What is software sustainability anyway?

NSF SI2 PI Meeting, 17-18 January 2013
Neil Chue Hong (@npch)
N.ChueHong@software.ac.uk
What do I do?
Why is this important?

A video interlude

Featuring a frustrated panda

http://tinyurl.com/datasharingpanda
No one starts writing software intending to abandon it completely
• “Sustainability is not an issue for researchers, we want others to take the software over subject to IP issues”

• “Sustainability is a big issue, we are producing complex tools we want to continue to use but it’s not clear how they’ll be sustained”

- Quotes taken from ENGAGE interviews
Software is no longer easy to define, let alone sustain
Authorship

http://beyond-impact.org/?p=175

www.software.ac.uk

• Which authors have had what impact on each version of the software?

• Who is responsible for each component when authors leave?
What do we choose to keep:
- Workflow?
- Software that runs workflow?
- Software referenced by workflow?
- Software dependencies?
What’s the minimum citable part?
Granularity

Algorithm

Library / Suite / Package

Program

Function
Why do we version?
- To indicate a change
- To allow sharing
- To confer special status
What do we sustain:
- Map?
- Software that creates map?
- Software that uses map?

Novel reuse of public sector data
http://www.mysociety.org
Sustainability in Context

Software Sustainability Institute

- Software Engineering
- Market Development
- Funding/Effort
- Support/Contributions
- Community Engagement
- Product Management
Sustainable Communities

• Cohesion and Identity: Creating a community
• Tolerance and Diversity: Smart growth through collaboration
• Efficient use of resources: Leveraging infrastructure
• Adaptability to change: Governing sustainably
How do you approach software sustainability?
Sustainability Challenges

• Software is increasingly transient
  ▪ Cloud computing and workflows

• Software often has complex dependencies
  ▪ Sensitive to changes in its environment
  ▪ May require expert knowledge outside of the team

• There may be many variations in many versions
  ▪ What are you seeking to sustain?
Why are you considering software sustainability?

Purpose

- Achieve legal compliance
- Create heritage value
- Enable continued access to data
- Encourage software reuse
- Manage systems and services

http://www.software.ac.uk/attach/SoftwarePreservationBenefitsFramework.pdf
How are you going to choose the right approach?

- Preservation (techno-centric)
- Emulation (data-centric)
- Migration (functionality-centric)
- Transition (process-centric)
- Hibernation (knowledge-centric)

Software Sustainability Institute
Software Sustainability: preservation vs sustainability

Sustainability?

Image courtesy of London Permaculture under CC-by-nc-sa license

Preservation?

Image courtesy of RGB Kew – not for reuse

Software Sustainability Institute
• There are several approaches we have identified that could be classed as “sustainability”

• The choice depends on a number of factors, which change through time

http://www.software.ac.uk/resources/approaches-software-sustainability
Exploiting software for sustainability

**Models**
- Grant Mosaic
- Institutional support
- Fully Costed Service
- External Enterprise / Consultancy
- Royalties and Fees
- Donations
- Advertising
- T-shirt (spinoff merchandising)

**Vehicles**
- University based
- Spin out company
- Consultancy and Customisation
- Industrial knowledge transfer
- Contracts
- Licensing
- Certification
- Support services / training
- Software as a Service
- Software Foundation

Software Sustainability Institute

www.software.ac.uk
Software sustainability should be part of the process

- Comparable to risk management
  - No one right “solution” but many examples of best practice and process
- Plan from before the start if possible
  - But must be reviewed regularly

- No longer considering timescales bounded by a *project*, but considering the *product*
Software sustainability is the ability to continue to use, support, maintain, and evolve software.
The word “sustainable” is unsustainable.

2036: “Sustainable” occurs an average of once per page.

2061: “Sustainable” occurs an average of once per sentence.

2109: All sentences are just the word “sustainable” repeated over and over.

Frequency of use of the word “sustainable” in US English text, as a percentage of all words, by year.

Source: Google ngrams

From xkcd.com
<table>
<thead>
<tr>
<th>Project Title</th>
<th>Value</th>
<th>Start</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>Centre for Plant Integrative Biology</td>
<td>£3,112,336</td>
<td>Feb 07 - Oct 12</td>
<td></td>
</tr>
<tr>
<td>Centre for Systems Biology at Edinburgh</td>
<td>£3,108,009</td>
<td>Jan 07 - Dec 12</td>
<td></td>
</tr>
<tr>
<td>DAASE: Dynamic Adaptive Automated Software</td>
<td>£834,903</td>
<td>May 12 - May 18</td>
<td></td>
</tr>
<tr>
<td>OMU-UK Centre and Managed Programme</td>
<td>£9,718,373</td>
<td>Jun 06 - Dec 10</td>
<td></td>
</tr>
<tr>
<td>Natural Speech Technology</td>
<td>£9,336,104</td>
<td>Apr 11 - Apr 18</td>
<td></td>
</tr>
<tr>
<td>EPSRC Centre for Innovative Manufacturing in Through-life Engineering Services</td>
<td>£5,787,928</td>
<td>Jul 11 - Jul 18</td>
<td></td>
</tr>
<tr>
<td>HUMAN-AGENT COLLECTIVES FROM FOUNDATIONS TO APPLICATIONS (FORCHIC)</td>
<td>£5,492,996</td>
<td>Jan 11 - Dec 15</td>
<td></td>
</tr>
<tr>
<td>LSBS-ITPS: Large-Scale Complex IT Systems Initiative - Research Programme v2</td>
<td>£3,065,294</td>
<td>Jun 07 - Jun 13</td>
<td></td>
</tr>
<tr>
<td>EPSRC Centre for Innovative Manufacturing in Advanced Metrology</td>
<td>£4,923,929</td>
<td>Mar 06 -</td>
<td></td>
</tr>
<tr>
<td>Endermological Laboratory and IT Development Area</td>
<td>£4,790,398</td>
<td>Aug 11 - Aug 16</td>
<td></td>
</tr>
<tr>
<td>EPSRC Centre for Secure Information Technologies (CSIT)</td>
<td>£4,560,903</td>
<td>Mar 08 - Feb 14</td>
<td></td>
</tr>
<tr>
<td>Numerical Algorithms and Intelligent Software for the Evolving HPC Platform</td>
<td>£4,557,814</td>
<td>Jul 09 - Jul 14</td>
<td></td>
</tr>
<tr>
<td>EPSRC Centre for Innovative Manufacturing in Through-life Engineering Services</td>
<td>£4,364,781</td>
<td>May 10 - May 15</td>
<td></td>
</tr>
<tr>
<td>EPSRC Centre for Innovative Manufacturing in Through-life Engineering Services</td>
<td>£4,340,018</td>
<td>Oct 12 - Oct 16</td>
<td></td>
</tr>
<tr>
<td>Analysis and prediction of protein structure and influence</td>
<td>£4,131,837</td>
<td>Mar 08 -</td>
<td></td>
</tr>
<tr>
<td>A new approach to Science at the Life Sciences Interface</td>
<td>£3,869,307</td>
<td>May 11 - May 15</td>
<td></td>
</tr>
</tbody>
</table>
A national facility for cultivating world-class research through software

• Better software enables better research
• Software reaches boundaries in its development cycle that prevent improvement, growth and adoption
• Providing the expertise and services needed to negotiate to the next stage
• Developing the policy and tools to support the community developing and using research software

Supported by EPSRC
Grant EP/H043160/1
SSI: Long Term Goals

- Provision of useful, effective services for research software community
- Development and sharing of research community intelligence and interactions
- Promotion of research software best practice

• **Mantra:**
  - Keep the software in its respective community
  - Work with the community, to increase ability
  - Don’t introduce dependency on SSI as the developer
  - Expand and exploit networks and opportunities

Software Sustainability Institute
SSI Drivers and Themes

• Two key drivers which cause people to seek the SSI’s advice:
  ▪ They want to be more productive in their research
  ▪ They don’t want to be embarrassed by appearing worse than their peers

• Broadly, our work falls into a few key themes:
  ▪ Developing the scientific computing / software development skill base
  ▪ The role and reward of software in research
  ▪ Recognition of software career paths
  ▪ Reproducible research
SSI Organisation

- Community Engagement (Shoaib Sufi)
  - Fellowship Programme
  - Events and Roadshows
- Consultancy (Steve Crouch)
  - Open Call for Projects
  - Software Evaluation
- Policy and Publicity (Simon Hettrick)
  - Guides and Case Studies
  - Best Practice and Policy
- Training (Mike Jackson)
  - Software Carpentry
  - Software Surgeries

- Collaboration between universities of Edinburgh, Manchester, Oxford and Southampton. 9.5 FTEs for 5 yrs supplemented by additional project funding.

Software Sustainability Institute
SSI Process

- **Weekly**
  - Development calls (project staff)
  - Management calls (investigators + senior staff)

- **Monthly**
  - Outreach calls
  - Community Engagement calls

- **Quarterly**
  - Mgmt/Ops F2F meetings

- **Half-yearly**
  - Advisory Board meetings

---


Software Sustainability Institute
Initially, two types of project

- Development Projects
  - 3-6 months duration
  - > 3 staff-months funding (either SSI or external)

- Consultancy Projects
  - Short, intermittent interactions
  - Perhaps over extended timescales
  - Both SSI and external funding
• A group of “software-savvy” researchers from all disciplines
• Agents connect their research discipline to the Institute
• Provide conference intelligence and more
  ▪ In return for travel expenses and career progression opportunities
• http://www.software.ac.uk/agents
• Agents from chemistry, energy, biomechanics, molecular sequencing, coastal engineering, medicine, semantic web, software development, environmental change, glacial geomorphology, bioinformatics

• Conference Intelligence reports:
  ▪ 3D Body Scanning Technologies
  ▪ Terra Cognita and International Semantic Web Conference
  ▪ Wellcome Trust Conference on Genome Informatics
  ▪ Digital Engagement
  ▪ ATLAS.ti workshop

• Ended up being motivated advocates for better software

Software Sustainability Institute
The Craftsperson and the Scholar

By James Hetherington, Research Software Development Team Leader at University College London

At Digital Research 2012 I presented a position paper with colleagues regarding the role of the Research Software Engineer. This paper followed on from a discussion I led at the Collaborations Workshop and some very interesting blog posts by Zirk Grensen and Ilan Tochorov. Rather than repeat these discussions, I’ve written this post for those who think the Research Software Engineer role could be for them.

A quick note about the Research Software Development Team at UCL

With the establishment of the Research Software Development Team at UCL, I hope we’re on the way towards establishing a successful home for scientific programmers. If you love learning about cutting edge research, and enjoy crafting robust, readable and efficient code, then please do join the UCL team.

Bringing together the best of two archetypes

A good scientific coder combines two characters: the scholar and the craftsperson.
• Software development
  ▪ Software development: general best practice
  ▪ Ready for release?
  ▪ Defending your code against dependency problems
  ▪ Developing maintainable software
  ▪ Writing readable source code
  ▪ Testing your software
  ▪ How to frustrate your users, annoy other developers and please lawyers
  ▪ How do I figure out if this software is “good”?

• Project Management
  ▪ Choosing the right open-source software for your project
  ▪ Choosing project and product names

• Repositories
  ▪ Choosing a repository for your software project
  ▪ Migrating project resources: what to remember
  ▪ Creating and managing SourceForge projects
  ▪ Retrieving project resources from NeSCForge

• Open source
  ▪ Adopting an open-source licence
  ▪ Supporting open-source software

• Community building
  ▪ Recruiting champions for your project
  ▪ Recruiting student developers

• Publicising software
  ▪ Creating videos for software projects
Digital preservation and curation
The danger of overlooking software

From preserving results from research, to storing photos for the benefit of future generations, the importance of preserving data is gaining widespread acceptance. But while digital software is easy to track on the preservation of data and other digital objects, like images and music samples, because they are typically the end products of the work that people wish to preserve. The software used to access the digital information is a hidden middleman in the preservation process. Without that software, it could be impossible to access the preserved data, which undermines the reason for storing the data in the first place.

In this case, the software falls under the Research Council’s preservation policy. This means that the software must be preserved as a condition of funding.

The software is a valuable historical resource.

In this example, the software was found. It was historically significant, even though the software has inherent value.

What are the issues?

Software presents some challenges to those who preserve, preserve and archive in particular software preservation is difficult, because software is sensitive to changes in its environment. If there is a change to the computer or operating system on which the software runs, the software will often stop working properly. This change might not cause a complete software failure, but it might cause a subtle, yet important, change in results. Expert knowledge is needed to fully understand how particular software component works and the effect a change may have.

Software comes in many different forms, is written in a bewildering range of languages and can be licensed in a number of different ways. Additional difficulties can arise from the increasing use of web services and the cloud, where the software is not stored. If the software is not available, it could be impossible to access the preserved data, which undermines the reason for storing the data in the first place.

Congratulations!
You’ve inherited some code

This briefing paper is targeted at software developers and project managers dealing with the issues of using open-source software in their ongoing development, where no proper handover occurred. It highlights some of the key pitfalls in working with inherited and “dusty” code and provides advice on the steps you need to take to ensure that code is handed over properly, see the related briefing paper “Help! Your developer is running away!”

It is important to deal with this as early as you can so that you are prepared, especially if you are required to maintain a continued service.

Why is this important?

In many cases, issues arise when a piece of software is still being used by after the last developer who has any understanding of the code has left the community. This is because of the lack of documentation, changes in direction or simply because the software has been in use for a long time. The bug is normally a change in the operating environment (e.g. the operating system changed) or a request for new functionality. Taking old code is often necessary and important, because users continue to rely on the software for their work and it is impractical to change the software itself.

How to deal with it?

The most important thing is to ensure that the software is documented properly, so that someone else can understand it. This is especially important if the code is being used to implement a critical system. If the code is not properly documented, it may be difficult to understand how it works, and it may be impossible to find the bugs. It is also important to ensure that the code is properly maintained, so that it can be updated as needed. This involves regular testing and updating, as well as keeping up with changes in the operating system and other dependencies.

Thank you for reading this briefing paper. If you have any questions, please don’t hesitate to contact us.

Software Sustainability Institute
SSI Blogs

- **Main SSI Blog**
  - Articles on software sustainability and related issues
  - Contributions from SSI Staff, SSI Agents, and Guest Contributors
  - Aiming for at least one post per week, increasing as more contributors come on board
- **Ask Steve**
  - “Your software development questions answered”
  - More technical – focused at developers
  - Aiming for one question per month
    - Plenty suggested by community!
• Software Carpentry
  ▪ [http://software-carpentry.org](http://software-carpentry.org)
  ▪ International initiative to teach basics of software engineering to computational researchers
    ▪ The “why” more than the “how”
  ▪ Phenomenally successful – 2x oversubscription
  ▪ Cheap to run but budget for 3x the coffee
  ▪ 19 workshops already confirmed in Q1/Q2 2013, another 24 requested

• Software Sustainability Surgeries
  ▪ “Bring your own code”
  ▪ “Code Smells”
  ▪ Run at existing conferences, and for software funding programmes
  ▪ Offering bespoke advice as well as training
SSI Project Evaluation Criteria

- **Importance**: the alignment of the research domain to the UK’s strategic research roadmap.
- **Enthusiasm**: the impact which the work will have on the community, engagement of software authors with process and the likely additional contribution that would be gained from the community.
- **Value**: the impact on the research outputs. Would the science enabled be significantly improved by the work? This is a measure of the User Demand for improvement.
- **Availability**: the likelihood that the work would enable the software to reach a new stage of availability e.g. taking it from within one collaboration to make it fit for the whole research community or a new community.
- **Tractability**: the impact on the software. Will it be possible to improve easily the quality or performance of the software?
- **Opportunity**: will the work lead to new opportunities for sustainability, e.g. collaboration with other groups, commercialisation, alternative funding or new effort?
Case Study: Ligand Binding

• Centre for Computational Chemistry, Bristol
  ▪ New methods for rapid MC sampling of biomolecular systems modelled using QM/MM
  ▪ Developed two codes ProtoMS (F77) + Sire (C++)
  ▪ Water-Swap Reaction Coordinate method to calculate absolute protein-ligand binding free energies

• SSI’s work is helping to scale development
  ▪ ProtoMS and Sire both single developer codes
  ▪ ASPIRE/ACQUIRE framework has multiple devs
    • Split architecture between ASPIRE (adaptive multiresolution hybrid MD simulation) and ACQUIRE (WorkPacket scheduling system with optimisation for time to result vs “green-ness”)

• http://www.siremol.org/adap2ve_dynamics
Case Study: Fusion Plasma

- Culham Centre for Fusion Energy
  - GS-2 used to study low-frequency turbulence in magnetized plasma
  - No common visualisation across different groups
  - Deliver mutually agreeable framework that can be extended easily and can be maintained by the small fusion community

- SSI’s work means the software can be used between groups
  - Simplified & enhanced plasma visualisation tool
    - Based on ParaView o/s tool
    - For simulations using GS-2 o/s package
  - Aim to allow CCFE to contribute back to GS-2 community
  - “I am very confident the tool will be invaluable”
    - Colin Roach, CCFE

- [http://www.software.ac.uk/who-do-we-work/culham-centre-fusion-energy](http://www.software.ac.uk/who-do-we-work/culham-centre-fusion-energy)
- [http://www.ccfe.ac.uk/](http://www.ccfe.ac.uk/)

Software Sustainability Institute
Case Study: Brain Imaging

- Brain Research Imaging Centre, Edinburgh
  - Develop PrivacyGuard software, a DICOM image deidentification toolkit
  - Created software to support new multispectral colouring modulation and variance identification technique (“MCMxxxVI”) to identify white matter lesions that are indicative of declining cognitive ability
  - BRIC are not principally software developers, but do provide software to other researchers
- SSI’s work means the software has been reviewed and refactored
  - looked at exploitation
    - Usability review, Naming/trademark review
  - Made it easier for BRIC staff to maintain and develop
    - Move to standard repositories, testing and documentation processes
    - Examination of licencing for MCMxxxVI
    - Extraction and refactoring to create standalone tools

- [http://www.software.ac.uk/who-do-we-work/brain-research-imaging-centre-edinburgh](http://www.software.ac.uk/who-do-we-work/brain-research-imaging-centre-edinburgh)
- [http://www.bric.ed.ac.uk/](http://www.bric.ed.ac.uk/)

Software Sustainability Institute
Case Study: SARoNGS

- UK NGS
  - Previous SARoNGS project aimed to provide a simple mechanism to obtain the security tokens required to access the National Grid Service.
  - This service uses the UK Access Management Federation to identify people using Shibboleth.
  - First in ongoing series of “NGS recommends...”

- SSI’s work means the software has been reviewed and issues corrected
  - Identified, fixed deep bugs in SARoNGS WMS
    - It is now possible to run SARoNGS jobs successfully on NGS systems
    - Though still needs NGS to roll out more widely...
  - “Happy with the report, using it to beat people now”
    - David Wallom, OERC & NGS

- https://cts.ngs.ac.uk/
Case Study: Climate Policy Modelling

- CIAS team at Tyndall Centre for Climate Change Research, University of East Anglia
  - Develop linked climate and economic models for detailed analysis
  - Their software was not ready to be used by other groups
    - One researcher/developer at UEA, several users
- SSI’s work means the software is robust enough that it can be installed and used by others
  - Enabled use of the software by the WWFN’s Climascope project and James Cook University
    - Documented software to allow extensions by contributors
    - Made it easier to maintain and backup
    - Added job scheduling to improve modeling throughput
    - New modelling framework enables new models i.e. new science
- http://www.tyndall.ac.uk/research/cias
Case Study: textual studies

- TextVRE team at CeRCH, Kings College London
  - Developed an environment which is used to integrate various tools used in the e-Humanities textual studies lifecycle
  - Builds on the German TextGrid project, and many other existing tools
- SSI’s work means the software is can be run “out of the box” – an important requirement for the researchers
  - Developed a VM image containing the TextVRE installation
    - Improve installation instructions
    - Develop tests to check each installed component
    - Improve modularisation to allow others to contribute and maintain
  - Feeding back work to TextGrid
- [http://textvre.cerch.kcl.ac.uk](http://textvre.cerch.kcl.ac.uk)
Case Study: NeISS

- Evaluate impact of traffic control measures over next 5/10/15 years
- Access baseline demographic data about the city
- Execute simulation of traffic system and population
- Visualise simulation outputs
- Augment with new forms of data
- Run dynamic models to assess future patterns (congestion, health, social inequality)
Case Study: NeISS
Case Study: NeISS

Meet neISS and the Tweet-o-Meter

In the late seventies, according to the pop group M, ‘New York, London, Paris, Munich’.
Software sustainability is:
- community engagement
- data management
- promotion
- lowering barriers
- improving capability
What do we sustain:
- Function?
- Form?
- Knowledge?
Collected Wisdom

- 5 Myths
- 10 Simple Rules
- 5 Development Stages
- 2 Barriers
- 3 Questions
5 myths of software sustainability

1) Collecting software in repositories sustains it *(the myth of preservation)*
2) Scientific software can be handed off to experts *(the myth of robustness)*
3) One heroic programmer can keep it going *(the myth of personality)*
4) Software in one domain is the same as in any other *(the myth of generalness)*
5) Open sourcing gives me free effort *(the myth of puppies, or build it and they won’t come)*

http://www.slideshare.net/npch/software-sustainability-looking-past-the-myths
10 Simple Rules
for the open development of scientific software

1) Don’t reinvent the wheel
2) Code well
3) Be your own user
4) Be transparent
5) Be simple
6) Don’t be a perfectionist
7) Nurture and grow your community
8) Promote your project
9) Find sponsors
10) Science counts

10 Best Practices for scientific computing

1) Write programmes for people, not computers
2) Automate repetitive tasks
3) Use the computer to record history
4) Make incremental changes
5) Use version control
6) Don’t repeat yourself (or others)
7) Plan for mistakes
8) Optimise software only after it works correctly
9) Document the design and purpose of the code, rather than its mechanics
10) Conduct code reviews

Submitted to PNAS. http://arxiv.org/abs/1210.0530
5 development stages

1) **Idea** (Seed) *(one person)*
2) **Prototype** (Start-Up): *1-2 people contributing, few people using, normally within same organisation, informal support*
3) **Research** (SME): *few people contributing, more people using, a few different organisations mostly known to creators, limited support*
4) **Supported** (Expanding company): *several people contributing, many people using, several organisations, formal support and contribution mechs.*
5) **Product** (Established company): *many contributors, users, organisations, support mechanisms.*
Software development comes in stages

- Bridging criteria: strength of team; strength of market; proximity of software to market

1. Idea
2. Prototype
3. Research
4. Supported
5. Product

- An idea to solve a problem
- Understand the functionality
- Scaling to work for others
- Allow others to participate
Software is not developed in isolation in each domain

Not all software *should* make it to the next stage
Management changes through time, requiring planning
2 Barriers

Collaboration Coordination

Remove the barriers to each and you’ll go far
5 Stars of Software

• Do we need a 5 stars for software?
  - Existence – there is accurate metadata that defines the software
  - Availability – you can access and run the software
  - Openness – the software has an open permissible license
  - Linked – the related data, dependencies and papers are indicated
  - Assured – the software provides ways of assuring its “correctness”

C.f.
5 Stars of Linked Data
(Berners-Lee)
5 Stars of Online Journals
(Shotton)
Question 1
How much time should you spend on:
- collaborating with scientists?
- new features?
- publicity / promotion?
- code improvement?
- training + usability?
Question 2

How do you identify your next:
- users?
- developers?
- champions?
- funders?
- demo?
Question 3
How do you find out about your community?
- what software is used and how?
- major barriers to engagement?
- size of userbase and skillset?
- impact of effort invested?
3 Golden Rules

- Act Amiably
- Breakdown Barriers
- Cultivate Contributors
A: Act Amiably

- Bugzilla project did analysis of their current and previous contributors
  - Once someone started contributing, what kept them around?
    - Don’t freeze the trunk for long periods
    - Turnover is inevitable
    - Respond to contributions immediately
    - Be extremely kind and visibly appreciative
    - Encourage a total absence of negativity

- Software sustainability relies on many people working happily together!


Software Sustainability Institute
B: Breakdown Barriers

- Basics: Website, mailing list, code repository, issue resolution
- Remove barriers to participation, increase efficiency

- 1993: First public release; 2 devs
- 1995: Code open sourced; 3 devs
- 1996: r-testers list set up
- 1997: lists split: r-announce, r-help, r-devel; public CVS; 11 devs
- 2000: CRAN split and mirror
- 2001: BioConductor
- 2003: Namespaces
- 2005: l8n, L8n
- 2007: R-Forge
- Today: BioConductor (33 core devs), R-Forge (532 projects, 1562 devs), CRAN (1400+ packages)

Software Sustainability Institute
C: Cultivate Contributors

Building intellectual access ramps to support incremental engagement – building capacity and capability

- Individual
- Group
- Consortium
- W/ industry
- Community
- Active

Teams change as project matures

Software Sustainability Institute  
http://www.castep.org/
D: Diaspora + Dispersion

- Sometimes, serendipity plays a part
- E.g. GEOS-CHEM
- But they also have great support and contributor mechanisms

http://acmg.seas.harvard.edu/geos/

Software Sustainability Institute
The modern researcher...

• ... worries about:
  ▪ Data management and analysis
  ▪ Reproducible research
  ▪ Scalable simulations
  ▪ Integration of models and workflows
  ▪ Collaboration

Where do they learn how to do this?

Picture of Otto Stern courtesy of Emilio Segre Visual Archives

Software Sustainability Institute
Software philosophy as part of the process

- Foundations of scientific computing in undergraduate courses
  - Like presentation skills
- Methods of scientific computing in postgraduate courses
  - Like statistics and ethics
- Show the benefits from the knowledge and methods of digital research
  - Not just programming 101