Open Source Sustainability and RDM

Scott Wilson
scott.wilson@oucs.ox.ac.uk

JISC

What does sustainability mean?

To be sustainable a project must meet its own costs. Most projects have their initial costs covered by an injection of funding from a parent body or sponsor. However, what happens when this money runs out?

In some cases, the parent body will adopt the project as a production service; this is great news for future sustainability, but will not necessarily be sufficient to retain developers.

In some cases, it may make sense to spread the risks across more than one institution, as a shared service or as an open development project.

In the rest of this talk, I'll take a look at sustainability of third-party software you may use for RDM, and also cover common open-source business models that might apply to RDM software you may have developed.

JISC

How does sustainability apply to you?

- Sustaining software developed by another organisation or project, that you use and will be relying on in future
- Sustaining software that you have developed, and are considering sharing with others

http://www.oss-watch.ac.uk/resources/researchinfrastructure-sustainability.xml

JISC

Sustainability of third-party software

- Risk mitigation
- Evaluation
- Engagement

JISC

Evaluating sustainability

- Maturity of the software
- Viability of the community supporting the software
- Governance of the software and openness to contributions

JISC

Models

- **Openness** Evaluation (OSS Watch)
- **SSMM** - Software Sustainability Maturity Model (OSS Watch, Open Directive, CENATIC…)
- **RRL** - Reuse Readiness Level (NASA)
- **QSOS** - Quality and Selection of OSS
- **CMMI** - Capability Maturity Model Integration

JISC

Engagement

- Use only
- Contributing to the community
- Shaping the future direction

JISC

Engagement (2)

- Engagement options available depend upon the governance model used by the project

  e.g. meritocracy, benevolent dictator, single-company etc.

- This is another risk factor to consider:
  - You may have limited options for contributing to the future sustainability of the software
  - Can be evaluated using models mentioned earlier

JISC

Engagement (3)

- Types of engagement
  - Bug reports
  - Feature requests and requirements
  - Documentation
  - Translations
  - Marketing and publicity
  - Software development

JISC

Sustainability of software you have developed - and want to share as open source

- Business models
- Community

JISC

Business and Sustainability Models

- These are mostly not mutually exclusive, and will most often be used in combination as appropriate – more accurately they are elements of business models
- This is still an emerging area of business practice
- Some of the current success of FOSS software exploitation techniques may be attributable to dissatisfaction with more traditional proprietary techniques and their associated big-name vendors, rather than any innate superiority
- It remains to be seen whether the current global financial difficulties will help FOSS business or hinder it. Analysts are currently predicting both.

http://www.oss-watch.ac.uk/resources/businessandsustainability.xml

JISC

Overview of models

Academic community development
Spin out entity
Existing foundations (ASF, Eclipse…)
“Community Source” Foundation
Consultancy
Internal Cost Reduction
Paid Support, Documentation
Integration, Upgrades
Software as a Service
Competitor Disruption
Advertising and Referrals
Training
Trademarks and Merchandising
Dual licensing
Proprietary version (“bait and switch”)

JISC

First - what you cannot / should not do

- Charge for licences for specific uses of your code, for example commercial use (Open Source Definition point 6)
- Charge for licences in general (Possible but subject to low/zero-cost competition from all recipients)
- Tweak an existing FOSS licence for your purposes and still call your software Free Software or Open Source Software (Strong community rejection of these practices)
- Silently incorporate FOSS software in your proprietary offering without abiding by the licence conditions (detection is likely, and although legal damages are unlikely, damage to reputation is certain)

JISC

Academic Community Development

- FOSS licensing permits a varied group of contributors to work on software that addresses a particular problem domain.
- Institutions and their academics can gain public profile by contributing to such projects and becoming associated with respected tools in specific areas of research. It can also help ensure the continued existence of useful solutions.
- Examples include **BioImage Suite** (biological image analysis software) **YARP** (experimental robotics software) and **The Versioning Machine** (software for aligning differing versions of xml-encoded texts).
- Recognition for work on academic tools is still, however, some way behind more traditional forms of academic recognition for publication etc

JISC

Establishing a separate legal entity

- Adds to sustainability by isolating risks (IP infringement, event organisation, damages from failure) from the parent institution
- Facilitates donation of money and simplifies tax issues
- Most research institutions are already well-practised in setting up spin-out companies. In the case of sustaining FOSS projects some kind of not-for-profit entity may be just as or even more appropriate
- Such an entity can still have an affiliated commercial entity engaged in exploiting the software and the brands that it stewards

JISC

Moving into an external foundation

- The benefits of foundation status have led to the establishment of umbrella foundations holding multiple FOSS projects.
- Examples include the **Apache Software Foundation**, which supports Apache HTTP Server, Cocoon, Lucene, **Software in the Public Interest,** which supports the Debian Linux distribution and PostgreSQL, and the **Software Freedom Conservancy**, home to Samba, Busybox and Wine
- Entering an umbrella foundation can radically reduce running costs for projects that receive financial donations, as the foundation will handle the necessary book-keeping, as well as providing the risk management benefits that come with separating legal responsibility for a project from your host institution

JISC

'Community Source' Foundations

- Where a number of separate institutions see a benefit in jointly developing a piece of FOSS, they can adopt a model which has come to be known, somewhat confusingly, as 'Community Source'
- Each institution contributes resources to developing the code, the ownership of which rests in an external foundation
- In the initial phases the code may be unavailable outside the foundation, although it will eventually be released under a FOSS licence
- Contributing resources to the foundation buys institutions early code access and influence on the governance of the project and its functionality
- Mellon-funded projects **Sakai** and **Kuali** both began using this model

JISC

Consultancy

- Consultancy is another traditional technique for educational institutions looking to financially exploit their resources
- A more traditional model might be to sell licences to a piece of research-derived software and sell consultancy services and/or bespoke development services alongside it
- Potentially a FOSS release of the software can improve uptake, given its low cost of acquisition, and drive the market for associated consultancy and development services more successfully than the traditional model

JISC

Internal Cost Reduction

- Institutions may be happy to sustain an internally-developed FOSS project themselves if the project can demonstrate that it drives down the running costs of that institution or solves an institutional problem
- Projects that reduce costs in one institution may have good potential, when mature, to be deployed in others. This provides opportunities for paid consultancy and/or provision of the software as a service (see below)

JISC

Provision of Paid Support / Documentation

- Just because your code is freely available, it does not mean that the documentation or your help needs to be (as with the consultancy and bespoke development model)
- Support can be provided in time- or incident-limited bundles
- Support can be in the form of guaranteed performance on specific hardware
- Documentation can take the form of paid access to a knowledge base of previously resolved issues
- HOWEVER, in this case one is in competition with the software's user base/community, who may be willing to provide peer support for free

JISC

Integration / Managed Upgrades

- Managing the integration of various FOSS technologies, with their varying dependencies and release cycles, is a service that people are prepared to pay for
- Similarly managing the deployment of upgrade patches can be a paid service
- Bundles of tested, integrated FOSS software can be sold along with, potentially, support agreements
- HOWEVER, close integration may trigger responsibilities in particularly copyleft licences that could prevent integrated distribution – read the licences
- Example: **MapR** provide integrated Big Data solutions based on Apache Hadoop and related open source components

JISC

Competitor Disruption

- Sometimes a FOSS alternative to a competitor's product can disrupt their business model and provide competitive advantage (although this is almost never the sole motivation behind the release or distribution)
- Examples (arguably) include **Sun**'s OpenOffice.org, **Google**'s bundling and distribution of Microsoft-competing software such as OpenOffice.org, Firefox and Chrome (the 'Google Pack'), **Netscape Corporation**'s FOSS release of Netscape Navigator

JISC

Software as a Service

- Increasingly consumers are becoming comfortable with so-called 'cloud'-based software offerings – software that is accessed and used over the internet, and which stores data remotely from the user
- SaaS can be a useful solution to the problem of institutionally developed software that relies integrally on copyleft-licensed code
- Provision of service using copyleft software does not count as distribution, and thus does not trigger copyleft's reciprocal licensing responsibilities
- HOWEVER – this is a known 'bug' in copyleft licensing, and licences such as the GNU Affero GPL v3 are already in existence to 'fix' it.
- Another model is to use permissively-licensed components as the basis of the service, with, optionally, proprietary “glue” code and user interface

JISC

Advertising / Referral

- Your software or accompanying web site may be able to direct network traffic to an entity that is willing to pay for hits (although of course this functionality can always be engineered out by technically apt users)
- This is **Mozilla Foundation**'s main source of income
- **Firefox**'s built-in search box directs queries to Google
- The vast majority of **Mozilla Foundation**'s revenue ($132m in 2010) comes from **Google** under this deal.
- **Wordpress**, the FOSS blogging software and hosting platform is partly funding their parent company Automattic through this model

JISC

Training and Accreditation

- As well as support and consultancy, generalised training documents, courses and qualifications may be viable products
- Control of an associated trademark enables the provision of *'X-Certified Professional'* style programmes
- Actual training and examination are readily out-sourced

JISC

Trademark Licensing / Merchandising

- Just because your code is available under a FOSS licence, you do not have to permit universal use of your project's name and associated symbols
- Unlike copyright, trademarks are a registered form of IP, meaning that you have to apply to relevant government agencies for ownership. However, compared to patent application, trademark registration is relatively inexpensive
- Owning your trademark facilitates the sale of associated merchandise and accreditation and marks like “Powered by X” and “Using X technology”
- Can be a deterrent to forking if the brand is strong enough – the motivation to increase personal reputation by providing functionality outside project *“X”* is partially undermined by the inability to call the new project *“Improved X”*

JISC

Proprietary Versions and Components

- Sometimes referred to disparagingly as the 'Bait and Switch' model
- A FOSS edition of software is offered which lacks some of the functionality of a paid edition, either throughout its code or in the form of missing proprietary components
- While the existence of better-supported or hardware-accredited forms of FOSS offerings is generally accepted by the FOSS community, proprietary components and versions are less well-liked (although there is perhaps growing acceptance as the community matures)
- HOWEVER, this is another example of competing with the community. The FOSS model means that anyone can produce freely available versions of your paid functionality, given enough time and expertise

JISC

Dual Licensing

- Provided that you have the necessary ownership or sub-licensing rights over your project's code, you can provide it under differing licences
- In the classic case, these would be a copyleft licence and a paid proprietary licence
- Customers who wish to build software product incorporating your code and who do not wish to use the copyleft licence must pay for the proprietary licence
- This is therefore most suitable for code which is readily susceptible to inclusion within commercial software products, for example database backends

JISC

Community

Successful open source software projects tend to develop committed communities of developers and users. These communities can include everyone from hobbyists to professional software writers to end users. They may form naturally or they may be significantly assisted by a large corporation

http://www.oss-watch.ac.uk/resources/researchinfrastructure-community.xml

http://www.oss-watch.ac.uk/resources/buildingcommunities.xml

JISC

Resources

http://oss-watch.ac.uk

JISC

Do get in touch:

info@oss-watch.ac.uk
http://www.oss-watch.ac.uk
@osswatch

JISC