JOURNAL OF INNOVATION IN HEALTH INFORMATICS

Editorial

Open source paradigm: A synopsis of *The Cathedral and the Bazaar* for health and social care

Tim Benson
R-Outcomes Ltd and Abies Ltd, Newbury, UK and UCL Institute of Health Informatics, London, UK

Accepted April 2016

ABSTRACT

Background Open-source software (OSS) is becoming fashionable in health and social care, although the ideas are not new. However, progress has been slower than many had expected.

Objective The purpose is to summarise the Free/Libre Open Source Software paradigm in terms of what it is, how it impacts users and software engineers and how it can work as a business model in the health and social care sectors.

Method Critical appraisal of key learning from Eric Raymond's seminal book *The Cathedral and the Bazaar*, which was the first comprehensive description of the open-source ecosystem, set out in three long essays.

Outputs The first part contrasts open and closed source approaches to software development and support. The second part describes the culture and practices of the open-source movement. The third part considers business models.

Conclusion A key benefit of open source is that users can access and collaborate on improving the software if they wish. Closed source code may be regarded as a strategic business risk that that may be unacceptable if there is an open-source alternative.The sharing culture of the open-source movement fits well with that of health and social care.

Keywords: delivery of health care, medical informatics, open access to information, software

INTRODUCTION

Open source in health and social care is more than 50 years old.[1,2] It is also referred to as Free/Libre Open Source Software. The Internet runs largely on open-source software (OSS), as do some successful EHR systems, such as VistA in the USA[3] and openMAXIMS in the UK.[4] NHS England has introduced an OSS programme.[5,6] However, it has taken time for OSS to become mainstream in health and social care.[7,8] Predictions that health and social services would rapidly adopt OSS proved to be premature.[9]

The objective of this paper is to summarise the open-source paradigm in terms of what it is, how it impacts users and software engineers and how it can work as a business model.

METHOD

This paper is a critical synopsis of Eric Raymond's seminal book *The Cathedral and the Bazaar*, developed over a 15-year period. The original version of the paper was prepared soon after the book was published in 1999.[10] Direct quotes from the book are used liberally, without reference to specific passages. Raymond's achievement was to provide an early easy-to-read account of what others have subsequently described in more detail, although it has been criticised for appearing to be more innovative than it is.[11]

The cathedral in the title is a direct reference to Fred Brooks' 'must read' classic *The Mythical Man-Month*, originally published in 1975.[12] Brooks praises the conceptual integrity Reims' Cathedral and contends that conceptual integrity is *the most important consideration in system design* (original italics), which must proceed from one mind, or from a very small number of agreeing resonant minds, needing a top-down approach to software development. The *Mythical Man-Month* represents the established wisdom of the waterfall software development method, which Raymond challenges.

In presenting a metaphor of bazaar development against that of cathedral building, Raymond offers open source as an alternative way of working. He defines open source as the process of systematically harnessing open development and decentralized peer review, in order to lower costs and improve software quality. He describes how this method of 'cooperative software development effectively overturns Brooks' Law (adding manpower to a late software project makes it later) leading to unprecedented levels of reliability and quality on individual projects'.

The book is structured around three long essays:

1. *The Cathedral and the Bazaar* compares open-source development with closed-source development;
2. *Homesteading the Noosphere* describes the culture and control of open-source projects;
3. *The Magic Cauldron* shows how open-source business models differ from closed source.

These essays cover most of the key features of the open source paradigm and contrast it with the closed source approach exemplified by Microsoft Office. By legally restricting access to knowledge via proprietary binary-only software licences, Raymond argues that closed source methods result in less freedom for users and slower innovation than open source.

The core idea of open source is to give freedom to suppliers and customers and involve them in product development. However, supplying open-source tools to the market requires new business models.

THE CATHEDRAL AND THE BAZAAR

The first essay, *The Cathedral and the Bazaar*, contrasts the open and closed source development approaches using examples of the history of Linux[13] and Raymond's own project, Fetchmail.[14]

Raymond proposes a set of aphorisms about effective open-source development. The most important ones are listed below in four groups in a revised order. Note that many of these ideas are now mainstream in modern software methodologies. Furthermore, most health care users are risk averse, not keen to be part of a development project and intolerant of software bugs.

Starting off

- Every good work of software starts by scratching the developer's personal itch.
- To solve an interesting problem, start by finding a problem that is interesting to you.
- If you have the right attitude, interesting problems will find you.

Working with users and co-developers

- Release early, release often and listen to or your customers.
- Given a large enough beta tester and co-developer base, almost every problem will be characterised quickly and the fix be obvious to someone.
- Treating your users as co-developers is your least-hassle route to rapid improvement.
- If you treat your beta-testers as if they are your most valuable resource, they will respond by becoming your most valuable resource.
- The next best thing to having good ideas is recognizing good ideas from your users. Sometimes, the latter is better.
- Provided the development coordinator has a medium at least as good as the Internet, and knows how to lead without coercion, many heads are better than one.

Design and development

- Perfection in design is achieved not when there is nothing more to add, much rather when there is nothing more to take away.
- Any tool should be useful in the expected way, but a truly great tool lends itself to uses you never expected.

- Often, the most striking and innovative solutions come from realising that your concept of the problem was wrong.
- Good programmers know what to write – great ones know what to re-engineer.
- Plan to throw one away; you will anyway (Fred Brooks).
- Smart data structures with dumb code works a lot better than the other way around.
- When writing gateway software of any kind, take pains to disturb the data stream as little as possible and never throw away information unless the recipient forces you to.
- A security system is only as secure as it is secret. Beware of pseudosecrets.

Finally

When you lose interest in a program, your last duty to it is to hand it off to a competent successor.

HOMESTEADING THE NOOSPHERE

The second essay *Homesteading the Noosphere* discusses the culture, ownership and control of OSS projects. Governance structures have a powerful impact on the motivation and actions of participants, with freedom being generally correlated with motivation.[15]

The noosphere refers to 'the world of all possible ideas, of which programming projects are a (very) small subset.' The noosphere should not be confused with cyberspace, which is the real world of networks and computers. Raymond suggests that when someone founds a new open-source project, this is analogous to building a homestead on the frontier, and hence 'homesteading the noosphere'.

The ownership and property rights of OSS projects are like land property rights on the frontier. The open source culture has an elaborate set of ownership customs, supported by three taboos:

- Do not allow a project to fork (split into competing projects).
- Do not distribute changes without the co-operation of the moderators (this is usually specified by the open source licence).
- Never remove a person or company's name from the project's history or credits list.

The 'owner' of a software project has the exclusive right, recognised by the community at large, to redistribute modified versions. There are three ways to become an owner, one is to found the project, second is to have ownership of the project handed to you by the previous owner (passing the baton), and the third way is to adopt an orphan.

Raymond suggests that programmer motivation in an open-source project is governed by what anthropologists describe as a gift culture. Participants in a gift culture seek to enhance their status by contributing to the common good. Status is determined by one's reputation among one's peers.

In the jobs market, open software makes it easier for a potential employer to review a candidates abilities as an employee. This provides an additional incentive for programmers to demonstrate their capabilities.

In the absence of survival issues, reputation enhancement becomes a driving goal. This behaviour is also found in academia and amongst clinicians, which is why it is particularly well suited to health computing.

Raymond suggests that a gift culture may be the optimal way to cooperate for generating (and checking) high-quality creative work. Support comes from psychological studies on the interaction between art and reward.

THE MAGIC CAULDRON

The third essay, *The Magic Cauldron*, addresses the issues of how open-source development can sustain itself in a conventional market economy – how to make money at it. First, we must recognise that open source implies that the software is free as in 'free speech' not 'free beer', or in French *libre*, not *gratis*.

Software is a service industry operating under the delusion that it is a manufacturing industry (factory model). This encourages price structures that are way out of line with the actual development costs.

However, most of a software project's whole-life costs accrue in maintenance, support and extensions (services). Therefore, the common price policy of charging a high, fixed purchase price and low service fees leads to results that serve all parties poorly. The incentives cut against a vendor offering good after-sales service.

Raymond asserts that if the vendor's money comes from selling products, most effort will go to making products superficially attractive and shoving them out the door. The help desk, which is not a profit centre, becomes a dumping ground for the least effective staff. Using the product leads to service calls, which cut into profit margins, unless you charge for service. This incentivises 'shelfware' that is sufficiently well marketed to make sales, but useless and not used in practice.

In contrast, open source seeks the largest possible user base to maximise feedback and promote vigorous secondary markets. This suggests a price structure founded on service contracts or subscriptions and the continuing exchange of value between the vendor and the customer. The price a consumer pays is related to the *expected future value of vendor service* (original italics), where service is construed broadly to include enhancements, upgrades and follow-on projects. This forces open-source suppliers towards a service-fee dominated world.

The following example provides an economic explanation of what makes sustainable cooperation. Consider a person using OSS who makes a modest but useful improvement to a program (a patch). There is no market for such patches, so he has two options: to sit on the patch or to throw it into the pool for free. Sitting on the patch gains nothing. Indeed, it incurs a future cost – the effort involved in a remerging the patch into the source base for each new release. On the other hand, throwing the patch into the pool may also gain nothing, or it

may encourage reciprocal giving from others that will address some of your problems in the future. It may also enhance the reputation and status of both the individual contributor and their employer. Overall, sharing is the better option.

Raymond outlines nine business models to create niches in which open-source development can flourish, two not-for-profit and seven for-profit models.

1. *Cost sharing*: a group of users fund open-source development to get a better product at lower cost. (e.g. Apache web server).
2. *Risk spreading*: a program may be released as open source to hedge against the risk that the original developers may leave.
3. *Loss-leader/market positioner*: in this model, OSS is used to create or maintain a market position for proprietary software that generates a direct revenue stream. For example, open-source client software enables sales of server software or subscription/advertising revenue associated with a portal site.
4. *Widget frosting*: this model is for hardware manufacturers, who have to supply software – the frosting on the cake. The vendor has no revenue stream to lose, but gains access to a larger developer pool, more rapid and flexible response to customer needs and better reliability through peer review.
5. *Give away the recipe, open a restaurant*: in this model, OSS is used to create a market for services. One of the business models for OSS is for the *support and maintenance* charges to provide a level of assurance so that legal claims can be made against the supplier if the software is at fault.
6. *Accessorising*: sell accessories to open-source software, such as books and manuals.
7. *Free the future and sell the present*: release the software in binaries and source with a closed licence, but include an expiration date on the closure provisions.
8. *Free the software and sell the brand*: you open source software technology, but charge a commission for certified software that uses the brand name.
9. *Free the software and sell the content*: as ports are made to new platforms, your market for content automatically expands.

Raymond contends that open-source peer review is the only scalable method for achieving high reliability and quality. Customers seeking high reliability and quality will reward software producers who go open source and work out how to maintain a revenue stream in the service, value add and ancillary markets associated with the software. No software consumer would rationally choose to lock itself into a supplier-controlled monopoly by becoming dependant on closed source, if an open-source alternative of acceptable quality is available. If sources are open, the customer has options if the vendor goes belly up.

With closed source software, your key business processes are executed by opaque blocks of bits that you cannot even see inside (let alone modify) – you have lost control of your business. You need your supplier more than your supplier needs you – and you will pay and pay, and pay again for that power imbalance. You pay in higher prices, you pay in lost opportunities, and you pay for lock-in that grows over time as the supplier tightens its hold. In no other industry are the products deliberately kept secret when that secrecy cannot be justified by safety or security concerns.

Contrast this with open source. You have the source code and no one can take it away from you. You now have multiple service companies bidding for your business. You even have the option of building your own support organisation if that looks less expensive than contacting out. The logic is compelling; closed source code is a strategic business risk that is unacceptable if there is an open-source alternative.

Commercial companies can make money out of OSS by charging for services such as distribution, installation, support, warranties and tailoring. These fees are likely to have some relation to the costs involved. The up-front licence fees charged for closed-source software are out of line with the cost structure.

One route forward for public sector funding would be to require that all software developed at the public's expense be licensed as open source.[16] Licensing the software as open source provides more protection for the taxpayer than copyright law.

The sharing culture ethos of the open-source movement fits well with that of health and social care.

Acknowledgement

I am grateful to Paul Cooper of IMS Maxims, Carl Reynolds of Open Health Care UK, Ewan Davis of HANDI and an anonymous reviewer for their valuable comments on earlier drafts.

REFERENCES

1. McDonald C J, Schadowa G and Barnesa M et al. Open Source software in medical informatics – why, how and what. *International Journal of Medical Informatics* 2003; 69(2–3): 175–184. http://dx.doi.org/10.1016/S1386-5056(02)00104-1.
2. Weber S. *The Success of Open Source*. Massachusetts, United States:Harvard University Press, 2004.
3. Noll J. What Constitutes Open Source? A Study of the Vista Electronic Medical Record Software. *Proceedings of the 5th IFIPWG* 2, 13 International Conference on Open Source Systems (OSS 2009), Skövde, Sweden, 3–6 June 2009.
4. Thick M. Trusting data is the next stage of the NHS' digital journey. IMS MAXIMS 2015. Available from http://www.imsmaxims.com/opinion/trusting-data-is-the-next-stage-of-the-nhs-digital-journey/. Accessed 26 March 2016.
5. NHS England. Open Source Programme Update. Available from http://www.slideshare.net/mongodb/nhs-england-open-source-programme-mongo-db-digital-masterclass-november-2014. Accessed November 2014.
6. Liao Y, Fernandes K, Downs D and Foley G. Facilitating the Adoption of Open Source Technologies for UK Health Care:

An Ecosystem Framework. *International Journal of Health Information Management Research* 2013;1(1): 37–47.

7. Reynolds CJ and Wyatt JC. Open Source, Open Standards, and Health Care Information Systems. *Journal of Medical Internet Research* 2011;13(1):e24. http://dx.doi.org/10.2196/jmir.1521. PMid:21447469; PMCid:PMC3221346.
8. Karopka T, Schmuhl H and Demski H. Free/Libre Open Source Software in Health Care: A Review. *Healthcare Informatics Research* 2014;20(1):11–22. http://dx.doi.org/10.4258/hir.2014.20.1.11. PMid:24627814; PMCid:PMC3950260.
9. Carnall D. Medical software's free future: Open collaboration over the Internet is changing development methods. *British Medical Journal* 2000; 321(7267):976. http://dx.doi.org/10.1136/bmj.321.7267.976. PMid:11039943; PMCid:PMC1118811.
10. Raymond ES. *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. California, United States:O'Reilly, 1999.
11. Bezroukov N. A Second Look at the Cathedral and Bazaar.*First Monday* 1999;4(12). Available from http://firstmonday.org/issues/issue4_12/bezroukov/index.html. Accessed 26 March 2016.
12. Brooks F. *The Mythical Man-Month: Essays on Software Engineering*. Boston, United States:Addison-Wesley, 1975.
13. Hertel G, Niedner S and Herrmann S. Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel. *Research Policy* 2003;32(7): 1159–77. http://dx.doi.org/10.1016/S0048-7333(03)00047-7.
14. SourceForge. Fetchmail – the mail-retrieval daemon: Client daemon to move mail from POP and IMAP to your local computer. Available from http://sourceforge.net/projects/fetchmail/. Accessed 24 March 2015.
15. Shay SK. Motivation, governance and the viability of hybrid forms of open source software development. *Management Science* 2006;52 (7):1000–1014. http://dx.doi.org/10.1287/mnsc.1060.0553.
16. Benson T. Medical software's free future: all software developed at public's expense should be licensed as open source. Letter to the editor. *British Medical Journal* 2001;322(7290):863. PMid:11321010; PMCid:PMC1120027.