Mar 29th, 10:15 AM - 11:00 AM

Open Future: Assessing and Adopting Open Source Technologies for Your Library

Stewart C. Baker
Western Oregon University, bakersc@wou.edu

OPEN FUTURE

Assessing and Adopting Open Source Technologies for Your Library

Presented by Stewart Baker
Systems/IR Librarian
Western Oregon University
at Online Northwest 2019

OPEN SOURCE

The Basics

What is Open Source?

- Short Version: Open Source software (OSS) is "licensed to guarantee free access to … source code" (Bretthauer, 1)

- Similar to Open Access, Open Knowledge, and some flavours of Creative Commons licensing like CC-BY.

- Formal definition maintained by the Open Source Initiative at https://opensource.org/osd-annotated
  - Requires free redistribution, distribution of source code and a license, permission to create derivative works, non-discriminatory practices, and a number of other things.

Acronym Soup

- Free Software – Original term, dating from the 1950s
- OS – Open Source
- OSS – Open Source Software
- FOSS or F/OSS – Free and Open Source Software
- Libre – Used to avoid connotations of monetary cost
- FLOSS – Free/Libre and Open Source Software

Although all these movements have subtle philosophical differences, they all essentially have the same aims: Make software development and distribution freely accessible.

Why Use Open Source Software (OSS)?

- Faster, more responsive development.
- Easier access to developers when things go wrong.
- No (or low) monetary cost.
- Ability to develop the software further to meet own needs.
- (Mature) OSS is just as reliable as proprietary equivalents, if not more so.
- No licensing problems.

OSS Development

- OSS projects essentially use an "Agile" development model:
  - Interaction
  - Collaboration
  - Responsiveness to change
  - Working software

(Agile Alliance, 2001)

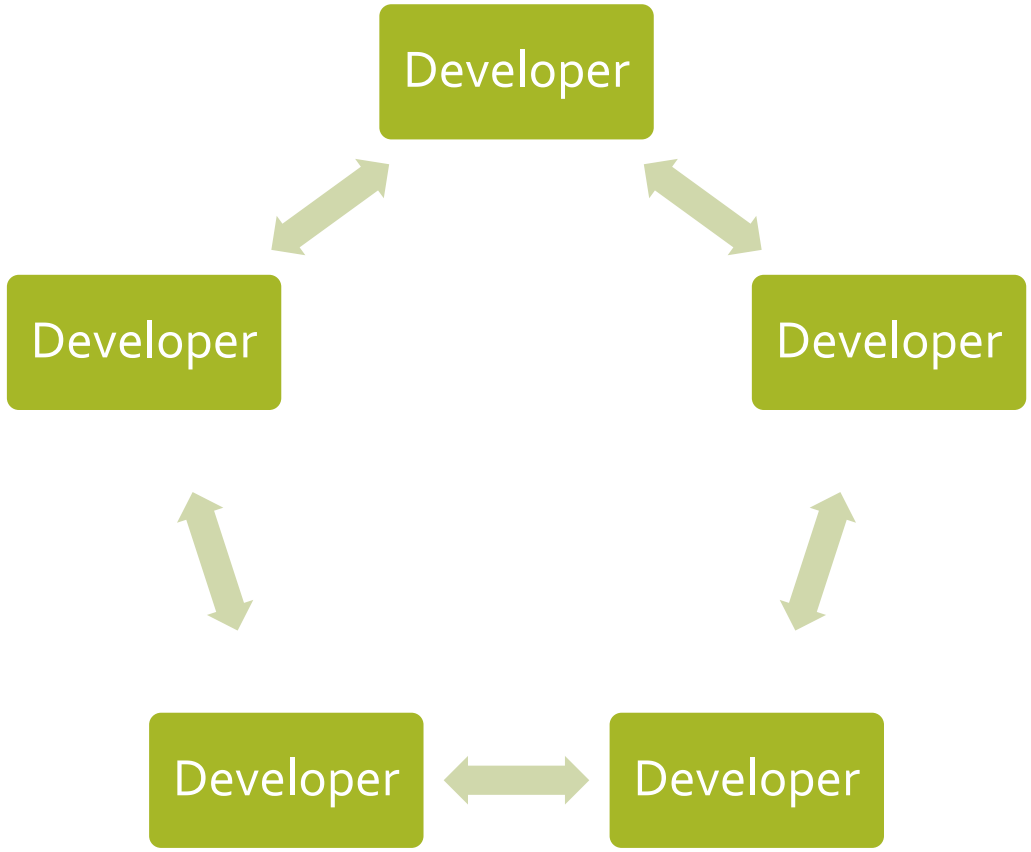
The Cathedral and the Bazaar

- Eric Raymond's *The Cathedral and the Bazaar* (2001) compares OSS development to traditional models of software development.

- **Cathedral model** – Typical commercial model: the focus is on a single developer who passes down the software to users.

- **Bazaar model** – Software is developed in a distributed, collaborative way that relies on a "large and active community of users and co-developers" (Raymond, 2001).

Collaborative and Decentralized



Community-driven

Volunteer

- Large projects are often supported financially by a Foundation, and have paid staff.
- The majority of OSS developers, however, are volunteers.
- Passion about the project's need or long-term success.
- Burnout or losing interest can be a problem for smaller projects.

OSS Benefit: Developer Responsiveness

- Much easier for users to affect path of OSS project development.
- User-contributed code can be merged into main project.

OSS Benefit: Cost

- "Free as in free speech, not as in free beer." (Free Software Foundation)
- Even so, OSS is often considerably cheaper than commercial alternatives.
- Upgrades and add-ons are usually user-created, and free to install.
- Some projects offer paid hosting or paid support as options.

OSS Benefit: Customizability

- OSS projects actively encourage users to modify code.
- Some OSS projects provide user-created modules.

Some Drawbacks of OSS

- Main drawback is the amount of expertise needed.
- Less likely to work straight out of the box.
- Morgan & Finnegan (2007) argue that technical and business drawbacks include compatibility issues, poor documentation, and limited support and training.
- Smaller OSS projects especially may run into problems with a lack of core maintainers to keep the project on track.

Finding OSS

- GitHub: https://github.com
- Sourceforge: https://sourceforge.net
- Free Software Directory: https://directory.fsf.org/
- AlternativeTo: https://alternativeto.net

ASSESSING OSS

Software Assessment: Fit and Maturity

Fit

Does the software do what you need it to do for your project to succeed?

Maturity

How robust is the community which creates and maintains the software?

OSS Assessment

Software itself (Fit)

- Functionality
- Fit for your project

Community behind the project (Maturity)

- Sustainability
- Frequency of updates
- Responsiveness to bug reports
- Number of developers
- Documentation, roadmaps, etc.

Checking for Project Fit

	COMS	***OpenConf Editions***	Microsoft Academic Conference
File uploads	Y	Y	Y
Bulk import schedule data	?	N	Y
Bulk file uploads	Some	?	?
bulk file downloads	Y	Y	?
Adaptable logic fields	?	?	?
Customizable fields	Y	Y (if local install)	Y
word limit on open text fields	?	?	?
Ability to associate a single presenter with multiple submissions or vice versa.	Y	Y	?
Ability to limit (or not) the number of submissions a presenter can be primary on.	Y	N	?
Submissions are editable until a set deadline.	Y	Y	Y

Checking for Project Maturity

- Stol and Babar (2010) found 20 different assessment frameworks/models/methodologies.
- Some can be hard to hunt down, especially if they are older or were produced by a corporation.
- Pay attention to what, exactly, the model is assessing, and who its intended users are.
- When choosing a model to use for assessment, consider how easy it is to use and what its purpose is.

Acronym Soup, Redux

- Capgemini OMM
- OSMM
- OpenBRR
- QSOS
- OITOS
- FOCSE
- OpenBQR
- SQO-OSS
- OpenSource Maturity Model (OMM)
- QualiPSO OMM

QualiPSO's OMM (Open Maturity Model)

- QualiPSO – EU project (2006-2010), aimed to improve quality of OSS production.
- OMM can be used by developers of new OSS projects **or** by end-users.
- Available online at http://qualipso.icmc.usp.br/OMM/
- If you can't find all the information you need, you can do a partial assessment.

OMM Basics

- Assess OSS project's maturity at one of three levels: Basic, Intermediate, Advanced.
- Each level has a number of Trustworthy Elements (TWEs), which check things like documentation, roadmaps, number of commits and bug reports, etc.
- TWEs are further broken down into "LookFors"—things to find as evidence of whether a project has implemented the TWE or not.
- Number of fully implemented TWEs provides a rating of project's maturity.

Example Assessment (Ambra, OMM)

PDOC: Project Documentation			
Purpose: Develop and maintain project documentation, making it readily accessible to the community.		**Implementation Priority**	**Rating**
Goal PDOC 1	Provide high quality documentation		
Practice PDOC-1.1	Create development documentation	Mandatory	3
LookFor	Check the availability of requirements specification		
	Check the availability of high level design / product architecture		
	Check the availability of detailed design		
	Check the availability of technical documentation (e.g. for use in debugging)		
	Check the availability of workflow guidelines (for checking, testing...)		
Practice PDOC-1.2	Create user documentation	Mandatory	3
LookFor	Check the availability of a user's guide		
	Check the availability of FAQ documents		

Example Assessment (Ambra, OMM)

DFCT: Number of commits and bug reports Purpose: Specifically analyze the activity related to source code commits and bug reports provided to the project		Implementation Priority	Rating
Goal DFCT 1	Provide a user-friendly environment for contributing bug reports		
Practice DFCT-1.1	Provide a standardized and well documented contributing mechanism	Mandatory	3
LookFor	Corrections to source code and new code can be contributed easily		
	Bug reports can be contributed easily		
	Corrections to documentation can be contributed easily		
	Suggestions for new features can be contributed easily		
	The defect reporting system is well aligned with the development environment		

Example Assessment (Ambra, OMM)

Trustworthy Element	Rating	Total number of practices	Fully Implemented Practices	Partially Implemented Practices	Not Implemented Practices	Practices not Rated
Overall Rating	**58.97%**	79	23	5	11	40
Licenses	2.87	9	7	1	0	1
Product Documentation	2.14	9	4	0	3	2
Configuration Management	1.8	10	1	2	2	5
Environment	3	6	3	0	0	3
Maintainability and Stability	1.25	6	0	1	3	2
Number of commits and bug reports	3	5	5	0	0	0
Quality of Testing Procedures	0	10	0	0	0	10
Standards	2.75	8	3	1	0	4
Requirements	0	4	0	0	0	4
Project Planning	0	9	0	0	0	9
Roadmap	1	3	0	0	3	0

Example Assessment (OJS, OMM)

Trustworthy Element	Rating	Total number of practices	Fully Implemented Practices	Partially Implemented Practices	Not Implemented Practices	Practices not Rated
Overall Rating	**89.66%**	78	52	5	1	20
Licenses	3	9	9	0	0	0
Product Documentation	2.88	9	8	1	0	0
Configuration Management	2.75	9	6	2	0	1
Environment	2.83	6	5	1	0	0
Maintainability and Stability	3	6	5	0	0	1
Number of commits and bug reports	3	5	5	0	0	0
Quality of Testing Procedures	2.71	10	6	0	1	3
Standards	2.75	8	3	1	0	4
Requirements	0	4	0	0	0	4
Project Planning	3	9	2	0	0	7
Roadmap	3	3	3	0	0	0

Do your own OMM Assessment

- OMM documentation: http://qualipso.icmc.usp.br/OMM/

- Google Sheet template for running an OMM assessment: https://bit.ly/2E5idXb

OSS in an Institutional Setting

- IT willingness and ability to support
- Alignment with institutional goals
- Branding concerns
- Project longevity (position, not person)

USING OSS

Installation

- Depends on size and scope of project.
- Most projects are likely to have executable files that install just like proprietary software.
- Smaller or more obscure projects may only distribute source code via GitHub or the command line, or via other means.
- Useful to read the documentation before getting started—make sure you know what you're getting into!

Updates and Maintenance

- Most OSS projects make large updates available via a downloadable patch.
- Not all projects will notify you of updates within the software itself—although many do.
- For smaller or more obscure projects, may need to use a command line interface or manually update the source code in other ways.
- Remember to back up data before updating!

Getting Help

- OSS is community-based!
- Unofficial support from other users on forums, chat rooms, and mailing lists.
- Official support from developers and power users. (Bug reports, pull requests)
- Large projects may offer paid hosting and paid customer support options.

Getting Involved

- OSS is **community-based**!

- Even if you're not an expert coder, there are always ways you can help:
  - Check or create documentation
  - Help new users with common errors
  - Project management

- Check out https://www.firsttimersonly.com/ for more suggestions and to find projects actively looking for help from people new to OSS.

QUESTIONS?

You can also e-mail me: bakersc@mail.wou.edu

Sources

Agile Alliance (2001). The Agile Manifesto. Retrieved from https://www.agilealliance.org/agile101/the-agile-manifesto/

Bretthauer, D. (2001). Open Source Software: A History. Retrieved from https://opencommons.uconn.edu/libr_pubs/7/

Free Software Foundation (2019). What is Free Software? Retrieved from https://www.gnu.org/philosophy/free-sw.html

Morgan, L. & Finnegan, P. (2007). Benefits and Drawbacks of Open Source Software: An Exploratory Study of Secondary Software Firms. In Feller, J., Fitzgerald, B., Scacchi, W., & Sillitti, A. (Eds.), *Open Source Development, Adoption and Innovation* (307-312): Boston, Springer.

Open Source Initiative (n.d.). The Open Source Definition (Annotated), version 1.9. Rerieved from https://opensource.org/osd-annotated

Raymond, E.S. (2000). *The Cathedral and the Bazaar*. Retrieved from http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/

Stol, K. & Babar, M.A. (2010). A comparison framework for Open Source software evaluation methods. In P.J. Agerfalk, C. Boldyreff, J.M. Gonzalez-Barahona, G.R. Madey, & J. Noll (Eds.), *Proceedings from OSS 2010: 6th International IFIP WG Conference on Open Source Systems* (389-394). London: Springer.

Wheeler, D.A. (2015). Why Open Source Software? Look at the Numbers! Retrieved from: https://dwheeler.com/oss_fs_why.html