"This is where my time goes!"

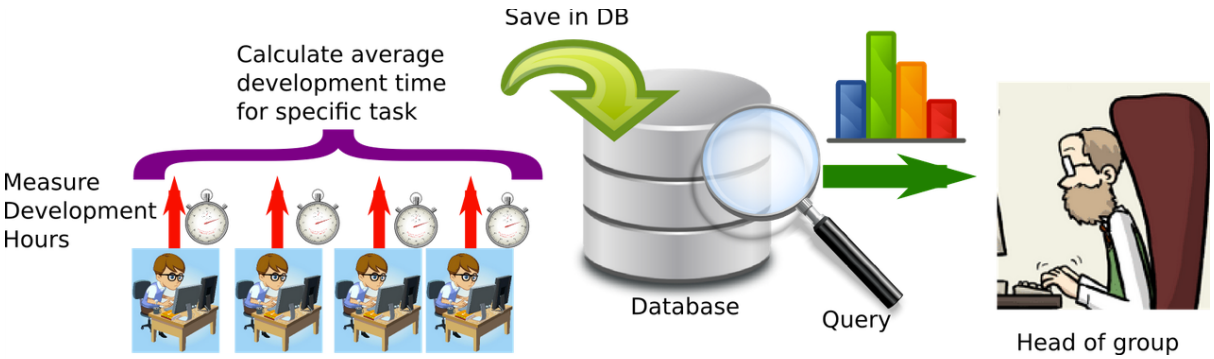
Collaborative Idea team members
Robyn Grant, Fabian Renn, Ian Gent, Mike Jackson

Context
Any involving software development.
Problem
It can be hard for non-coders e.g. PIs, to appreciate how challenging, and time-consuming, development tasks are.
It can be hard for coders to estimate how long they need for development tasks, when encountering new tasks or technologies. This especially applies to new coders. But they'll often be asked to estimate regardless by PIs/managers.
Solution
A web-based app that allows people to record time spent on development tasks e.g:

- Install a product
- Set up web server
- Set up build system
- Design, develop, test component
- Fix bug
- Write user guide
- Answer support query
- etc.

Users create an account and record this information and their perceived expertise level.
App can process data across many users anonymously to see average time to do certain tasks.
Help your own personal estimation, help with justifying time estimates to PIs/managers, help PIs/managers when seeking funding.
Policy drivers e.g. SSI, can use the data to drive for recognition of the contribution of and effort invested by coders in research software.

Diagrams



Jenkins Job Joggler

Collaborative Idea team members
Jens Nielsen, Olexandr Konovalov, Robert Haines, Robert Davey
Hackday pitch leader
Jens H Nielsen
Context
Continuous integration is the automatic testing of your code whenever someone pushes a new version to your repository. Combined with a large test suite this ensures that bugs are caught early. Automatic tests of pull requests with feedback to the pull request are also possible.
Problem
Continuous integration systems such as Travis allows the definition of testing jobs in a simple text file defined within the code repository. Thus the definition of the tests is version controlled too. Jenkins on the other hand uses a web based GUI along with a xml format to define jobs. This makes it easy to create jobs but difficult to versions and make fundamental changes such as altering the job type. Jenkins Job builder by OpenStack allows the definition of Jenkins jobs in simple YAML files. However, no support for deploying these scripts exists.
Solution
Implement a Jenkins plugin that uses the GitHub api to travel the repositories within an organisation and looks for jenkins.yaml files defining the jobs. These jobs will be subsequently be rebuild and uploaded to the Jenkins test server using the Jenkins Job builder.

Diagrams

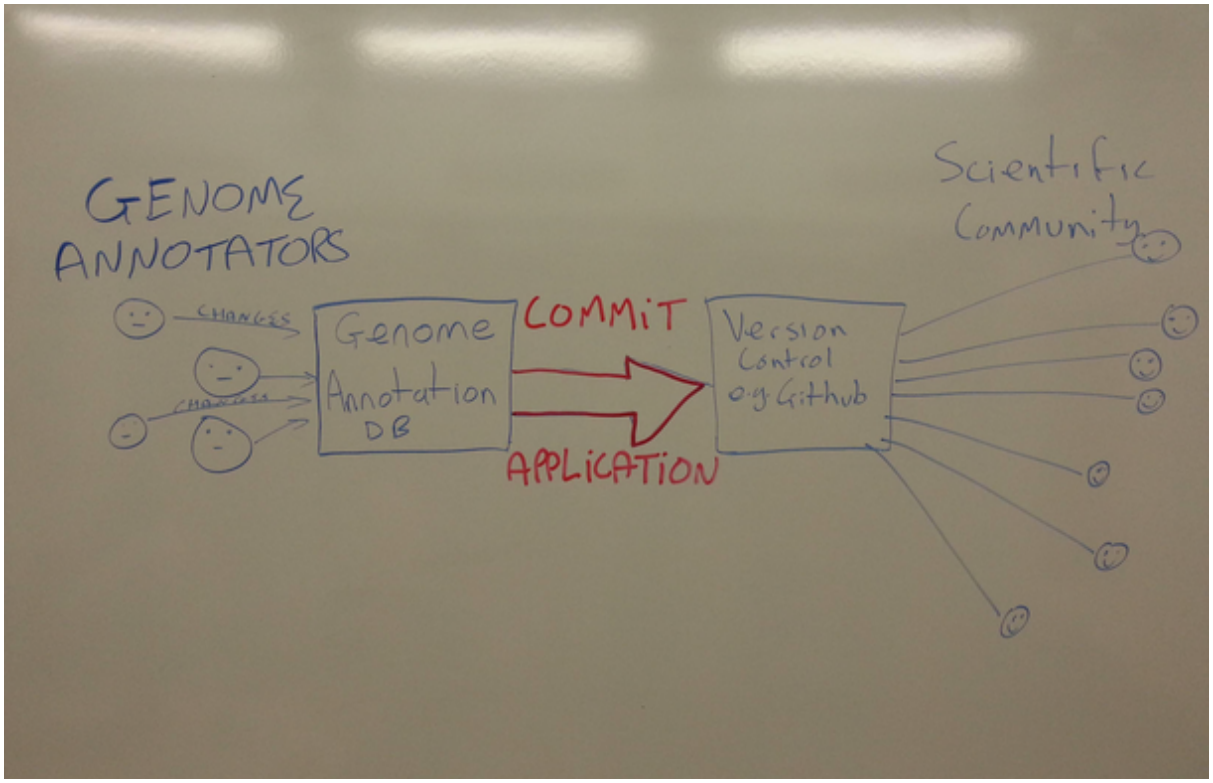
Domain-specific Git without knowing it

Collaborative Idea team members
Robin Wilson, Graham Etherington, Clyde Fare, Laurent Gatto, Arfon Smith
Hackday pitch leader
Graham Etherington
Context
Many domains - create specific tools for separate domains

Problem
Version control is really useful for all sorts of applications within science, but it's really hard - both conceptually and in terms of practical usage. Git is particularly hard to understand, and using Github or one of the many Git GUIs still requires understanding the conceptual model underlying Git. Because of this complexity, people don't use version control - even in situations when it would really help them.

Solution
Creating domain-specific tools which use Git for versioning, but hides this complexity from the user. This would reduce the barriers to entry, and allow users to access things from a simple interface - but also allow advanced users to access the repository using standard git interfaces, and to share the code using Github.
A good example of this would be an application which allows users to annotate genome sequences. The application would have a GUI to display the sequences and allow users to annotate and edit them. All of the data would be stored in text files within a Git repository on Github which would automatically deal with versioning (commit updated annotations, push them to Github, display a nice GUI to deal with merge conflicts etc). Thus, users with no understanding of the underlying concepts of version control can still edit and contribute to a fully-versioned repository of genome annotations.
This application could be a standard genome editor tool which interfaces with a 'commit application' to deal with versioning, or the versioning could be built-in to the application (for example, as a plugin), or the committing could even be done on a time-basis (eg. every hour, or when the application is closed).
Other examples include text editors (either local software or webapps) which automatically produce commits when you save - plus a nice GUI for scrolling back through previous versions (almost like TimeMachine on a Mac) - along a similar idea to www.substance.io.

Diagrams



Increasing citation of software in publications

Collaborative Idea team members
Tim Parkinson, Filippo Mortari, Leanne Wake
Context
Scientists are worried about losing intellectual property and have to rely on collaborators or end users. There is not a universal method for citing software and citing software in scientific literature is in its infancy.
Problem
How can we make sure that software used in research is consistently cited?
Solution
1. Introduce a module in Endnote which searches your submission for keywords such as "model", "software", "algorithm" and suggests links to citations.

or (Better Idea from Fillippo)

1. Introduce a novel formal language called Software Description Language "SDL". SDL is XML-based metadata capable of describing the intellectual provenance of the software used.
2. Provide XML based file contained in the package along with, e.g. license.txt, source.txt, SOURCE/ called 'CITEME.SDL'
3. 'CITEME.SDL' will contain the following (example):
e.g. for a journal paper with BIBTEX
@article{weisberg1998second,
 title={A second genetic polymorphism in methylenetetrahydrofolate reductase (MTHFR) associated with decreased enzyme activity},
 author={Weisberg, Ilan and Tran, Pamela and Christensen, Benedicte and Sibani, Sahar and Rozen, Rima},
 journal={Molecular genetics and metabolism},
 volume={64},
 number={3},
 pages={169--172},
 year={1998},
 publisher={Elsevier}
}

EXAMPLE: SDL Language
<software>
<title>
…..
</title>
<author>
…….
</author>
<version>
……
</version>
……
……
</software>"

HackerNews for Open Science

Collaborative Idea team members
Jane Charlesworth, Alexandra Simperler, Ross Mounce and Devasena Inupakutika
Hackday pitch leader
Jane Charlesworth
Context
Building a friendly, reproducible research community that listens and helps each other out
Problem
Lots of useful Scientific information is currently scattered across various information domains and open data portals (websites, journals, Blogs, Twitter, Reddit and other social business sites etc.).
Currently content on various sites is not moderated and comes across as patronising/ excluding which might prove to be offensive to diverse communities (software/ non-software field, scientific researchers etc.).
Keeping oneself updated on latest news (e.g. Difficulty that currently persists in catching up on news after returning from vacation.)
Solution
Solution 1: Centralising all the scattered information on one site and making it a global open sustainable portal easily accessible across research community as a gateway to all this distributed data.
Solution 2: Community self-moderation following stack overflow way of commenting, up-voting or down-voting sensible or offensive information respectively so as to captivate, keep research community engaged and active.
Solution 3: Configured daily, weekly and monthly digest which involves notifying users of the missed updates from fellow users/ communities during their sabbatical.

Diagram

Finding ways to increase recognition for developers of scientific software

Collaborative Idea team members
Jonathan Cooper, Bruno Vieira, James Spencer, Sebastian Gibb, Jure Triglav
Hackday pitch leader
Jure Triglav
Context
Research impact, specifically impact of scientific software.
Problem
We think that there is not enough reward or recognition for tool builders.
Solution
Build upon the ideas of ImpactStory and ScienceToolbox and build a better profile for scientific software developers, where their software and its citations are gathered in one place.

Codiverse

Collaborative Idea team members
Philp Fowler, Derek Groen, Jennifer Molloy, Adam McMaster, Martin Hammitzsch
Hackday pitch leader
Derek Groen
Context
Software engineering: Getting feedback on your code
Problem
How do you know how well your code is written?
How do you learn new code patterns?
Isn't asking your colleagues a bit, well, old-fashioned?
Solution
Crowd source the rating of your code! Think Zooniverse but with code snippets.
Develop an app that:

- selects small (10-30 lines) of code randomly from GitHub repositories based on your preferences (e.g. python/Java or even just README.md files)
- displays them on the screen
- rate them based on "code elegance" or in a more structured way (documentation / readability / etc)
- could be web only, ideally mobile devices too
- maybe even add it into a Pomodoro approach: "take a break, rate code and learn"

Diagrams

Spreadsheets as a data service

Collaborative Idea team members
Nicolas Gruel, Michael Fischer, Graham Klyne, Kewei Duan, Dominic Orchard

Hackday pitch leader

Graham Klyne
Context
Lots of researchers use spreadsheets to store and share data, and explore data relationships.
Problem
Spreadsheets are very poor for sharing data and mixing with other data sources, and generally are not sustainable. They are also difficult for writing programs against. Yet, we do not see them going away anytime soon.
Solution
We propose a system for turning spreadsheets into flexible data sources for sharing, mixing with other sources, and programming against. This system has two phases:
Phase 1: create a web service that will provide a simple web API to access data in the spreadsheet. There is a Google-defined API for accessing spreadsheets: one might try to replicate a subset of this interface. This would allow programmers in a range of languages to create tools to access and present the spreadsheet content in more accessible/sustainable

formats, as well as use spreadsheets as data sources for models. Python already has some Excel support which is related to this goal [2].
Phase 2: create a tool using the API to re-present the data in some more sustainable self-describing format (e.g. XML, RDF, etc.).
Note: to keep things simple, it may be best to concentrate initially on just the data and not the formulae or graphics. Also, initial implementation might concentrate on single-worksheet workbooks.
The service may be run on a localhost or remotely, providing URIs to the data, for example (phase 1 API):
http://localhost:8000/spreadsheetname?gid=n (nth worksheet, per Google API)
http://spreadsheet.example.org/spreadsheetname/3/2 (e.g. 2nd row of third worksheet)
etc.
For phase 2, we might invoke content negotiation to access a different format, e.g.
GET/spreadsheetname/3/2 HTTP/1.1
host: spreadsheet.example.org
accept: application/json

might return
{ “col1”: val1, “col2”: val2 }

Details to be finalized.

Specific ideas/challenges:
May require a configuration language to describe properties of the spreadsheet.
Equations are common, these may be set as ‘read-only’ fields in the data source sustainable format? Versioning should be include so new version of the format will not break the system or transform the data in an agnostic format (text)?
Libraries may be read-only access, not allowing updates to the spreadsheet.

Reproducibility of chemistry and biological experiments

through controlled data and metadata sharing

Collaborative Idea team members
Liberty Foreman, Michael O'Hagan, Katalin Phimister, Ling Ge, Stephen Crouch
Hackday pitch leader
Liberty Foreman
Context
Chemistry/biogical platform for data sharing, and interlinking of data & metadata multi-institutionally.
Oxford currently rolling out the E-Lab Notebook to close stakeholders (first chemistry), then expand to others - what should be considered?

Problem
How do we increase the reproducibility of chemistry and biological experiments, and enable

secure sharing of data and metadata for experiments to other stakeholders?
Key problems:
- Existing lab notebook solutions often make it difficult to organise all your data into a coherent form for reproducibility. Those researchers generate a lot of data (e.g. in spectroscopy), write a bit of code, build it, encrypt it, it becomes a black box. You typically don't have all the information to capture all the configuration, parameter data to fully replicate the experiments.
- An intrinsic issue is that there is often significant value in the data and metadata, so how to support the sharing of data to protect this where necessary i.e. against competitors?
- Another problem is to be able to support the in-lab messy experimental process: researchers use written notebooks.
Three big issues that need to be solved:
- The culture needs to change to share not only the data, but where this is already done, sharing the key metadata that is also crucial to the experiment.
- Providing the tools to support this process.
- What's in it for them? Need to motivate people to submit their data, which again is part of the culture problem.
Two perspectives to data: the actual experimental data and the 'dark' or negative data (e.g. metadata), and you need to capture both.
If there were a connection between the data captured on the lab notebook and the code, with version of software, parameters, data, etc. it would be much easier.

Solution
Long-term:
- Identify and engage with a high-profile group that are keen to be involved and champion the ideas and the technology.
- Develop ELN to become a promotion platform for pharma companies, to promote takeup and avoid duplication of results. For this, it's important to support degrees of access - to relax the security as the collaboration/business opportunities progress.
- Develop a set of benefits for why this is a good idea e.g.:
-- Avoiding duplication of experiments
-- For close collaborations, it captures the precise information for allowing others to replicate the experiments e.g. a close partner
-- A cohesive platform allows for the searching for collaborators in similar areas
- Develop formal templates for capturing all those things that need to captured for each field, but start with one field and then develop others.
Engage a single group (smaller, easier scope) to work out what a) needs to be shared and b) what they're willing to share:
- First step: metadata, image created through analytical software, notes for software. This makes it an initially an easier but still valuable process. The tool interface critically needs to be developed to support this process in an easy way.
- Next steps: add to this set of information as other aspects are determined to be useful to be captured.
Specifically for the Hackday: mock-up a dataset and template that could be used as a basis for developing a strawman Wagtail site on the day, to be taken forward, evaluated and refined after the Hackday

Diagrams

MASHBOARD

Collaborative Idea team members
Paul Barrett, Niall Beard, Marta Ribeiro
Hackday pitch leader
Marta Ribeiro
Context
For any scientific research
Problem
Multiple platforms for storing and representing information about research artifacts (e.g Code on Github, a document for proposals , FigShare for data). Makes it difficult for people to find everything, plus there is the username/password reminding issue.
Solution
MASHBOARD. Mash together online research objects such as papers, proposals, code data and anything else into one online dashboard. Use scrapers and APIs to pull in and aggregate information from an array of sites to be decided upon.
Gives researchers the ability to share it as well as make it publicly available.
Diagrams