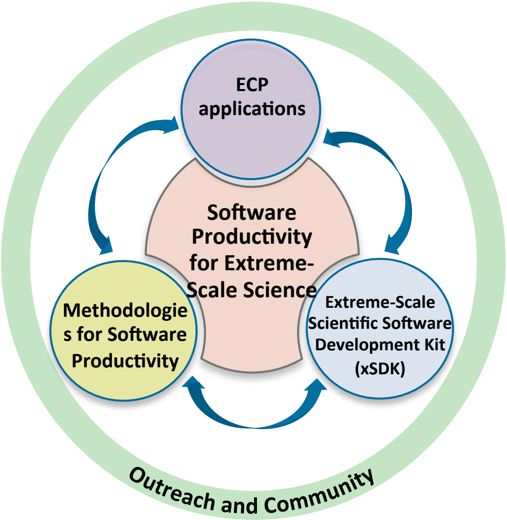
IDEAS productivity

Motivation

Enable ***increased scientific productivity,*** realizing the potential of extreme- scale computing, through ***a new interdisciplinary and agile approach to the scientific software ecosystem***.

Scientific Productivity is concerned with measuring the number and quality of science results for a research team over a span of time.





Objectives

Address confluence of trends in hardware and increasing demands for predictive multiscale, multiphysics simulations.

Respond to trend of continuous refactoring with efficient agile software engineering methodologies and improved software design.

Approach

Partnership with application teams ensures delivery of both crosscutting methodologies and metrics with impact on real application and programs.

Interdisciplinary multi-lab team

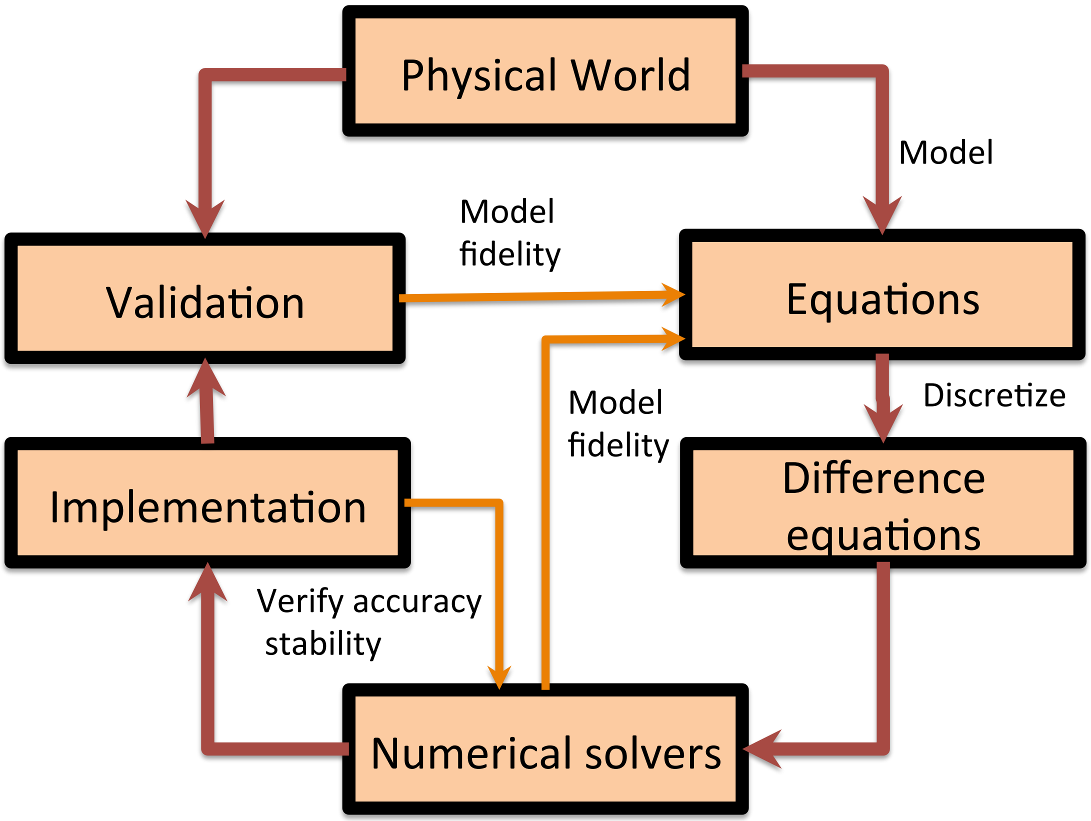
www.ideas-productivity.org

U.S. DEPARTMENT OF ENERGY
Office of Science

OUTLINE

- ❑ IDEAS-ECP Project Motivation
- ❑ **Productivity Challenges in Scientific Computing**
- ❑ Community Best Practices
- ❑ How the IDEAS Project Helps

LIFECYCLE



- ❑ Modeling
  - ❑ Approximations
  - ❑ Discretizations
  - ❑ Numerics
    - ❑ Convergence
    - ❑ Stability
- ❑ Implementation
  - ❑ Verification
    - ❑ Expected behavior
  - ❑ Validation
    - ❑ Experiment/ observation

GENERAL CHALLENGES

Technical

- ❑ All parts can be under research
- ❑ Knowledge growth => change in requirements
- ❑ Real world is messy, so is the software

Sociological

- ❑ Competing priorities and incentives
- ❑ Limited resources
- ❑ Perception of overhead without benefit
- ❑ Interdisciplinary interactions

VALIDATION CHALLENGES

- Interdisciplinary
  - Domain knowledge
  - Applied mathematics
  - Software engineering
- Exploring uncharted territories
  - Existing knowledge is of limited interest
  - Need to push the boundaries
  - The behavior of solvers not always predictable in regimes of interest

VERIFICATION CHALLENGES

- Inadequate granularity definition
  - Especially in composable codes
- Code coverage gives incomplete picture
  - Interoperability coverage as important
- Legacy components
  - No existing tests of any granularities
  - Examples – multiphysics application codes that support multiple domains

TESTING CHALLENGES

- Testing needs differ
  - Extent and granularity
  - Degree of formalization
  - Floating point issues
    - Different results
      - On different platforms and runs
      - Ill-conditioning can magnify these small differences
        - Final solution may be different
      - Number of iterations may be different
  - Unit testing
    - Isolating behavior can be difficult

OUTLINE

- ❑ IDEAS-ECP Project Motivation
- ❑ Productivity Challenges in Scientific Computing
- ❑ **Community Best Practices**
- ❑ How the IDEAS Project Helps

BASELINE

- ❑ Invest in software design
- ❑ Use version control and automated testing
- ❑ Institute appropriate verification and validation regime
- ❑ Define coding and testing standards
- ❑ Clear and well defined policies for
  - ❑ Auditing and maintenance
  - ❑ Distribution and contribution
  - ❑ Documentation

Customize process for project needs

CONSIDERATIONS FOR CUSTOMIZATION

- There is no “all or none”
  - Focus on improving productivity
  - Minimize bias
- Fine balance between buy-in and imposition
  - Show benefit to convert
    - Overcome resistance to change
    - Allay suspicion of new processes

EVALUATE PROJECT NEEDS

- Project Objectives
  - Proof of concept
  - Limited research use
  - Library
  - Production
- Team
  - Number of developers
  - Background of developers
  - Geographical spread
- Project Lifetime
  - New code versus some legacy components
- Complexity
  - Number of modules, models, data structures, solvers
  - Degree of coupling and interoperability requirements
- Lifecycle stages

INTERDISCIPLINARY INTERACTIONS

A partnership model that works

- Science users treat the code as a research instrument that needs its own research
- Developers and computer scientists interested in a product and the science being done with the code
  - Helps to have people with multidisciplinary training
- Comparable resources and autonomy for the developers
  - And recognition of their intellectual contribution to scientific discovery
- Careful balance between long term and short term objectives

DESIRABLE PROCESSES

- ❑ Project management methodology
- ❑ Provenance and reproducibility
- ❑ Lifecycle management
- ❑ Open development and frequent releases

OUTLINE

- ❑ IDEAS-ECP Project Motivation
- ❑ Productivity Challenges in Scientific Computing
- ❑ Community Best Practices
- ❑ **How the IDEAS Project Helps**

RESOURCES FOR SOFTWARE PRODUCTIVITY & SUSTAINABILITY—KEY ELEMENT OF OVERALL SCIENTIFIC PRODUCTIVITY

IDEAS productivity

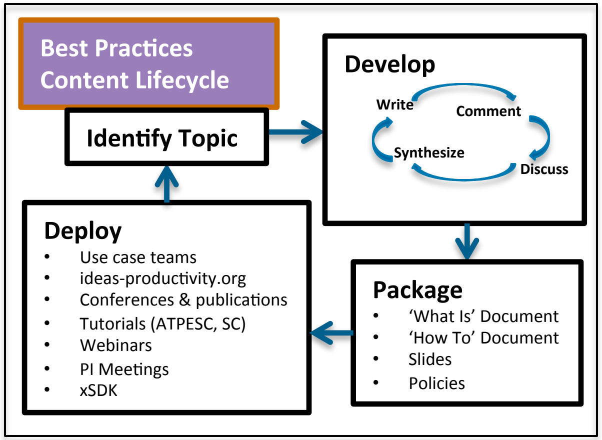
Approach: Collaborate with the community to curate, create, & disseminate software methodologies, processes, and tools that lead to improved scientific software

Modern learning theory:

Build from knowledge base: Elaboration and models

Vast body of SE content from broad community

Learn, adapt, adopt, assimilate



Webinars

- *Webinars 2016* *https://www.olcf.ornl.gov/training-event/webinar-series-best-practices-for-hpc-software-developers/*
- *2017 series started - https://ideas-productivity.org/events/hpc-best-practices-webinars/#scheduled-webinars*

***What Is and HowTo* docs**: brief sketches of best practices

- *What Is CSE Software Productivity?*
- *What Is Software Configuration?*
- *How to Configure Software*
- *What Is Performance Portability?*
- *How to Enable Performance Portability*
- *What Are Software Testing Practices?*
- *How to Add and Improve Testing in a CSE Software Project*
- *What Is Good Documentation?*
- *How to Write Good Documentation*
- *What Are Interoperable Software Libraries?*
- *What Is Version Control?*
- *How to Do Version Control with Git*

[More topics under development]

UNDER DEVELOPMENT: NEW WEB-BASED HUB FOR SCIENTIFIC SOFTWARE IMPROVEMENT EXCHANGE

better scientific software

Front-end coming summer 2017

BSSw Software Platform

Component Technology	Backend		Frontend
	Google Docs	GitHub	Ruby on Rails
Location	Google Drive	https://github.com/betterscientificsoftware	https://betterscientificsoftware.io (coming summer 2017)
Purpose	• Rapid collaborative content development • Multi-user typing, suggested edits, comments	• Content creation, refinement, management (from Google Drive) • Content packaging for use with BSSw.io	• User-facing portal • Polished backend content • Blogs, forums • Mailing lists
Contributors	Community content experts	Community content experts, BSSw staff	BSSw staff, web development experts
Consumers	BSSw GitHub Backend	BSS Frontend	CSE community
Content Notes	Content migrates to GitHub after it stabilizes	Content is managed in git repos, markdown	Content from backend, community

Contribute! Share your insights on CSE software practices and processes:

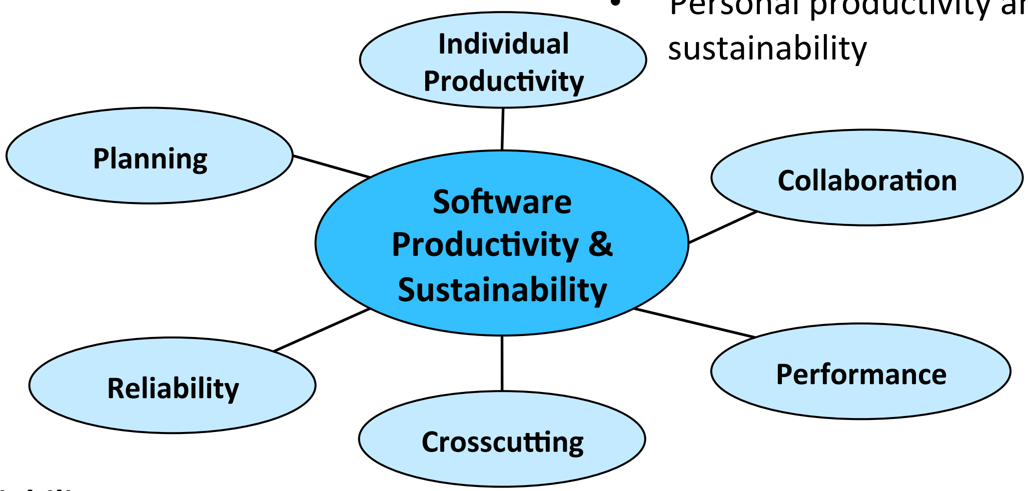
- https://github.com/betterscientificsoftware/betterscientificsoftware.github.io/blob/master/README.md
- Or search "github betterscientificsoftware"

RESOURCE TOPICS

better scientific software

Key:
blue text: covered in CSE17 tutorial
Black text: pointers to other resources

Planning:
- Requirements
- Design
- Development
- Configuration and builds
- Deployment
- Legacy code
- Refactoring



Individual Productivity:
- Personal kanban
- Individual learning plans
- Personal productivity and sustainability

Collaboration:
- Version control
- Licensing
- Strategies for more effective teams
- Documentation
- Issue tracking

Performance:
- Performance portability
- Software interoperability
- High-performance computing

Reliability:
- Testing
- Continuous integration testing
- Reproducibility
- Debugging

Crosscutting:
- Projects and organizations
- Discussion forums, Q&A sites
- Software publishing and citation
- Funding sources and programs

IT IS EXTREMELY IMPORTANT TO RECOGNIZE THAT SCIENCE THROUGH COMPUTING IS ONLY AS GOOD AS THE SOFTWARE THAT PRODUCES IT