

OPEN SOURCE: TOWARDS SUCCESSFUL SYSTEMS DEVELOPMENT PROJECTS IN DEVELOPING COUNTRIES

Rinette Roets (rroets@ufh.ac.za)
University of Fort Hare

MaryLou Minnaar (maryloumin@absamail.co.za)
University of Fort Hare

Kerry Wright (kwright@ufh.ac.za)
University of Fort Hare

Abstract: Open source software (OSS) has recently become the subject of scrutiny and debate, not least for the promise it holds to solve systems development challenges. This paper investigates the benefits as well as the challenges of open source development. While significant process benefits exist, open source is not without several limitations to more widespread use and implementation. The report notes the specific application to developing countries.

Despite these flaws, open source remains highly suited, both as a software product and as a development methodology. However, no standard development lifecycle exists. A model of the open source development process life cycle is derived from current OSS models. The model is intended to be used as an alternative development framework that potentially solves many problems associated with systems development. This holds the promise of increasing OSS development, which in turn can benefit organisations in developing countries.

Keywords: open source software; OSS; systems development; developing countries

OPEN SOURCE: TOWARDS SUCCESSFUL SYSTEMS DEVELOPMENT PROJECTS IN DEVELOPING COUNTRIES

1. INTRODUCTION

Open source has recently attracted significant interest from the IT community. It has been hailed as the panacea for the ills of traditional software development, criticised as a threat to innovation, intellectual property and the application industry as a whole, and seen as a solution to the ICT woes of the developing world. Whatever the perspective, there is little doubt as to the impact the open source paradigm has on the processes of the development of information systems.

In developing countries, especially for Governments, the costs associated with licensing fees and the problem of software not meeting the perhaps idiosyncratic needs specific to developing countries is a cause for concern. There has thus been a great deal of interest in using Open Source Software with reasons given as cost-cutting and skill development, amongst others. However, no standard development lifecycle exists. This paper therefore attempts to answer the question “What Systems Development Lifecycle would be suitable for OSSD in developing countries?”.

The paper starts by examining how the free and open source software phenomenon can alleviate system development challenges, and then focuses on the relevance of these features for developing countries. The paper next examines the distinguishing features of open source software development (OSSD) that represent potential benefits to the system development process (as well as the challenges) compared to traditional software challenges. The specific appeal of OSS development for developing countries is noted.

The paper continues with an analysis of existing OSS development life cycle models. It uses a theory based evaluation approach (as the first phase of a research project) in order to devise a proposed life cycle model incorporating aspects of the discussed OSS development models for use in developing countries.

Such a software life cycle would assist in the development of software in developing countries by providing a standard useful for novice designers, and so doing increase usage of OSSD to achieve the benefits outlined in the paper.

The paper concludes by briefly describing the research method to be employed to verify this model.

2. WHAT IS FOSS/OSS

Free/Libre and Open Source Software (FLOSS) refers to software whose licenses give users four essential ‘freedoms’:

- to run the program for any purpose,
- to study the workings of the program, and modify the program to suit specific needs,
- to redistribute copies of the program at no charge or for a fee, and

- to improve the program, and release the improved, modified version (Perens, 1999; Perens, 2004).

OSS users do not pay royalties as no copyright exists, in contrast to proprietary software applications which are strictly protected through patents and intellectual property rights (Asiri, 2003; Wheeler, 2003).

The term “free software” has an unintended meaning, “Software you can get for zero price,” which fits the term just as well as the intended meaning, “software which gives the user certain freedoms.” OSS is software for which the source code is publicly available, though the specific licensing agreements vary as to what one is allowed to do with that code. (Stallman, 2007)

3. PROBLEMS WITH TRADITIONAL DEVELOPMENT PROJECTS

Proponents of open source argue that ‘traditional’ software development projects suffer from various ills. Such projects have been shown to be prone to time and cost overruns, are largely unmaintainable, with questionable quality and reliability. The 1999 Standish Group report revealed that 75% of software projects fail in one or more of these measures, with a third of projects cancelled due to failure. In addition, systems often fail to satisfy the needs of the customer for whom they are developed (Sommerville, 1995).

These failures are ascribed to:

- Inadequate understanding of the size and complexity of IS development projects coupled with inflexible, unrealistic timeframes and poor cost estimates (Hughes and Cotterell, 1999; McConnell, 1996).
- Lack of user involvement is touted as contributing to project failure (Addison and Vallabh, 2002; Keider in Frenzel, 1996; Hughes and Cotterell, 1999; McConnell, 1996).
- Shortfalls in skilled personnel: Team members with insufficient technical expertise, managerial skill or knowledge about the problem domain can affect project success (Addison and Vallabh, 2002; Boehm, 1991; Frenzel, 1996; Hughes and Cotterell, 1999; Satzinger, Jackson and Burd, 2004; Turban, Mclean and Wetherbe, 2002).
- Project costs are further exacerbated by the price of license fees for software and tools required for application development as well as add-on costs for exchange controls.

The last two points are of specific concern to developing countries. The issue of skill is particularly pertinent to developing countries. Developed countries have a relative glut in IT skills. In developing countries sourcing skilled individuals for projects is a struggle due to a lack of training or the brain drain (Kunda and Brooks, 2003). The cost of software licenses is problematic for the software industry in developing countries, which tend to rely on donations or concessions from software companies such as Microsoft (Frenzel, 1996; Kunda and Brooks, 2003; Ould, 1999).

4. BENEFITS OF OSS

Does OSS have answers to these problems? It is claimed that OSSD produces reliable, high quality software in less time and with less cost than traditional methods. Adelstein (2003: 1) is even more evangelical, claiming that OSSD is the “most efficient” way to build applications. Schweik and Semenov (2003: 1) add that OSSD can potentially “change, perhaps dramatically, the way humans work together to solve complex problems in computer programming”. While there is a level of exaggeration, OSS should be afforded serious consideration as an alternative to traditional software development.

Raymond (1998a: 1) likens OSSD to a “bazaar” – a loosely centralised, cooperative community where collaboration and sharing enjoy religion status. Conversely, traditional software engineering is referred to as a “cathedral” where hierarchical structures exist and little collaboration takes place.

The OSSD model has the following features:

- Collaborative, parallel development involving source code sharing and reuse
- Collaborative approach to problem solving through constant feedback and peer review
- Large pool of globally dispersed, highly talented, motivated professionals
- Extremely rapid release times
- Increased user involvement as users are viewed as co-developers (Feller and Fitzgerald, 2000; FLOSS Project Report, 2002; Scacchi, 2003; Weerawarana and Weeratunge, 2004).

4.1. Quality Software

Furthermore, it is maintained that OSS features result in quality software as collaborative development allows for multiple solutions. At the same time there is little tolerance for failure to adhere to the tacitly accepted norms (FLOSS Project Report, 2002; Valloppillil, 1998). Reliability of products has withstood the test of time (Netcraft, 2004; Weerawarana and Weeratunge, 2004).

4.2. Development Speed

Reuse of code implies speedier development: the more people there are creating code and adding value to a project, the quicker the product is released and becomes valuable to a user group (Scacchi, 2003).

4.3. User Involvement

Users are treated as a valued asset in the development process. Viewing users as co-developers leads to code improvement and effective debugging. If encouraged, users can assist developers in finding system faults and improvements, thereby reducing the need (and cost) for extra developers to perform the same function (Raymond, 1998a; FLOSS Project Report, 2002).

4.4. Access to existing code

Developers have access to the “open source toolset” (Adelstein, 2003: 4), a huge amount of open source project code which can speed up development.

4.5. Collaboration

A further important feature of the OSSD model is the nature of the development community. Large numbers of geographically dispersed programmers are joined by the Internet to produce complex software. They do so largely without pay. Reasons for participation in open source projects range from challenge and improving skills, to altruism and fun, as well as for financial reward (FLOSS Project Report, 2002; Hars and Ou, 2001; Rajani, 2003).

4.6. Cost

Total cost of ownership (TCO) of OSS is widely debated, particularly within developing countries. A basic tenet of open source is that all source code is free and available to any user to modify and improve. By contrast proprietary software requires paid-for licensing that is generally considered a barrier to access for developing countries. The cost of Windows XP and Office XP is about US\$560 (Ghosh, 2003) which equates to approximately 2.5 months of GDP per capita in South Africa, and 16 months in Vietnam.

In some phases of ownership, there is evidence that OSS may have advantages in the area of TCO. OSS can be tested without cost. Once acquired, OSS has no license fees, removing the necessity to purchase additional licenses as the organisation grows. Throughout usage and maintenance, where the bulk of TCO is typically spent, software can be fixed or configured by in-house developers due to the availability of source code (Weber, 2003; Wheatley, 2004).

5. PERCEIVED DISADVANTAGES OF OSS

5.1. Speed of development

Critics question whether open source provides a rapid development environment and suggest that the result could be slower given the absence of formal management structures. The open source community is likened to a “large, semi-organised mob with a fuzzy vision” (Bezroukov, 1999; Levesque, 2004; Valloppillil, 1998; Zawinski in Bezroukov, 1999).

5.2. User involvement

Strong user involvement and participation throughout a project is a central tenet of OSSD. However, involving users closely can become problematic as users tend to create bureaucracies which hamper development (Bezroukov, 1999).

5.3. Scope creep

Because the open source community is meritocratic and ego-driven, open source projects run the risk of falling prey to feature creep – a primary software risk. Levesque (2004) suggests that programmer credibility within the community is often viewed as more important than the philosophy of ‘keeping things simple’ while providing the required functionality.

5.4. Releases

OSS is premised on rapid releases and typically has many more iterations than commercial software. This creates a management problem as a new release needs to be implemented in order for an organisation to receive the full benefit. The informal requirements analysis process is problematic amongst CIOs as it is impossible to predict what newer versions of a particular piece of software might include, and whether these newer versions will continue to support business needs (Farber, 2004; Valloppillil, 1998).

5.5. Usability issues

OSS is traditionally perceived to be ‘code-centric’, targeted mainly at high-end power users. Less attention is focussed on who the potential audience might be and what their user interface needs are. Generally, the user interfaces of open source products are not intuitive (Levesque, 2004; Valloppillil, 1998; Wheatley, 2004).

5.6. Support issues

Wheatley (2004: 2) mentions the lack of accountability from a single vendor. While open source projects have a wide variety of resources (developers themselves, Internet mailing lists, archives and support databases) that can be tapped for support, the problem is that there is no single source of information, no help desk that provides ‘definitive’ answers to problems. Open source developers are not contracted and therefore cannot be forced into creating documentation (Bezroukov, 1999; Levesque, 2004).

5.7. Cost and development

System deployment and training is often more expensive with OSS as it is less intuitive and does not have the usability advantages of proprietary software.

5.8. Credibility issues

Flagrantly anti-Microsoft sentiments often expressed in the name of open source serves to discredit the value of the open source product (Levesque, 2004).

5.9. Tools

The confusion surrounding the wide variety of open source licensing models (Farber, 2004) is a risk. Many CIOs are concerned about the implications of implementing code that they cannot verify the right to use. Fear of legal action is a principal deterrent to using OSS as part of an enterprise solution.

6. ISSUES FOR DEVELOPING COUNTRIES

6.1. Independence from monopolistic suppliers

A noticeable trend is emerging in which many governments table policy regarding state use of OSS, largely motivated by savings in cost. Governments everywhere encourage the use of OSS as a means to curb software maintenance and acquisition costs (Weber, 2003). The South African government views foreign currency savings as an explicit motivator for open source use (Government Information Technology Officers Council, 2002). Countries want to minimise their over-reliance on an elite group of suppliers. Thus, particularly for developing countries, lower total cost of ownership, and the independence from commercial monopolistic models make open source an alternative to proprietary systems. Its use allows organisations to avoid software lock-in, and maintain autonomy and control of corporate information systems (Ghosh, 2003; Weber, 2003; Weerawarana and Weeratunge, 2004).

6.2. Technological development

When a country converts to OSS, Steffen (2003) argues that there is the potential for development of a local software industry. The skills needed to install and manage OSSD will, in theory, be a by-product of this industry. While Steffen verges on being overly optimistic, the potential for local IT industries to be empowered exists, which in turn can create development and further employment opportunities (Weerawarana and Weeratunge, 2004).

In addition, internationalisation of software is a by-product of the open source movement. Given the global nature of the developers and users, functional requirements are not only directed from, for example, the USA. This allows for development of applications that specifically fit the indigenous needs of emerging markets in developing countries (FLOSS Project Report, 2002; Valloppillil, 1998 Weber, 2003). OpenOffice has been released in several languages other than English, namely Zulu, Northern Sotho and Afrikaans (Translate.org.za, 2004).

6.3. Intellectual property rights

Internationally, there is an increased emphasis on enforcing intellectual property rights. Threatened with sanctions and other punitive action, governments and organisations are being forced to distance themselves from piracy (Weber, 2003). Organisations are looking increasingly to open source because of cost and compliance with intellectual property rights. The South African Government Information Technology Officers Council on OSS (2002) has recognised the threat of usage of unauthorised proprietary software.

6.4. New business opportunities

The advent of the OSS phenomenon has resulted in new business ventures, including the distribution and retail of open source products, and the provision of several open source related services. Also, OSS is generally complex to use and user interface issues are not considered a priority. This gap in usability opens a business opportunity. This is particularly relevant in the developing world where service and support is perceived to be lacking for open source products (FLOSS Project Report, 2002).

6.5. Skills development

The IT skills shortage in developing countries necessitates the development of such skills. The OSS model, with the existence of code examples which novices can use, as well as the collaborative aspect of development can provide for unique experiential learning. The tenet that users can co-develop is perhaps unrealistic in developing countries, although their involvement is part of mutual learning.

6.6. Threats and limitations of OSS

In the context of developing countries, several specific critical factors are required for the success of open source conversion. Internet access is vital for participation in the open source community as most communication takes place via e-mail or through development portals. Heeks (1999) points to limited Internet access due to a lack of telecommunications links, reliable electricity supply and sufficient computers. A good educational infrastructure is necessary to foster a culture of learning (FLOSS Project Report, 2002). Freedom of information is necessary for a thriving open source community. (The exception here is China, with a large open source community despite restrictions on freedom). Finally, English-skilled developers are necessary in an environment where English remains the lingua franca.

7. MODEL FOR DEVELOPMENT

Mockus, Fielding and Herbsleb (2000) suggest there are several basic differences between OSSD and traditional methods. Firstly, OSS systems are built by large numbers of people, largely volunteers, although increasing numbers of OSS projects are supported by companies. Secondly, work is not assigned; rather individuals choose to participate in specific project activities. Thirdly, there is no clear design process, at either a system or detailed level (Vixie, 1999). In addition, there is no explicit project plan, list of deliverables or schedule.

All these differences suggest a weakening of traditional process models. However, successful projects such as Linux and Apache and others are testimony to the viability of the OSSD model as a successful alternative to traditional software models. While there is proof that the OSSD paradigm can produce high-quality software with less cost and often in less time, there are few models to understand how this is achieved. This report examines the importance of adopting a life cycle approach to direct the development process.

7.1. Traditional models

The systems development life cycle (SDLC) approach can be viewed as a management framework that leads to “well-engineered software” (Sommerville, 1995: 6). An SDLC comprises many different phases or sets of related activities, depending on the process model that is adopted. There are, however, generic phases into which all project activities can be organised: planning, analysis, design, implementation and support (Satzinger et al, 2004). These activities can be organised different ways, often referred to as process models (Hughes and Cotterell, 1999: 64).

Different process models or software development approaches exist. Each has specific sets of related activities and deliverables, representing adaptations of the generic SDLC. In an attempt to define the OSSD process, researchers have compared OSSD with several of the more traditional models found in commercial environments. These are briefly, the waterfall model with sequential phases, which has been the standard model for large projects (Hughes and Cotterell, 1999; Satzinger et al, 2004; Sommerville, 1995). The waterfall model has limitations, and other models have been put forward to overcome the rigid and lengthy

process that is resistant to changing specifications. Iterative and incremental development methods, as well as Boehm's spiral model and prototyping, are models that break tasks up into smaller modules in different ways. More recently the RUP model, XP (eXtreme programming) and agile modelling have also been introduced. Despite the differences, the phases identified in the waterfall model are still generally adhered to.

There is no universal project process model that simultaneously shortens a project lifespan, decreases cost and increases quality on all projects (Glass, 2004). There is also no single process model that adequately describes OSSD.

7.2. Existing OSS development models

One critique of the open source methodology is that it is too loosely defined (McConnell, 1999), resulting in perceptions that OSSD is largely chaotic.

Several researchers have proposed life cycle models derived from analyses of successful open source projects. Opinions differ as to the stages that comprise a typical open source development project. However, regardless of the open source life cycle model that may be subscribed to, the OSSD paradigm demonstrates several common attributes:

- parallel development and peer review,
- prompt feedback to user and developer contributions,
- highly talented developers,
- parallel debugging,
- user involvement, and
- rapid release times.

7.2.1. Comparative model

Vixie (1999) holds that an open source project can include all the elements of a traditional SDLC. Classic OSS projects such as BSD, BIND and SendMail are evidence that open source projects utilise standard software engineering processes of analysis, design, implementation and support, albeit informally.

Vixie (1999) suggests that during the analysis phase requirements for a new project are often based on what open source developers themselves want or need. What is ultimately included in a piece of software is battled out over the Internet. Developers reach consensus on requirements.

System-level and detailed design is often the casualty in the process. Design issues are often inherent in the system that is being built, and therefore, assumed to be widely understood, or they evolve or are reverse-engineered over time. Either way, the fact that design issues are not made visible, or written down anywhere limits the quality and credibility of the project.

Implementation and coding is really why developers involve themselves in open source projects. Here people experiment, post code and wait for peer acknowledgement. There are no formal review procedures, but given the nature of the community, feedback is almost immediate.

Testing in open source projects involves large numbers of developers, often users of the software, reading and scrutinising the code for errors. The danger of an unstructured, informal testing phase is lessened by the advantages of having “uncounted strangers” with real world experiences looking for bugs (Vixie, 1999: 6).

In his comparison between OSSD and the traditional SDLC, Vixie (1999) recognises the fundamental differences that the OSS life cycle present, but fails to suggest an appropriate model that analyses this new process.

7.2.2. Organisational models

Schweik and Semenov (2003) propose an OSSD project life cycle comprising three phases: project initiation, going ‘open’, and project growth, stability or decline. Each phase is characterised by a distinct set of activities.

Project initiation occurs for similar reasons as outlined in the initial stages of Vixie’s model. Developers decide to take on projects for a variety of reasons. During project initiation, the project core is developed upon which others build. Project initiation is premised on modularity, such that future development is organised around small manageable pieces. The advantages are: multiple programmers can work on the same module; competition for the best solution code increases quality; and there is greater control over project progress (Schweik and Semenov, 2003; Torvalds, 1999).

Going ‘open’ involves a choice on the part of the project founders to follow OSS licensing principles. It also ensures that the project enjoys the support of a core group of dedicated developers, shows technical promise and is of interest to future developers, and that a sufficient amount of the original requirements have been solved to create a framework in which future development can take place (Schweik and Semenov, 2003).

In this phase appropriate technologies and web sites need to be chosen to act as a vehicle for sharing code and recruiting developers. Some web sites offer project hosting services, including version management, problem tracking, and other project management tools. Possibly the most important aspect of this phase is developing operational designs to provide some level of project leadership in the absence of clearly defined hierarchies (Schweik and Semenov, 2003: 8).

The final phase, growth, stability or decline, poses an element of risk for open source projects: will the project generate enough interest to attract developers and users globally to use the product and participate in further programming, testing or documentation (Schweik and Semenov, 2003)?

Wynn (2004) proposes a similar open source life cycle but introduces a maturity phase in which a project reaches critical mass in terms of the numbers of users and developers it can support due to administrative constraints and the size of the project itself.

The OSSD life cycle models proposed by Wynn (2004) and Schweik and Semenov (2003) provide interesting insight into the managerial aspects and organisational structure behind a typical open source project. However, these models do not provide a task-related analysis of the OSSD process.

7.2.3. Task-related models

Several researchers have derived life cycle models from investigating successful open source projects such as Apache and the FreeBSD Project (Mockus et al, 2000; Jorgensen, 2001). Mockus et al (2000) describe a life cycle that combines a decision-making framework with task-related project phases. The model comprises six phases:

- Roles and responsibilities,
- Identifying work to be done,
- Assigning and performing development work,
- Pre-release testing,
- Inspections, and
- Managing releases.

The model has a strong managerial focus emphasising developer management and the work to be done, rather than on product-related activities. While there is more detail regarding task-related issues than the organisational models (Schweik and Semenov, 2003; Wynn, 2004), it fails to do so in sufficient detail to be considered a true development process model. Process models incorporate at some level all the phases of the traditional SDLC.

The model proposed by Mockus et al adequately caters for the planning phase of the SDLC but is less explicit regarding other phases. Furthermore, Mockus et al assume that some sort of prototype already exists, failing to explain where design and analysis phases occur within their model.

Jorgensen (2001) provides a more detailed description of specific product related activities that underpin the OSSD process. The model (fig. 1) explains the life cycle for changes that occurred within the FreeBSD project.

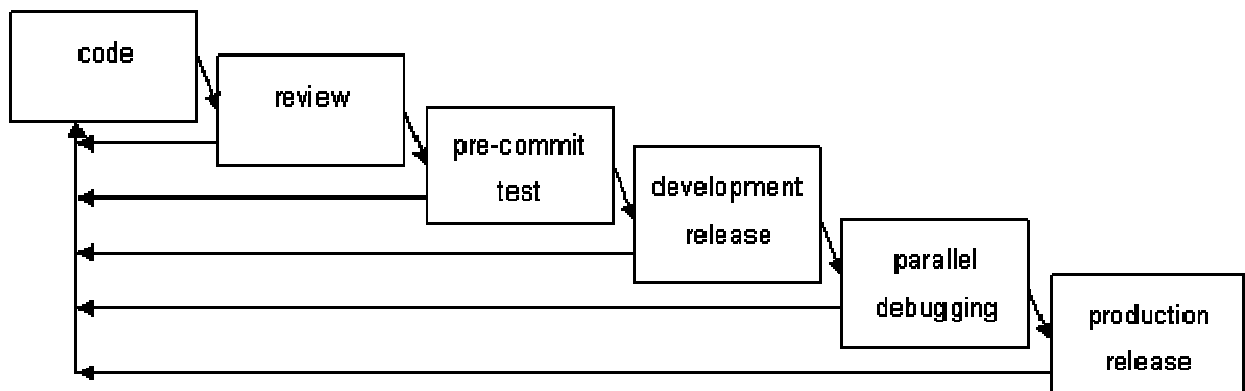


Figure 1. Jorgensen life-cycle, 2001

Several stages or sets of activities are proposed:

- **Code:** Code is submitted by talented developers for review and improvement by respected peers.

- **Review:** Most (if not all) code contributions are reviewed. This independent peer review is a central strength of the OSSD process.
- **Pre-commit test:** Review is followed by an unplanned, yet thorough, testing of all contributions for a particular code change. While informal, this phase is taken very seriously as negative implications of permitting a faulty contribution can be considerable.
- **Development release:** If the code segment is deemed release-ready it may be added into the development release.
- **Parallel debugging:** Development releases of software undergo a rigorous debugging phase where any number of developers is able to scrutinise the code in search of flaws.
- **Production release:** Where development versions are deemed stable, they are released as production versions.

The process is repeated incrementally for new modules – reinforcing the cyclical nature of all open source projects where there is no real end point - unlike many commercial projects.

Jorgensen's model is widely accepted (Feller et al, 2001; FLOSS Project Report, 2002) as a framework for the OSSD process, on both macro (project) and micro (component or code segment) levels. However, flaws remain. When applied to an OSS project, the model does not adequately explain where or how the processes of planning, analysis and design take place.

7.2.4. Summary of existing OSS project models

Existing OSSD models explain some aspect of the OSSD approach. However, an explanation of where the analysis and design phases occur within the project life cycle remains distinctly absent. Existing models seem to start at the implementation phase, either assuming details regarding initial SDLC or ignoring them completely.

7.3. Proposed Model

The proposed model (fig. 2) expands on Jorgensen's life cycle model (fig. 1) and incorporates aspects of previous models, particularly that of Schweik and Semenov (2003). In addition, the proposed model attempts to encapsulate the phases of the traditional SDLC.

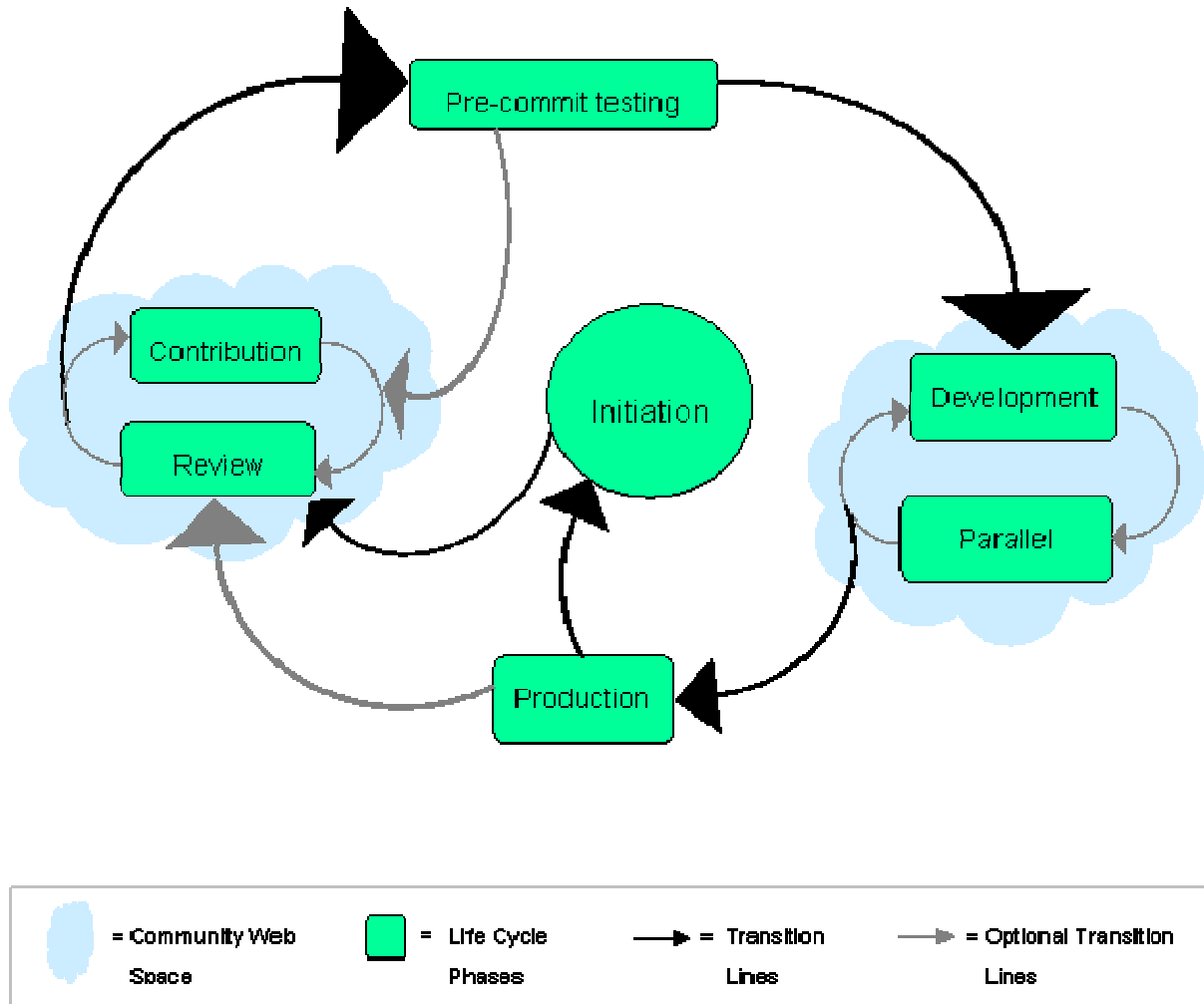


Figure 2. Proposed life cycle model of OSSD projects

The introduction of a generic initiation phase replaces Jorgensen’s code phase. The initiation phase can be applied at both the micro and macro level of a project. This first phase refers to developed code that is used as a prototype for further progress on a particular project, be that code a small code segment or the initial version of an entire project. The phase is undertaken either by a developer who develops a piece of code for an existing project, or the project founder in a new project.

The initiation phase moves into a cycle of code review and further contribution. The number of iterations occurring at this phase is dependent on the interest that the code segment, component or entire project generates within the developer community. Independent peer review and prompt feedback characterise this phase. The cycle of code contribution and review take place within the wider Internet developer community, employing tools such as e-mail, bulletin boards and discussion groups.

Once a piece of code is considered adequate for inclusion in a development release, pre-commit testing is performed to ensure that this new piece of code, once added, does not break the existing release. Testing is usually performed by core developers. No rigorous testing schedule exists and, indeed, the process is not even required. However, the consequences of allowing faulty code into a development release can be severe for the programmers involved and in turn the reputation of the project as a whole.

A process of debugging and reincorporation of code into the development release then takes place. This is again an iterative process occurring within the community web space. No formal planned debugging occurs; individuals volunteer. This is another area exemplifying the strength of open source projects. The more people that seek, find and remove bugs, the better the quality of the software.

Eventually, code forms part of a production release which is generally managed by a core developer. Production releases take the form of a prototype that can be used in the initiation phase of the next iteration of that project, component or code segment.

It is useful to compare the proposed model to the traditional SDLC (fig. 3), as most process models encompass these phases in some way. Planning, analysis and design phases are undertaken largely by the project founder. Rather than glossing over design issues, as is the perception of open source projects, it is suggested that it may be even more important to get design right prior to actual programming so that all developers are working towards a clearly defined common purpose. As such, the OSS development life cycle is largely positioned in the traditional implementation phase of the SDLC (Feller et al, 2001; The FLOSS Project Report, 2002).

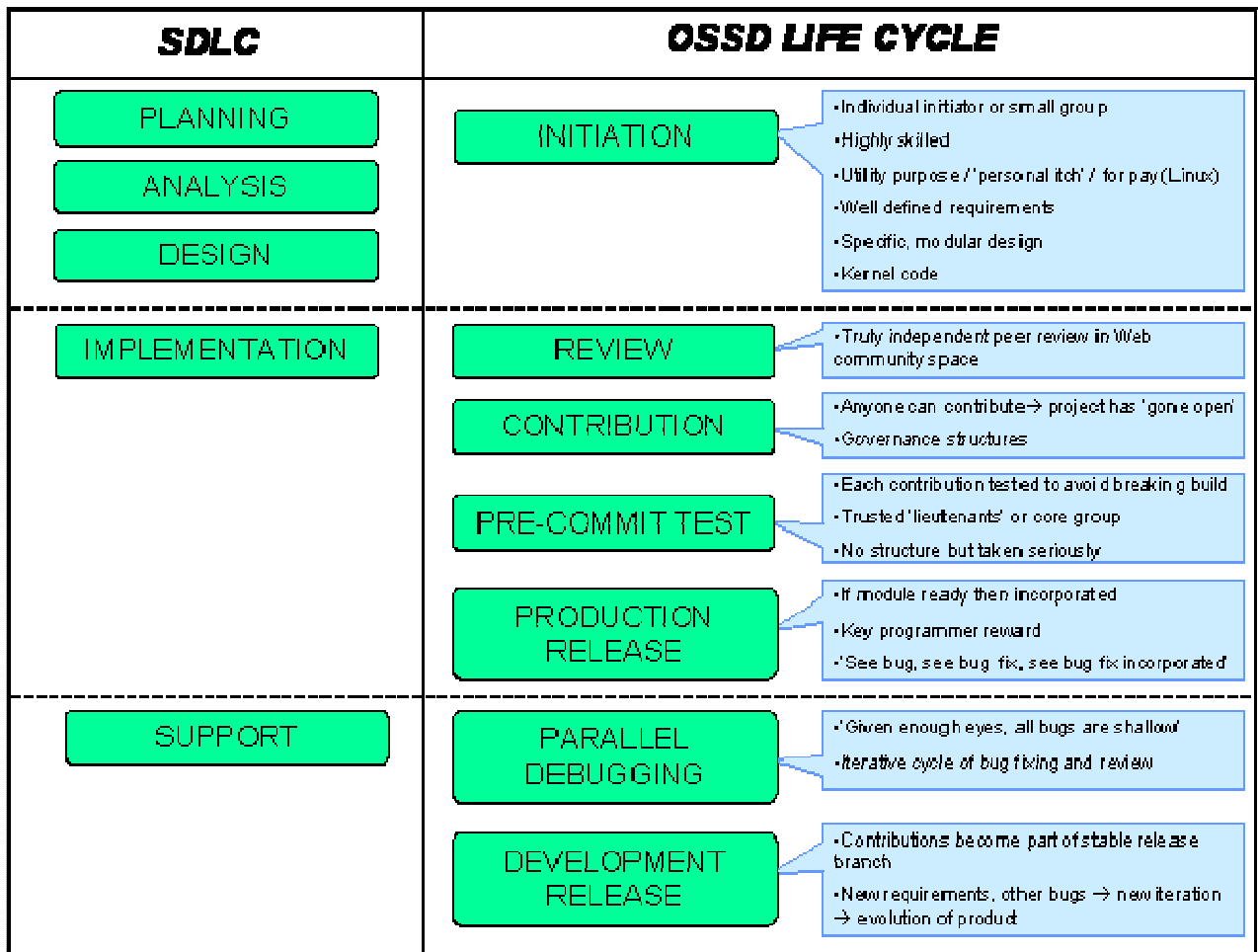


Figure 3. Comparison of SDLC and a proposed model of OSS

7.4. Model Validation

The next phase of the project is to test the model empirically. OSS development is taking place at the Office of the Premier of the Provincial Government in the Eastern Cape. A case

study methodology will be used to assess existing practice in the Government and compare it to that proposed in the model. Interviews will be conducted with the developers (programmers and analysts), as well as project initiators and users in order to assess the perception of the value of the projects, the OSS environment and then more importantly, the nature of the development cycle. The model will be critiqued in the light of the findings.

8. CONCLUSION

The purpose of this research paper was firstly to identify how the open source phenomenon can alleviate system development challenges faced by organisations, particularly within the developing world where the quoted failures of traditional development are a luxury that developing countries cannot afford. OSSD combines features found in traditional software processes with other features in a unique way that can potentially produce high-quality software, faster and cheaper within the rapidly changing Internet environment.

Although OSSD is not a faultless solution, it provides potential benefits and opportunities to the system development process. The most obvious benefit is the reduced cost that open source provides. In addition, open source offers new business models that can be exploited, particularly within developing economies, and has an added benefit of skills development.

There is, however, no standard development lifecycle to produce OSS. The main purpose of the paper was therefore to arrive at a model for an OSSD development lifecycle. Such a model is proposed which, by facilitating OSS development, would provide benefits for developers in developing countries, specifically in terms of promoting improved programming skills, through the availability of expertise and model code, as well as software cost reduction.

9. REFERENCES AND CITATIONS

- Addison, T., & Vallabh, S. (2002). *Controlling software project risks - an empirical study of methods used by experienced project managers*. Paper presented at the Proceedings of SAICSIT 2002 (ACM), Port Elizabeth.
- Adelstein, T. (2003). *How to misunderstand open source software development*, Retrieved 3rd May 2004, from <<http://www.consultingtimes.com/ossedev.html>>
- Asiri, S. (2003). Open source software. *Computers and Society*, 32(5).
- Bezroukov, N. (1999). *Open source software development as a special type of academic research: Critique of vulgar Raymondism.*, Retrieved 3rd May 2004, from <http://firstmonday.org/issues/issue4_10/bezroukov/>
- Boehm, B. (1991). Software risk management: principles and practices. *IEEE Software*, 8(1), 32-41.
- Farber, D. (2004). *Six barriers to open source adoption*, Retrieved 24th March 2004, from <<http://www.techupdate.zdnet.com/>>
- Feller, J., & Fitzgerald, B. (2000). *A framework analysis of the open source development paradigm*. Paper presented at the Proceedings of the 21st International Conference on Information Systems, Brisbane.
- Feller, J., & Fitzgerald, B. (2001). *Understanding open source software development*. London: Addison-Wesley.
- FLOSS Project Report. (2002). *Floss Project Report: Free/Libre and open source software (FLOSS): Survey and study*, Retrieved 24th March 2004, from <<http://www.infonomic.nl/floss/report/>>
- Frenzel, C. (1996). *Management of Information Technology. 2nd ed.*, Cambridge, MA: CTI
- Ghosh, A. (2003). *Open source: A case for developing countries*, Retrieved 24th March 2004, from <<http://www.infonomics.nl/floss/papers/>>
- Glass, R. L. (2004). Practical programmer: Matching methodology to problem domain. *Communications of the ACM*, 47(5), 19-21.

- Government Information Technology Officers Council. (2002). *Using open source software in the South African government*, Retrieved 2nd August 2004, from <<http://www.oss.gov.za/osspolicyframeworkv1.pdf>>
- Hars, A., & Ou, S. (2002). Working for free? Motivations for participating in open-source projects. *International Journal of Electronic Commerce*, 6(3), 25.
- Heeks, R. B. (1999). International perspectives: Software strategies in developing countries. *Communications of the ACM*, 42(6), 15-20.
- Hughes, B., & Cotterell, M. (1999). *Software project management. 2nd ed.* Berkshire, United Kingdom: Mcgraw-Hill.
- Jorgensen, N. (2001). Putting it all in the trunk: Incremental software development in the FreeBSD open source project. *Information Systems Journal*, 11(4), 321-336.
- Kunda, D., & Brooks, L. (2003). *Component-based software engineering for developing countries*. Retrieved 12th May 2004, from <<http://www-users.cs.york.ac.uk/~kimble/research/dc-paper.pdf>>
- Levesque, M. (2004). *Fundamental issues with open source software development*, Retrieved 3rd May 2004, from <http://firstmonday.org/issues/issue9_4/levesque/>
- McConnell, S. (1996). Best practices: Classic mistakes. *IEEE Software*, 13(5).
- McConnell, S. (1999). Open source methodology: Ready for prime time. *IEEE Software*, July/August.
- Mockus, A., Fielding, R. T., & Herbsleb, J. D. (2000). Two case studies of open source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology*, 11(3), 309-346.
- Munro, R., Negrette, J., & Aitken, A. (2003). *Engaging with the open source community (Part One): What is the open source community?*. Retrieved 25th March 2004, from <<http://www.theinquirer.net>>
- Netcraft. (2004). *Web server survey*. Retrieved 2nd August 2004, from <<http://news.netcraft.com/archives/2004/07/index.html>>

- Ould, M. (1999). *Managing Software Quality and Business Risk*. Chichester: John Wiley & Sons, Ltd.
- Perens, B. (1999). The open source definition. In M. Stone, S. Ockman & C. Dibona (Eds.), *Open sources: Voices from the open source revolution*. Sebastopol, California: O'Reilly & Associates.
- Perens, B. (2004). *The open source definition*, Retrieved 4th May 2004, from <http://opensource.org/docs/def_print.php>
- Rajani, N. (2003). *Free as in education: Significance of the Free/Libre and open source software for developing countries*, Retrieved 25th March 2004, from <<http://www.maailma.kaapeli.fi/flossreport1.0.html>>
- Raymond, E. (1998a). *The cathedral and the bazaar*, Retrieved 24th March 2004, from <http://firstmonday.org/issues/issue3_3/raymond/>
- Satzinger, J. W., Jackson, R. B., & Burd, S. D. (2004). *Systems Analysis and Design in a Changing World. 3rd ed.* Boston: Course Technology.
- Scacchi, W. (2003). *When is free/open source software development faster, better and cheaper than software engineering?.*, Retrieved 3rd May 2004, from <<http://www.ics.ucl.edu/~wscacchi/papers/new/scacchi-bookchapter.pdf>>
- Schweik, C. M., & Semenov, A. (2003). *The institutional design of open source programming: Implications for addressing complex public policy and management problems*, Retrieved 11th May 2004, from <http://firstmonday.org/issues/issue8_1/schweik/index.html>
- Sommerville, I. (1995). *Software Engineering. 5th ed.* Harlow: Addison-Wesley Longman Limited.
- Stallman, R (2007). Why “Free Software” is better than “Open Source”. Retrieved 27 March 2007, from <<http://www.gnu.org/philosophy/free-software-for-freedom.html>>
- Steffen, A. (2003). *World changing essays: Redistributing the future*, Retrieved 17th March 2004, from <<http://www.worldchanging.com/archives/000789.html>>
- The Standish Group International, Inc. (1999). *Chaos: A recipe for success*, The Standish Group International, Inc.

- Torvalds, L. (1999). The Linux edge. In M. Stone, S. Ockman & C. Dibona (Eds.), *Open sources: Voices from the open source revolution*. Sebastopol, California: O'Reilly & Associates.
- Translate.org.za. (2004). *Newsflash*, Retrieved 2nd August 2004, from <<http://www.translate.org.za>>
- Valloppillil, V. (1998). *Open source Initiative (OSI) Halloween I: A (new?) software development methodology*, Retrieved 3rd August 2004, from <<http://www.opensource.org/halloween/halloween1.php#comment28>>
- Vixie, P. (1999). Software Engineering. In M. Stone, S. Ockman & C. Dibona (Eds.), *Open sources: Voices from the open source revolution*. Sebastopol, California: O'Reilly & Associates.
- Weber, S. (2003). *Open source software in developing economies*, Retrieved 25th March 2004, from <http://www.ssrc.org/programs/itic/publications/itst_materials/webernote2.pdf>
- Weerawarana, S., & Weeratunge, J. (2004). *Open source in developing countries*, Retrieved 17th March 2004, from <<http://www.sida.se/publications>>
- Wheatley, M. (2004). *The myths of open source*, Retrieved 3rd August 2004, from <<http://www.cio.com/archive/030104/open.html>>
- Wheeler, D. (2003). *Why open source software/free software (OSS/FS)? Look at the numbers!*, Retrieved 28th March 2004, from <<http://www.dwheeler.com>>
- Wynn, D. E. (2004). *Organizational structure of open source projects: A life cycle approach*. Paper presented at the Proceedings of the 7th Annual Conference of the Southern Association for Information Systems, Savannah.