Community Building

Discussion Lead: Ross Walker

Scribe: Tim Cockerill

Participants:

- Jay Alameda
- George Biros
- Dennis Castleberry
- Tim Cockerill
- Cynthia Gibas
- Eric Heien
- Brandon Hill
- Kevin Jorrisen
- George Karypis
- Robert Kosara
- Frank Loffler
- David McQueen
- Abani Patra
- John Rehr
- Jay Roloff
- Sarom Sok
- Eric Van Wyk
- Vineet Yadav

Takeaways:

Developers

-- Mostly grad students/post-docs doing development with PIs guiding.

-- Grad students/post-docs have finite lifetime on the project, then move on.

-- Lack of consistent programming techniques, software not hardened/robust.

-- Need for training/summer school - software best practices, software engineering.

-- Support for software professionals - establish consistency, promote best practices.

-- Need a coordinated, ongoing approach for scientific software developers to share ideas.

-- Expanded program for industry collaboration - synergistic, noncompetitive projects and shared ideas (Amber/NVIDIA collaboration).

User Community sustainability

-- Challenge: users that don't understand coding, they blindly run software...

-- Funding agencies should provide support for making an application more user friendly to broaden participation

-- Develop training and education materials to grow the user base.

-- Maintain a user support service.

Questions:

- How can we build a community of developers?
    - Often users of scientific software become developers (e.g. Amber) as they modify/enhance the software for their particular research needs.
    - Scientific codes are often too complex and not commented - need to teach the users better coding techniques. Need standards and frameworks that can overcome the basic foundation that PIs are competing with each other. (soft standards)
    - In Bio, have users that want a blackbox solution, don't want to develop or understand code, but then report results that lack knowledge of the code.
    - Software engineering and design needs to be brought into this community of developers. There are no rewards for following standards, nor penalties for not - no motivation. The goal is to innovate and write papers - that's validation.
    - Bio has established data standards that must be followed to publish a paper.
    - Some communities are beginning to recognize computational science side and encouraging code repository.
    - Often "not enough time" to follow software engineering protocol. Implementing a solution to get the science done.
    - Need to test the standards - break them down into rules - see if they are adding value using quantitative empirical methods.
    - Bio community will set a standard and then engineer into the software such that users don't see the standard, they are compliant via the software.
    - NSF could encourage standards for a given science domain so the different simulation methodologies can be compared.
    - Need to do a survey of each science domain's needs in computational science and software development, and what is important to them. Look for commonality.
    - Why not study collaboration from a scientific standpoint, rather than working purely on the basis of intuition? How else do we resolve a disagreement of intuitions about what variables encourage collaboration? (dc)
    - If we do want to study collaboration scientifically, we should have an *operational definition* of it, i.e. something we can quantify. (dc)

- What are the career path issues?
  - Funding agencies' recognition that software sustainability is important, not just the initial innovations/science/research behind the software.
  - Support for hardening/robustness of software.
  - Graduate students/post-docs need to publish to succeed. Sustaining software is not in their best interest.
  - Challenge to get software professionals involved that do want to sustain software. These people cost more than grad students /post-docs.
  - At a certain size of the code, need software professionals - too much for a grad student.
  - Training and education - teaching programming techniques, software engineering protocols.
  - Workshops/summer schools on best practices software engineering. What is meant by best practices? Moving target as HPC technologies evolve - number of cores, GPUs, MIC, etc
  - Programming fundamentals - comment your code, etc - needs to be taught. Incentive? "clean up your room theory" 2 minutes every day, or 4 hours every 6 months.
  - Without comments a code cannot be sustained, will not have long life.
  - Documentation and testing are key best practices. Human-computing interaction, usability, intuitive interfaces.
  - Encourage publication of new developments in software/programming, not just the science.
  - We can use workshops and summer schools as opportunities to research the impact of the "best practices" taught. Say we have N guesses about what "best practices" are--we can test them against N groups to measure the impact and find how whether they are making a difference, and how much. (dc)
- What qualifications do we need to succeed?
  - Even if members of a community are technically well-qualified, they need special social skills to collaborate well with others in a technical field. What can we do on this end? Do we work this skills training in to standards or a given collaboration framework--and if we do, to what extent can we guarantee this kind of training is taken seriously? (dc)
- How can we sustain developer careers on grants that fund non-integral personnel lines? (and how can we avoid losing talented developers in the face of this challenge)?
  - Local institutions can set up a team of software professionals that are funded as a recharge center to assist with the grant-funded activity. (local investment is a key)
  - XSEDE provides limited-term expertise for projects that need professional assistance.
  - Need non-scientist programmers who want to code. Challenge in keeping these people - they can make 3x their academic salary in the private sector.
    - Otherwise succeeding in developing a high-solidarity community can help keep people who are concerned about whom they work with. When we hire, we might give special consideration to people who seek a supportive work environment (if we're able to provide one). (dc)
  - Competing with National Labs and the job security provided by their sustained funding model.
  - Can NSF provide long-term sustained funding for software? Can a software sustainability program survive long-term?
  - Need recognition that scientific software development is its own domain--its own community, journals, etc.
    - Why don't we start labeling it "software science," then, or take advantage of other buzzwords traditionally associated with research? We can use widespread perceptions associated with powerful research-oriented words to help convince people of our case. (dc)
  - Software is a foundation for research, and needs to be recognized as such.
  - Cross-domain developer interactions. Don't reinvent the wheel. Identify the areas of strong overlap and get them working together. Should the SSI's be charged with identifying these? If SSI's are domain-specific, need to ensure the cross-connect between SSI's.
- How can we work with industrial contributions to our efforts? Can we define integration points for other (perhaps closed-source) industrial /proprietary components?
  - Industry is motivated by sales volume, time-driven, and profitability.
  - Benefit is the relationship is collaborative, not competitive.
  - Potential downside to professor - poach your best people. This is upside for students.
  - Large companies have research divisions that you can collaborate with.
  - How do you get companies to invest or license your software for redistribution? Do you want this? How does Open Source impact interactions with industry and should NSF require Open Source in their solicitations? Are all institutions as savvy about Open Source as they should be?
  - Should NSF encourage more joint proposals with industry?
  - Balancing act - companies mostly do not want to publish.
  - Collaborations within your science domain are not as likely as collaborations with those not directly interested in your software (example - nVIDIA, Intel, etc interested in selling hardware to users of your software)
  - Industry perception - software is easy - don't need academia. "code-monkey"
    - This is a consequence of *out-group bias*--quite generally, people within a certain group (e.g. academia) hold biases against mutually exclusive groups. What can we do to minimize bias? Perhaps if we dispel the notion of mutual exclusion between the groups by delineating academic-industry collaborative groups (i.e. encourage them to think of themselves as part of their own group). (dc)
- What about social psychological barriers to collaboration? Several come to mind: (dc)
  - Consider *social anxiety*; it obviously hinders collaboration. Some students (like me) experience this, but it can occur in all age groups. The general treatment for any phobia or crippling fear is exposure therapy (expose the person to the feared stimulus, a little at first, then more until they can handle it). Sudden exposure to overwhelming amounts of the stimulus can exacerbate the anxiety. We need to be aware of both of these facts when we expose socially anxious people to social situations. (dc)
  -
  - *Social comparison theory* holds that people evaluate themselves based on the behavior or performance of those around them. This has interesting effects, as some tend to seek out others who perform low on a trait in order to maintain self-esteem. (Consider _pre-test affiliation_, where students of similar preparation levels chat about the test--those ill-prepared find comfort talking to others in a similar situation.) We can use this to
  - *Evaluation apprehension* plays a large role. Evaluation apprehension is simply the fear of being evaluated and deemed inadequate by others. Sadly, this can turn out to be a type of _self-fulfilling prophecy_ because evaluation apprehension can inhibit performance, which can in turn influence others' evaluations, which lower their expectations and continue the cycle. We can prevent this by encouraging a non-judgmental and welcoming attitude toward newcomers. (dc)
  -
  - *Out-group bias* between disciplines can hinder collaboration. Because chemists, physicists, biologists, etc. are grouped linguistically, many perceive themselves as being part of mutually-exclusive groups. This may lead to out-group bias which accompanies harmful stereotypes (e.g. "physicists are asocial and absent-minded", "psychologists are technically-illiterate"),

prejudice (hostile or indifferent feelings toward the other group) and discrimination (the behavioral manifestation of stereotyping and prejudice). One thing we can do to dispel this is be aware of labels: if we apply a label for our community but not for our domains of research, we can actually take advantage of *in-group bias*, which provides the kind of solidarity that sustains a community. We can e.g. bring to mind that we are all, at the core, scientists of one sort or another. (dc)

- Studies show that shy people tend to be more active on social networking sites such as Facebook. This is because (1) the Internet tends to be associated with anonymity but also because (2) what some shy people find undesirable about communication are not the verbal but nonverbal aspects of communication; on-line text communication helps them avoid it. We can use this fact to our advantage in seeking to build communities by using it occasionally as a buffer for the shy and socially squeamish. (dc)

- How can we build a community of users and/or adopters of our software?
  - How can we understand our user's needs?
    - As far as user-friendliness goes, we can rely on surveys. User-friendliness is based on the perceptions of the users, so when we ask for their self-reports, we can be relatively sure we are measuring the user-friendliness construct. (dc)
    - Create the impression that voicing opinions about the software is the norm. Put a few examples of user communications with the development team in a prominent place. (dc)
      As far as user-friendliness goes, we can rely on surveys. User-friendliness is based on the perceptions of the users, so when we ask for their self-reports, we can be relatively sure we are measuring the user-friendliness construct. (dc)
  - How do we ensure we deliver what users need?
    - Challenge: users that don't understand coding, they blindly run software...
  - How do we encourage the users to view the software as a research tool?
    - The users need to think for themselves and not just blindly run the application.
    - Label it so and neurolinguistic self-programming will take care of the rest. Consider the Einstein Toolkit ~~it's hard to mistake for anything but research software. If we can't change the name, we should make it part of the headline; if not that, then we should make it the first sentence of any introduction of the software. That holds true of anything we want people to believe about the software~~ if key ideas, words and phrases do not trail far from the utterance of its name, people will hold them in mind. (dc)
  - How can we assure continued existence and support for our user community?
    - Materials Science - have seen greater uptake when a web interface/GUI is designed to simplify use of the code - get away from command-line.
    - Should the funding agencies provide support for making an application more user friendly to broaden participation? Often we stick to new capabilities, not simplifying the user experience.
    - Develop training and education materials to grow the user base.
    - Maintain a user support service.
      - Aside from maintaining a user support service, we need to show that the support service is available and that its support team is able and willing to provide support. If the support service is available through the web, it should be in a prominent location (as opposed to an e-mail contact link in fine print at the bottom of a page). Show an example of how the support service was used to resolve an issue successfully and give clear instructions on how to use it. This is important--two reasons people don't use support services is because they (1) believe it won't solve their problem in a timely fashion and (2) are unclear about the support team's expectations and experience evaluation apprehension as a result. (dc)
  - What role can industry play in adopting our capabilities?
    - Example: MS HPC group approached GAMESS to develop a Windows port after realizing that GAMESS is used as a benchmarking code for HPC platforms.
  - How can we work with international partners to adopt our software?
    - We need multi-language support, but we need our partners to translate across cultural as well as linguistic barriers. When our partners voice their opinions about culture-specific user demands (e.g. interface preferences), we should try to understand and address those user demands in a culturally relativistic context. (dc)
  - What role can other agencies (and agency projects) play in building a community of users/adopters?
  - Presentation of the software plays an important role in attracting new users to the software because it influences their first impression, which shapes their general impression (according to the *primacy effect*). Even if the match between user needs and implementation were perfect, users won't use the software unless they can be convinced it is worth their while--and for that we have to focus on presenting the software well. (dc)
- How can we build a community of support organizations
  - How can we work with integrating organizations (such as Extreme Digital) to help support our software?
  - How can we work with Resource providers to help support our software?
  - What role can campus organizations play in supporting our software?

Domain	**# Developers**	**# Users**	**Comments/Codes Used**
Materials Science	100	10,000	10 or so codes, many European, each with 5-10 developers, user count is across all codes
Computational Comparative Genomics	100's	multi-10,000	Galaxy, Genome Browser, NCBI
Electronic Structure Theory	30	> 100,000 worldwide	developer count at Iowa State University, GAMESS/NWChem
MD	40	10,000	Amber deployed at 850 sites, 1500-1600 citations/yr (CHARMM, NAMD, GROMACS, LAMMPS etc similar in size)
GeoPhysics	10	200 - 300	
Numerical Mathematics		countless	finite element solvers, BLAS, R
Bioinformatics			BLAST

Notes: