The Challenges of Developing and Maintaining HEP 'Community' Software

Amber Boehnlein

April 14, 2014

U.S. DEPARTMENT OF ENERGY
Office of Science

SLAC NATIONAL ACCELERATOR LABORATORY

A Reminder

- Life is good!
- Our challenge is ‘How to make the Good Life better’

- Disclaimer:
  - » limited in scope
  - » No gap analysis
  - » personal opinion in nature
  - » Source material: white papers
- Outline
  - » What is community software
  - » Software life cycle
  - » General discussion of the challenges.
  - » Brief discussion of 5 successful examples of community software and the challenges that they faced or are facing
  - » Going forward

What is Community Software?

- Software that is developed and used explicitly for the common good
  - » Addresses functionality that is common to multiple scientific (or technical) organizations.
  - » Has a strong user driven component
  - » Developed ‘in-house’ to HEP/NP
- Communities occur at different scales; support occurs at different scales
  - » A community of developers may serve a community of scientists
  - » Some communities are small and relatively self-contained

- Much of today's implementations of community software has roots in the mid-late 1990s
  - Fertile period intellectually and financially
  - Dramatic evolution in hardware architectures and costs
  - Vibrant experimental program (HERA; LEP)
  - BaBar, Belle, Tevatron Run II, LHC pushing requirements
  - ILC design
- At the time, very difficult to predict winners and losers in software projects
  - Many joint projects
  - Many innovative ideas

Software Life Cycle

- Recognition of needs while understanding potential solutions
  - Project Model
  - Start-up Model
  - Adaption Model
- R&D/beta phase: Functionality first
- Adoption
  - Typically requires a effective champion to develop a user community
- Expansion Phase: Production while Developing
- Steady state operations
- Evolution
  - Many of our community tools have multi-decade life spans
- End of product life
  - one of the hardest challenges due to change costs

In-house Developed Computing Suite

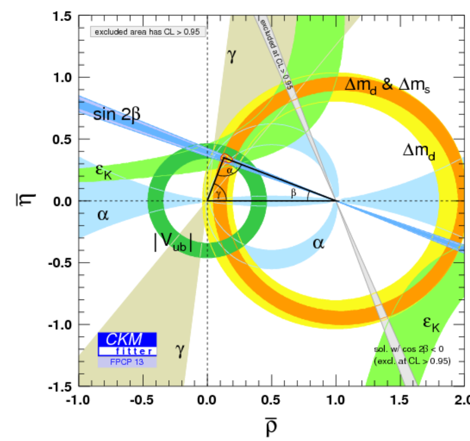
- Accelerator Simulations and Modeling Codes
- Simulations of fundamental particle interactions
- Particle transport codes
- Experiment specific code for data acquisition, triggering, reconstruction from detector hits to physical objects
  - » Conditions data
- Middleware stacks to handle all aspects of processing
- End User Analysis framework and tools
- Data Management and Preservation
- Field wide publication database (Inspire)
- On-line Atlas of combined results (Particle Data Group)

Examples

- Large Scale multi-year successful Community Software
  - Lattice QCD codes (Science Research)
  - INSPIRE (Information Technology)
  - Grid Projects (Underlying Technology)
  - Geant 4 (Toolkit for simulation)
  - Root (User Analysis framework)

Lattice QCD

- History
  - » Basic computation techniques developed in the ’70s and custom machines were built
  - » By the late 90’s systematic errors in key calculations around 10-20%- insufficient compared to experiment
- Formation of scientific collaborations in 2000
  - » Benefitted from dedicated software funding to develop common codes for gauge configurations
  - » Flagship measurements- systematic errors at 1% level
  - » Big consumers of HPC
  - » Continue strong relationship to hardware vendors
  - » Well developed and effective governance
- Challenges
  - » Must please many masters
  - » Insatiable demand for computing
  - » Career paths



INSPIRE

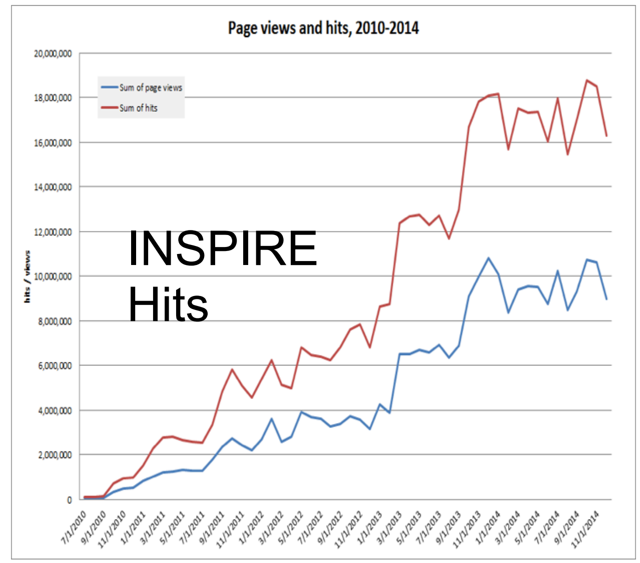
- INSPIRE is a free, open service that hosts comprehensive databases of HEP research
  - An effective international partnership between CERN, DESY, FNAL, IHEP and SLAC
- Much appreciated by the HEP community as a scientifically powerful tool
- Challenges
  - Overcame technical stagnation
  - Small support/development team for a large user base
  - Questions about relevance

INSPIRE Timeline

1969	development begins
1974	SPIRES-HEP database is released
...	
1991	First web site outside of Europe, first db on web
...	Functionality frozen; platform emulators
2007	search for new platform begins
2007	Partnership with CERN: Invenio platform selected
2009	development begins on INSPIRE
2010	beta release
2010	Explicit DOE funding/Advisory Board Charged
2011	Simultaneous development and production
2012	SPIRES front end deactivated
2013	INSPIRE best practices Dev/OPs
2015	Upgrading--

Survey: Information Services for Professional Research

	daily	weekly	monthly	less than monthly	rarely or never
Google	64.20%	19.70%	6.50%	4.90%	4.70%
arXiv	48.70%	29.90%	11.60%	6.00%	3.90%
INSPIRE	29.50%	27.70%	17.30%	12.10%	13.40%
publisher sites (ScienceDirect, etc.)	7.40%	36.20%	26.30%	17.50%	12.60%
university libraries and repositories	3.00%	16.10%	24.80%	32.90%	23.20%
Google Scholar	4.30%	13.70%	16.10%	20.10%	45.90%
other indexing services (Scopus, etc.)	1.30%	6.20%	12.90%	27.30%	52.30%
CDS	6.20%	9.60%	4.40%	6.60%	73.10%
ADS	4.70%	6.20%	5.10%	10.10%	73.90%
Other	2.50%	4.50%	7.30%	12.20%	73.50%
JACoW	0.40%	1.90%	2.10%	5.00%	90.70%



- Google's use now virtually ubiquitous for *all* net users
  - Some use likely more "casual" than INSPIRE
- The pie appears to have grown!
  - More use of information services in 2015 than 2007?
- Incredible user support for the unique services provided by INSPIRE
- You can help! Sign up for a ORCID and link it to your HEPnames

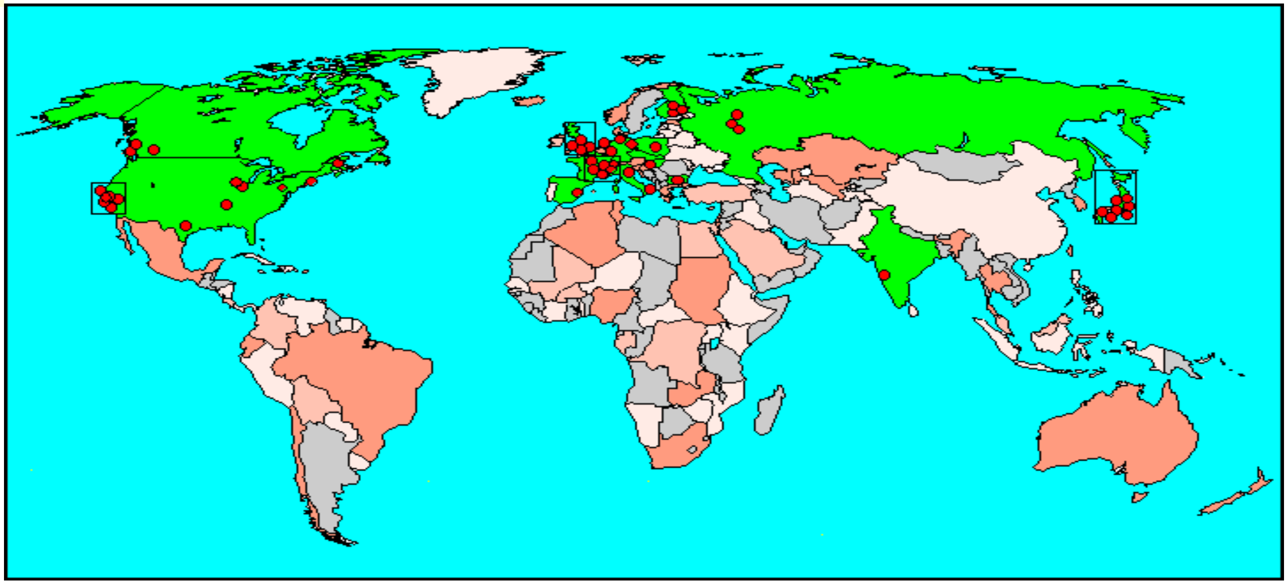
Community Tools: Grid Compute Projects

- To support the LHC distributed model, grid computing projects intended to support multiple experiments and disciplines
- WLCG-EGEE & Open Science Grid
  - » Conceived roughly 1999 with funding around 2002
  - » Largely targeted towards access to large scale compute resources
  - » Pushed boundaries for the better: cyber security
- Challenges
  - » Maintaining Funding: Soft money/essential infrastructure.
    - Continual re-invention and restructuring required
  - » Maintaining technical currency (See additional talks)
  - » Communities beyond LHC

Geant 4: Monte Carlo Transport Code

- G4 has the most comprehensive suite of physics processes and particle types
  - Tested and benchmarked over for 30 years
  - The complexity of geometrical descriptions leads to realistic representations.
- Geant 4 is effectively an exo-cortex of the field of particle physics
- Software Collaboration with a well defined governance model including user technical forum
- Extremely successful for the LHC physics program
- In wide use outside of HEP for aerospace and medical applications

Geant4 Collaboration



TRIUMF

Lebedev

J.W.Goethe Universität

Collaborators also from non-member institutions, including
Budker Inst. of Physics
IHEP Protvino
MEPHI Moscow
Pittsburg University

Geant 4 Timeline

1974	GEANT
1982	GEANT3 in production
1992	Two exploratory projects (CERN & KEK) focused on OO
1993	P58 submitted
1994	RD44 begins
1998	Geant 4 Beta Release
1998	1.0 production release
1999	RD44 ends and G4 collaboration is formed
2000	Collaboration expands to 100 members
…	BaBar Adopts; intense focus on LHC program
2010	Multithreading R&D
2013	V10.0 release
2014	Adoption of V10.0 by CMS; v10.1 released

Geant 4: Challenges

- Need for Speed…. Evolution to adapt to changing architectures
- Expanding the physics models – more physics, more accurate physics and faster physics
  - The Neutrino Community is very quick to point out that they are underserved by the current physics processes in G4
  - Prioritizing the needs across multiple physics communities
- Refreshing developer community and providing career paths

- Evolution—two proposed paths
  - Geant 4 roadmap: C++11/C++14
  - GEANT V R&D project ~ 2012

ROOT

- Physics Analysis Workstation (PAW): 1984
- ROOT: OO/C++ reimplementation: 1995
- It is the ultimate in-house developed community software
  - » ROOT is the tool that has conditioned us as a community
  - » no other science domains will adopt
- Comprehensive in services provided
  - » I/O through to event display
  - » C++ interpreter
- Governance: CERN Lead Project, FNAL collaboration
  - » Development team remains strong and active
- Challenges
  - » Evolving architectures
  - » Enabling and isolating at the same time

Recap of the general lessons

- Note the time scales
  - Prototyping to production is typically 10 years
- Many different governance models can be effective
- Resources are always limited
  - Too few people doing too much
  - Never enough hardware
  - Increasing scrutiny about ‘commercial’ solutions
- Staying connected to the community and its needs
  - Inattention to community software has led to gaps in the portfolio
  - Burgeoning needs in the neutrino community/Direct Dark Matter
- Providing intellectual and technical continuity
- Managing the projects through the full life cycle
  - Enabling R&D
  - Fostering innovation—what are the next great ideas?
  - Where can and should we move on?

Meta Challenges

- Long term international projects
  - » Community/Common software must become the norm.
- Community software can be recognized as ‘community assets’
- HEP Software Foundation (http://hepsoftwarefoundation.org)
  - » Mechanism to foster coordination of common efforts in HEP software and computing
  - » Has been growing by leaps and bounds
  - » Governance: Do-ocracy
  - » Join BOF on Friday afternoon!

Beyond HEP Software Foundation

- What distinguishes ‘Community Software’ is the need to serve a community
- Currently, prioritizing the resources for community software is piecewise
  - » Lab-centric
  - » Experiment-centric
- This leads to
  - » Unnecessary duplication
  - » Gaps in the portfolio of community tools
  - » Difficult to start anything new
- “Portfolio Management” can be lightweight
  - » Experience from the LHC experiments
  - » Does require engagement from a variety of stakeholders

Conclusions

- HEP has developed many successful community tools and sustained them over many years
- HEP Software Foundation is a general recognition that more coordination to foster collaboration is necessary and desirable
- Leveraging the HSF, the HEP community should broadly work together to foster and prioritize forward going common and community projects