Open Source Software development as a Complex Adaptive System: Survival of the fittest?

Dr. Albert van Aardt
Information Systems,
Northland Polytechnic, Whangarei
avanaardt@northland.ac.nz

Proponents of Open Source Software (OSS) claim that the OSS model results in software which they describe as being of better quality than "poorly-written commercial software". As a result, some organisations consider OSS-developed software as an alternative to propriety software. The theory of Complex Adaptive Systems (CAS) indicates that self-adjusting and evolving systems tend to "live" longer than "top-down" designed systems. In this paper the question asked is whether the OSS model is a CAS, and therefore whether or not OSS software would be more robust (and thus be more cost-effective). The main finding of this paper is that OSS probably leads to better quality software, but some precautions need to be taken.

Keywords

Open Source Software; Complexity, Computing research.

1. THE RESEARCH PROBLEM

Many authors have pointed out that the quality of software leaves a lot to be desired, and that many projects are canceled due to poor software. Paulk *et al.* (2001:3-4) give the following examples:

1. Seventeen American Department of Defense software contracts found that the average of twenty eight-month schedules was missed by twenty months.

2. Deployment of the B1 bomber was delayed owing to a software problem.

3. USD 58 billion A12 aircraft program was canceled partly due to software problems.

One can add the failure of the London Stock Exchange's Taurus share settlement system in March 1993 and the failure of the NZ Police INCIS system in 1999.

Proponents of the OSS model were quick to point out that it tends to provide much more robust and relatively error-free systems than Closed Source Software (CSS). Raymond (1999) makes the point that "many eyeballs tame complexity". By this he means that, the more people investigate and check the source code of programs, the more difficult it is for bugs to slip through. This peer-review attribute of Open Source is being touted as the solution to the problem of software failure. Geer *et al* (2003) concludes that the lack of peer review in CSS is a serious security threat.

In this paper this claim is investigated from a different perspective. The theory of Complex Adaptive Systems (CAS) is put in context of Open Source Software (OSS), and some conclusions are reached. In a nutshell this reduces to the question, "Is OSS a CAS? and if so, what are the implications for systems development? The fundamental research question to be answered is: *Can OSS provide a long term solution to the problem of poorly written software?*

2. DEPARTURE: "SOFTWARE"

The term "software" as used in this paper refers to programs used to process data and produce meaningful information; in particular, meaningful business information. The basic concept is that of "software as a tool" - the end product is not the consumption of the software, but producing information: a report, a bank statement, a graph to analyze statistics. The end user, in fact, is focused on the informational output of the software rather than the software itself.

This is in contrast with software where the end user is focused on the software, for example, com-

puter games. The user acquires the software for the sake of the software – and of course, this type of software has only limited functionality. While *software as a tool* can produce many different outputs, *software as a game* is in fact a consumable product. Once the consumer has played a game, they need to buy a new game if they want something different: you cannot play chess with a pacman game.

This is an important distinction; tools versus consumable products, which distinction has not yet been clearly made in the literature. Manufacturers of CSS are vigorously attempting sell their products as consumable, throw-away software, hence the cycle of continuous "upgrades". Although the process of development may be the same in both instances, the end-user's expectations are quite different. Additionally, software used for mission-critical business applications needs to be far more robust than (for example) games. Miscalculation in a spreadsheet program has far more serious implications than miscalculation in a game.

This paper therefore considers "software" to mean "mission critical tools", rather than games: a distinction surprisingly frequently overlooked by critics of the OSS model such as Healy (2004)

3. OPEN SOURCE SOFTWARE DEVELOPMENT AS A COMPLEX ADAPTIVE SYSTEMS

"Open Source" software is a term used to describe software that is developed by volunteers and freely distributed, typically via the Internet. The "Tux" website describes it as : "Our primary focus is supporting and advocating the development and use of software and systems whose source code and specifications were openly developed and are freely available to the public." (http://www.tux.org/)

The Open Source Initiative has a more detailed description at http://www.opensource.org/docs/definition.php - but the main point is the same, i.e. that OSS is developed and freely shared by contributors from around the world. (The word "free" refers to "freedom", as in "free to change the source code". It does not mean free as in "no-cost". OSS can be sold at whatever price the market will bear).

In a previous paper ("Applying complexity theory in business information systems", NACCQ, 2001), I argued that any Information System displays the characteristics of a CAS. However, it would be prudent to recall the definition of a CAS at this stage.

According to Dooley (1996) a CAS behaves/evolves according to three key principles (quoted verbatim):

a. Order is emergent as opposed to predetermined

b. A system's history is irreversible

c. A system's future is often unpredictable

The basic building blocks of the CAS are agents. Agents are semi-autonomous units that seek to maximize some measure of goodness by evolving over time. Agents scan their environment and develop schema representing interpretive and action rules. These schema are by definition incomplete and changing."

I argued in that paper that an "ordinary" information system would be somewhat like a CAS, but that typically the final objective of an IS is mostly predetermined. In other words, order is predetermined – the way that an IS is supposed to appear and behave is first established via the Functional Specifications, and then the system is developed to achieve those goals. The classic Systems Development Life Cycle (SDLC) and its variants typically attempt to define what the user requirements are before any development starts. This is in contrast with a CAS, where the final outcome is seldom known.

Holland (1995) showed that, all other factors being equal, a CAS would survive and grow easier than a "top-down" system. This, Holland argued, was owing to the incremental changes and adaptation of a CAS compared to the more rigid structure of a system designed top-down. Using this perspective, one can evaluate the OSS model as a CAS:

3.1 "Goodness"

Raymond (1998) and Perens (2001) argue that the programmer "has an itch to scratch", indicating that many programmers write programs purely for the intellectual enjoyment of solving problems. This can be interpreted in terms of Maslow's (1970) Hierarchy of Needs as being "self actualization". From this viewpoint, one could argue that the program-

mer's need for self-actualization is the "goodness" being maximized. The OSS programmer writes code to satisfy an intellectual need rather than simply to earn a living, as one would find with CSS programmers.

3.2 "Environmental Scanning"

Secondly, the informal manner in which OSS programmer co-operate via the Internet can be seen as the "environmental scanning". The Internet has no doubt made it very easy for a programmer to obtain information (for example, a specific piece of code or a software library) and incorporate that into their current program.

3.3 "Schema"

In the third place, from this environmental scanning, a programmer or group of programmers establish a set of rules, basically defining how a project is going to be developed. For example: the Source Forge website (http://sourceforge.net/) reports over 77,000 projects and more than 800,000 programmers. All of the active projects have a project leader, but programming work is done in a collaborative manner, according to certain rules. In other words, websites such as Source Forge can be seen as a manifestation of "schema representing interpretive and action rules".

3.4 "Unpredictable Future"

Linus Torvalds (1991), the creator of Linux, described his project as "This is a program for hackers by a hacker. I've enjoyed (sic) doing it, and somebody might enjoy looking at it and even modifying it for their own needs." This is taken from his now-famous email in which he called for participation. (Newsgroups: comp.os.minix Date: 1991-10-05) Today Linux has grown to become a serious operating system, used by many large organizations – hardly a "program for hackers" anymore.

3.5 "Adaptation"

The continuous, peer-reviewed model of OSS development, often with robust discussion, can be seen as the adaptive process taking place. OSS evolve over time; programmers come and go, but the project grows. As long as there are programmers wanting to improve the tools, the OSS project stays alive, not to say that the process is always smooth. There are a number of conflicts raging in the OSS community on various issues. However, these can be seen as part of the adaptation process - democracy in action, in the purest sense of the word.

The OSS model therefore resembles a CAS much more closely than the traditional SDLC. A far more detailed sociological analysis is made by Kuwabara (2000) in which he comes to a similar conclusion.

4. THE QUALITY OF OSS

There are a number of OSS projects which have earned a reputation of being robust and relatively error-free. Examples of these include the Linux operating system, Mozilla (web browser), OpenOffice (office suite), Apache (web server), MySQL, PostgreSQL, Firebird (database management systems), PHP, Perl and Python (scripting languages). Opinions vary wildly on this question (as can be expected) and heated debates are raging on the relative merits of OSS versus commercial software. (For example, see http://www.microsoft.com/mscorp/facts/default.asp ; and a response at http://searchwin2000.techtarget.com/originalContent/0,289142,sid1_gci948748,00.html) However, the mere fact that this issue is even being debated of course indicates that there certainly is a perception that OSS can compete with CSS. This paper is not going to even try to document all these arguments.

The SDLC has been criticized by various writers as being slow and not producing the desired results. (Watkins, 2003). Alternatives to the classic SDLC include the so-called "extreme programming" approach (see http://www.extremeprogramming.org/): an attempt to reduce the time taken to develop systems and provide more "user satisfaction". The main thrust of these alternatives is to shorten the feedback loop between the user and the programmer, so that functionality can be improved and bugs removed much earlier than in the SDLC model.

Proponents of the OSS model point out that, by using the Internet, programming bugs are being fixed quite quickly and that, being open, bugs are found more quickly than in CSS. Although no independent statistical figures are available comparing "apples with apples", logic indicates that it would make sense that "many eyeballs tame complexity". In addition, one could also argue that the quality of work from a person programming purely for intellectual pleasure would be better than that of the program-

mer simply doing a job. One may ask, however, whether this guarantees that the software will always be better. Once the itch has been scratched, so to speak, why would the programmer keep on scratching? The Source Forge website has many thousands of dormant and abandoned projects – many of them with a note from the programmer(s) stating "I am too busy right now to do any more work on this project." This is surely indicative of the fact that, in terms of Maslow's model, certain lower level needs must be attended to as well. In other words, even the most noble programmer must eat.

It is also telling us that many of the successful OSS projects have had (or still have) substantial backing (read: money) from commercial software companies. Mozilla used to be Netscape; OpenOffice came from Sun's StarOffice; Firebird was Borland's Interbase. Additionally, the flagship OSS project, Linux, currently has a question mark behind it in the form of the SCO Group's Intellectual Property court case against IBM, even though this is hotly disputed (http://www.groklaw.net/index.php). This begs the question, "Can any big software project be developed on purely OSS principles, without commercial backing of some sort?" The picture of the lone programmer slaving late at night in some cramped basement to produce bug-free code simply must be challenged (Lancashire, 2004).

5. CONCLUSION

Holland (1995) has shown that a CAS has a much better chance to survive in the long term than a top-down designed system. Therefore it would be logical to conclude that an OSS project would be more robust than a CSS project. Many OSS projects evolve over time, with programmers changing or existing programmers revisiting their previous work to implement improvements. The result of this is software tools that are apparently robust and relatively secure.

However, a number of concerns must be pointed out:

- OSS has no direct commercial market feedback. Users are encouraged to use bug reports, but there is no obligation on the programmers to fix a problem, purely a new "itch to scratch"
- Slower innovation would be expected, because of the voluntary nature of the work.
- The outcome suggests that technical rather than user focus would occur. One should remember that OSS is typically developed by programmers at the "back end" of the system, rather than easy to use "front end" programing.

A tendency in the software industry is evident to view each new development as a "paradigm shift": a breathtaking silver bullet that will solve all system development problems, from the first relational databases, through GUI's, object orientation, web development.... each touted as a major breakthrough. Of course, the reality is that all these developments are simply part of an unfolding industry. OSS is part of this landscape, and can, in some areas, be cost-effective to use. But there is no Santa Claus: there is also no magic cure for the woes of the software industry.

As Bezroukov (1999) puts it: "Open source development is now fashionable and it makes big news. The news too often emphasizes achievements and successful projects, but fails to address difficulties, failures and aborted projects..... little is known about how Internet-based virtual teams (IVT) really operate and what problems develop in that sort of co-operation."

Levesque (2004) states that there are five major problems with OSS:

1. User interface design
2. Documentation
3. Feature-centric development
4. Programming for the self
5. Religious blindness

On the other end of the scale, some benefits can be observed:

- Higher stability, leading to lower maintenance cost.
- A piecemeal approach probably has higher chance of survival than purely market one – especially in a monopolistic marketplace.
- Security appears to be better implemented in OSS (Geer *et al* 2003).
- A number of businesses that have implemented OSS found substantial savings http://www.computerweekly.com/articles/article.asp?liArticleID=129881&liFlavourID=1

http://techupdate.zdnet.com/techupdate/stories/main/0,14179,2860180,00.html

http://www.wired.com/news/infostructure/0,1377,62236,00.html

http://www.varbusiness.com/sections/News/breakingnews.asp?ArticleID=49098

http://www.e-cology.ca/canfloss/report/CANfloss_Report.pdf

- Support is freely available on the Internet, and a number of commercial support companies offer services, such as Novell, IBM, Xandros and Red Hat.

- Many of the criticisms against OSS flows from a lack of understanding of the process - once clarified, these concerns tend to disappear (Wheatley, 2004).

Finally, this study asks, "Can OSS provide a long term solution to the problem of poorly written software?" The answer - at this stage - must be: it depends. It is true to say that mission critical business systems are being run on OSS (Google and Amazon being two prime examples), but there is a very large number of applications lacking in the OSS world. The answer to the question would probably be that businesses most certainly should consider OSS for areas where these have matured, e.g. database management systems, web servers, operating systems and some "office" type applications. Failure to do so could cost organisations their competitive edge in the near future. Unfortunately, OSS lags in a number of areas, where CSS have a definite edge: accounting, CAD, workflow control, various bespoke packages. The prudent IS professional will no doubt consider both OSS and CSS.

However, there is no reason why businesses should not harvest the creative force of OSS programmers. Recognition for OSS programmers in the way they want it is nothing but a sound business investment. Frankly, businesses are in business for themselves, not to support other companies. If OSS can provide a more secure and less costly solution than CSS, it would make sense to use it where appropriate.

REFERENCES

Bezroukov, N. "Open Source Software Development as a Special Type of Academic Research" http://firstmonday.org/issues/issue4_10/bezroukov/

Dooley, K. 1997: "A Complex Adaptive Systems Model of Organization Change," Nonlinear Dynamics, Psychology, & Life Science, Vol. 1, No. 1, p. 69-97
(http://www.eas.asu.edu/~kdooley/casopdef.html)

Feller, J. and Fitzgerald, B. (2002) Understanding Open Source Software Development, Addison-Wesley, UK

Geer, D. Pfleeger, C. P., Schneier, B Quarterman J.S., Metzger, P. Bace, R Gutmann, P: (2003) "Cyberinsecurity: the cost of monopoly" (http://www.ccianet.org/index.php3)

Healy, T (2004) "Has Open Source Reached Its Limits?" http://www.ipi.org/ipi%5CIPIPublications.nsf/PublicationLookupFullText/F4992D9C7780355786256E49001E7595

Holland, J.H. (1995): "Hidden Order": Addison-Wesley. Reading, MA

Kuwabara, K 2000 "Linux: a bazaar at the edge of chaos" http://www.firstmonday.org/issues/issue5_3/kuwabara/index.html#k6

Lancashire, D. (2004) "The fading altruism of Open Source development" http://www.firstmonday.dk/issues/issue6_12/lancashire/

Levesque, M, 2004: "Fundamental issues with open source software development" - http://firstmonday.org/issues/issue9_4/levesque/index.html

Maslow, A. H. Motivation and Personality (2nd ed.), New York: Harper and Row,
1970 (http://web.utk.edu/~gwynne/maslow.html)

Perens, B (2001) http://perens.com/Articles/

Raymond, E (1998) (http://www.catb.org/~esr/writings/)

Watkins, J. A. Multimethodology: An Alternative Management Paradigm to Process Quality Improvement, PhD. Thesis. Rand Afrikaans University, 2003

Wheatley, M. 2004 "The Myths of Open Source" http://www.cio.com/archive/030104/open.html