Open Incremental Model- A Open Source Software Development Life Cycle Model (OSDLC)

Sourav Mandal
Assistant Professor,
Computer Sc. & Engineering
Haldia Institute of Technology
Haldia, West Bengal, India

Shyamalendu Kandar
Assistant Professor,
Computer Sc. & Engineering
Haldia Institute of Technology
Haldia, West Bengal, India

Palash Ray
Assistant Professor,
Computer Sc. & Engineering
Haldia Institute of Technology
Haldia, West Bengal, India

ABSTRACT
Open-source software abbreviated as OSS is computer software that is available with source code and is provided under a software license that permits users to study, change, and improve the software. For the commercial software the source code and certain other rights are normally reserved for copyright holders,i.e. the company who developes the software. A group of people in a collaborative manner often developes the Open source software, not under the roof of a large organization. This strategy makes open source software cheap, reliable and modifiable if needed.

In this context we shall discuss mainly the features of Open Source Software, Existing Open Source Software development models and our proposed model named open incremental model.

Keywords

OSS, OSDLC, Agile methodology , Open Incremental Model

1. INTRODUCTION
Open Source software offers significant benefits, compared to typical commercial products. Commercial products often stress on advancement and updation of visible features for getting marketing advantages. It is very difficult to measure qualities attributes such as stability, security, reliability etc. in case of Commercial Software. Commercial software put notices basically on the quality of mostly used features. Whereas Open Source software developing community consists of very bright, very motivated developers, who are mostly unpaid but are very disciplined to their work In addition to that all the users of Open Source software have access to the source code of the software and debugging tools.[1,2]. For this reason the users can suggest the developers about the bugs by feedback or they can fix the bugs if possible by modifying the source code and even can enhance the software by providing actual changes to the source code. Because of the availability of source code and right to modify the code by users, sometimes the quality of software produced by the Open Source software development community exceeds the quality of same type software produced by purely commercial organisations.[2]

As Open Source Software are not made by a group of people under a common roof,[3] for this the open source software does not follow the conventional model like waterfall[4], iterative enhancement, spiral [4, 5]etc. Even there is no standard open source development model also. Some open source developers use some model as their own.

In this paper we are going to discuss some of those models and have proposed a new model called open incremental model, which can be used for open source software development.

In this paper Section 2 describes the features of Open Source Software, Section 3 describes some existing open source software development model, Section 4 describes proposed Open Incremental model, Section 5 puts light on Validation of the proposed model, Section 6 describes the future scope and Section 7 draws the conclusion.

2. FEATURES OF OSS
A source software has several features of their own. Some of the features of Open Source Software are discussed below[1].

Free Redistribution: License of the software should not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license should not require a royalty or other fee for such sale.

Source Code: The source codes are open. The executable program or software should be accompanied with source codes so that it can be easily distributed or modified as needed.

Derived Works: Any person may modify, or update, or reuse the source code as required. The derived works also should be published and is allowed to redistribute under same term and condition as the license of the original software.

Author's Source Code Integration: Configuration management or change management is very tough for this kind of software. The license may restrict source code for modification but it allows the distribution of patch files with source code for modification. The derived work may come up as new version or release under the same license.

No Discrimination Against Persons or Groups: There are no such limitations or restrictions for selecting persons or group for modification or up gradation. For getting maximum benefit, the maximum diversity of persons and groups should be equally eligible to contribute to open sources.

No Discrimination against Fields of Endeavor: it may not restrict the program from being used in a business, or research. But it does not permit to be used commercially.

Distribution of License: The rights automatically distributed with the program to the co developers with out the need for additional license by those parties.

License Must Not Be Specific to a Product:

The program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights.

License Must Not Restrict Other Software: There is no restriction for other software to be open source that are distributed along with the licensed open source software.

License Must Be Technology-Neutral: The license may not highlight or refer any individual technology or style of interface. It should not bind with any specific hardware or software of Proprietorship Company.

3. EXISTING OPEN SOURCE SOFTWARE DEVELOPMENT MODEL

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,
- parallel debugging,
- user involvement, and rapid release times
- highly talented developers

Before describing the newly proposed model, we want to describe some other models that are somehow used in some open source software development projects. Mainly we have explored two models. One model is proposed by the United States Department of Defense (DoD) .The model is as follows.[6]

Major features regarding this model are

1) OSS is developed by collaborative process.

2) Most OSS projects have some web location as "trusted repository" where people can get the "official" version of the program or software also some related important information (documentation, bug report system, mailing lists, etc.). Users or developers can get the software directly from the trusted repository, or get it through distributors. Distributors are who acquire it and provide additional value such as integration with other components, testing, special configuration, support, and so on.

3) The trusted developers are developers who are allowed to modify the trusted repository directly. At project start, the project creators or initiator are mainly the trusted developers and they determine who else may become a trusted developer of this initial trusted repository. All other developers can improve the software by changing local copies and also can post their versions to the internet. But they must submit their changes to a trusted developer to update the trusted repository.

4) Users can send bug reports to the distributor or trusted repository and can be taken care accordingly

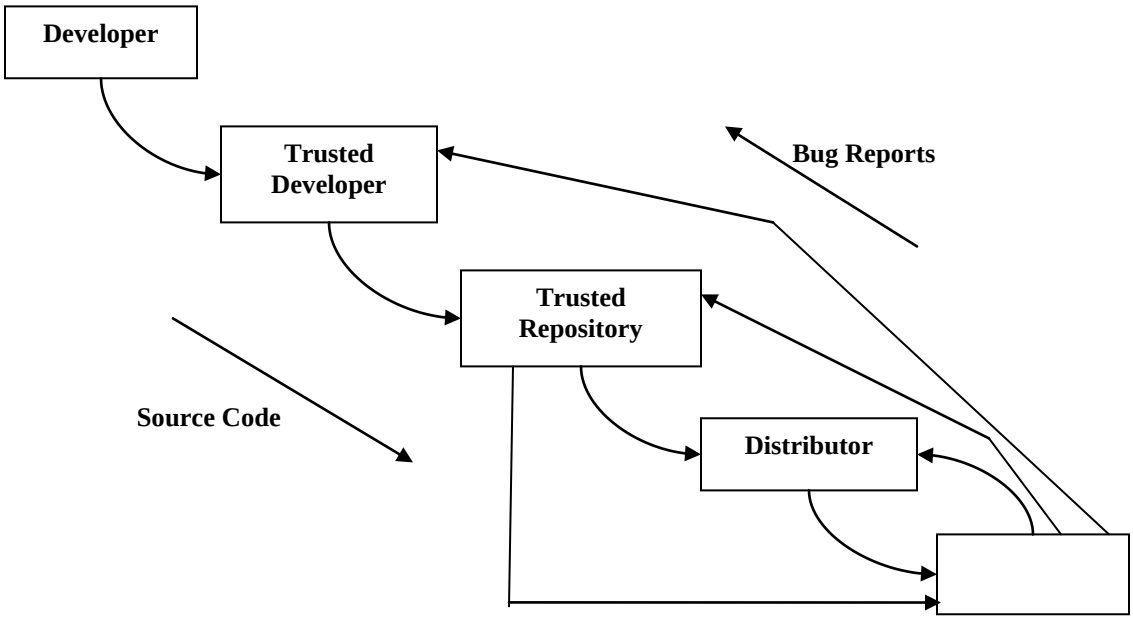

Figure 1.: OSS model proposed by DoD, USA

Another model we like to discuss is **'Agile methodology'** for software development. It is based on two popular SDLC 'iterative' and 'incremental' development model where requirements and solutions evolve through collaboration between self-organizing and cross-functional teams. This particular methodology suits better for OSS development.[7,8,9]

In February 2001, 17 a group of software developers met at Ski Resort in Snowbird, Utah, to discuss lightweight development methods. From their discussion a a manifesto named "Manifesto for Agile Software Development" was published to define a new approach of software development now known as agile software development. Some of the manifesto's authors formed a non-profit organization named Agile Alliance, that promotes software development according to the manifesto's principles.

Principles behind the Agile Manifesto [7,8]

a. Satisfying the customer through early and continuous delivery of valuable software.

b. Welcome changing requirements, even late in development.

c. Deliver working software frequently

d. Business people and developers must work together daily throughout the project.

e. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

f. Face-to-face conversation is the best form of communication (co-location)

g. Working software is the primary measure of progress.

h. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

i. Continuous attention to technical excellence and good design.

j. Simplicity.

k. Self-organizing teams

l. Regular adaptation to changing circumstances

The Agile model is described in Figure 2.

Agile methodology is very flexible to use. The opportunity given by this model to the user makes it suitable for OSS development also. More or less it's consisted of 5 phases. [10] Brainstorming is running among various people for requirement analysis all over the world using a common forum to gather the features to be included in the software according to their need. Then primary design along with the relevant documents are made and kept in the web. Those are controlled by a forum and supplied on request to the motivated developer all over the world. After some addition, the developers sends their version along with source code. Other people give their feedback or report bugs. The source codes are modified accordingly and after consecutive verifications those are added with original source code. Gradually over a certain period of time the software moves to the maturity level.

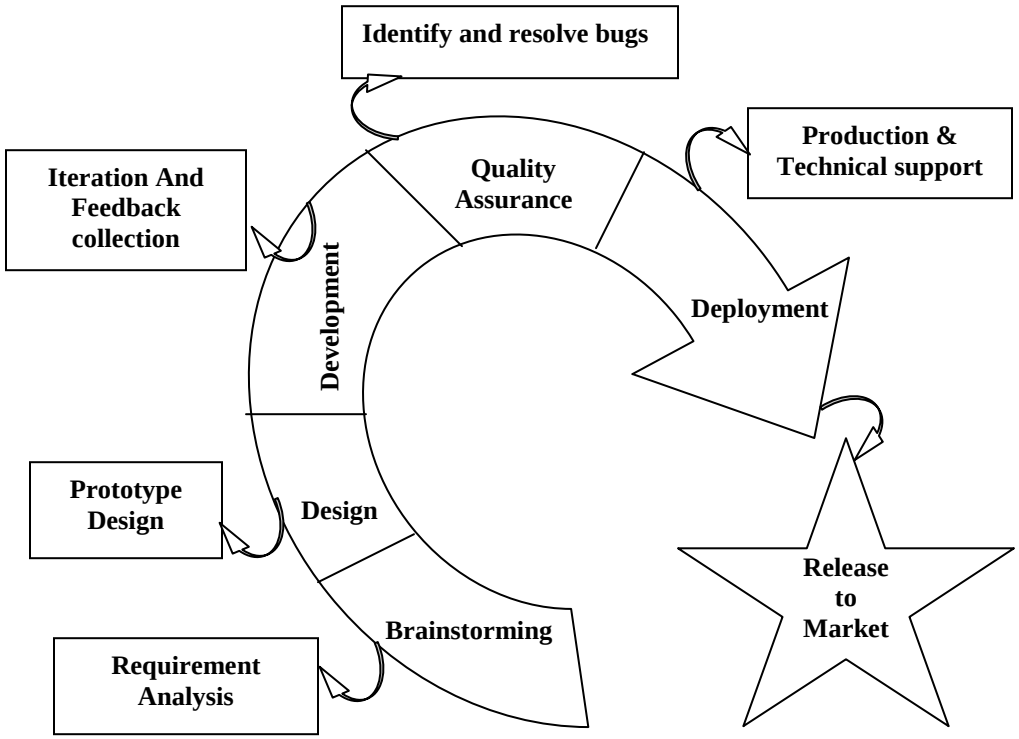

Figure2: Agile Methodology for Software Development

4. OPEN INCREMENTAL MODEL

Major OSS are initiated by some group of people from different organizations in voluntarily basis and contributed by all interested people of different country and culture. It some how may follow Evolutionary or Incremental Model as it is started from very small unit and gradually increased. But as this kind of software is not managed by a single organization the requirement specification is confusing so that design is dependent on individual perception and skill. But all interested people create an open forum regarding the development procedure where they put different queries and answered by all as possible. Or sometimes the first group of people who takes the initiation is creating an organization and other people join accordingly.

There are mainly 11 steps in the newly proposed Open Incremental Model. The diagram of the model is described in figure 3.

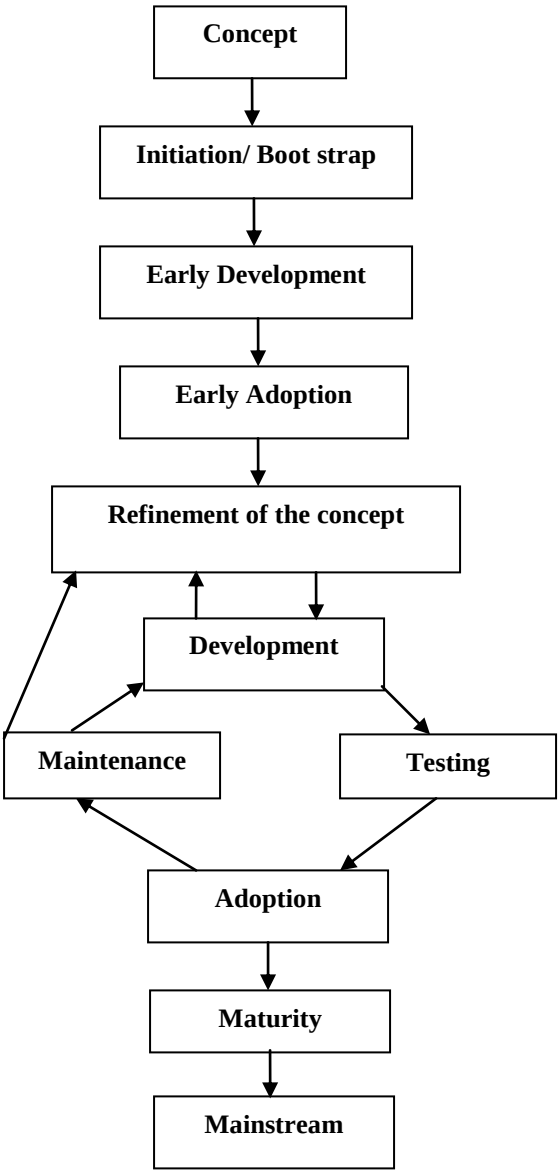

Figure 3: Open Incremental Model

Phase 1: Concept

Some ideas have been thrown around by a few people. Some codes have been written. A project site is up (Google Code, CodePlex, etc) and there is maybe a working prototype to show off and talk about. There are no such documents placed there. A forum or website can be launched due to this purpose.

Phase 2: Initiation/Bootstrap

There are one or two serious authors/committers. There might already be one or two actual users experimenting with the framework. The developments are continuously going on. Few discussions are going on using some blog posts on that forum or few related articles may be published. A controlling authority can be formed who guides or control the development process and program code contributed from various people.

Phase 3: Early Development

A few more committers have signed on, but still there are only one or two primary contributors. A few more users have started playing with it now and are providing feedback. A mailing list has certainly been set up. The build is more stable and works in many environments.

Phase 4: Early Adoption

More committers are signing on now and contributing more and more. The project is at a state with many of the baseline features are done and the product is quite usable now. There are a few dozen users experimenting with it and even building some interesting applications upon it. Multiple people are discussing about it and contributing a bit. The forum/authority is starting to materialize the discussion into categories and sub-topics. Some sparse documentation has started to emerge in the form of some limited API documents; some FAQ's and getting started guides. So after this phase all contributors get a piece of software and concrete guidelines. The tasks carried by controlling authority are increased. Because now they have to maintain the changes and modification made by different contributor through out the world.

Phase 5: Refinement of the concept / specification

In this phase based on the feedback and development scenarios the actual concepts may be refined and displayed as guideline. The prototype or very first version of OSS is now made. So any modification or refinement of the original concept is done. The specification guidelines are very concrete in this phase. This does not mean that the modification of the concept are not done later phase. Any good suggestions can lead to a new version of the OSS at any phase.

Phase 6: Development

There are several devoted committers/contributor now, cranking out serious features and functionality. The beta versions are distributed for free. There are several users using the OSS and post their feedbacks or queries on the forum's wall/blog. The codes are available for download regularly. Some restrictions may be imposed or not to get it. The code is growing by all. It is really taking shape and being fleshed out. At this point documentation is "moderate" and is growing into more scenarios and topics.

Phase 7: Testing/ Validation

The developed code of single contributor is tested individually by the developer and then uploaded to be added with the main program code. The authority may test it again after integrating with the main code. If some modifications required the codes are again send to the actual developer or published in the forum mentioning the bugs so that contributor may debug it accordingly. The forum can take help of the experts also.

Phase 8: Adoption

Many people are submitting patches and other forms of contributions. The inner circle of committers is growing. There are frequent "point" releases available for download. There's now likely a basic installer which helps configure your environment for the framework. Frequent, in-depth blog posts by noted authors. The wiki (A web site or set of web pages that allows almost anyone to edit and add content) is now very rich and there are frequent contributions. Documentation at this point is adequate. The codes are now packaged and are distributed for free or with very nominal cost.

Phase 9: Maintenance/ Support

After adoption of the OSS now it is time for maintaining the software. Maintenance can be many types.

Corrective: Correcting errors that were not discovered during the product development phase. This is called corrective maintenance. The corrections are made available to the users for free or as new release.

Perfective: Improving the implementation of the system, and enhancing the functionalities of the system according to the user's requirement. This is called perfective maintenance.

Adaptive: Porting the software to work in a new computer environment/platform or to cope up with new kind of operating system. This is called adaptive maintenance.

It is a time consuming process and there is no exact guidelines for that. Any type of maintenance may be required any time and may be raised by any user or developer any time. So that it is something like 'as when required'.

Phase 10: Maturity

The maintenance phase may add a new kind of directions and may lead to another phase of development and sometimes refines the requirement. The contributors again start to develop another version of the OSS. The concepts are put in the wiki and guideline also. This process cycle may run for several years or decades and slowly moves to different levels of maturity.

There are so many patches and contributors that a hierarchy has been set up to evaluate and approved changes. Multiple releases create management issues with patches and defects requiring more organization. There over a thousand users by now. There are releases and installers for various scenarios or environment. There is lots of documentation activity at this phase.

Phase 11: Mainstream

In this phase, many of the original contributors have moved on or are less involved. New generations of contributors have taken over and are taking the project into different directions and expanding it greatly. There are hundreds of thousands of users. Perhaps there are even consulting companies forming business services around the project and offering commercial support. The project has its own active web site with lots of content, guides, add-ons, forums, blogs, etc.

5. VALIDATION OF THE PROPOSED MODEL

Studying some OSSDLC models we can conclude that all models are having different characteristics but basic features are similar as stated in section 3.

All other existing models are based on the basic priority of OSSD and useful. They also comprise different advantages of OSSD. Also it is very tough to decide which one is better and what are the relative disadvantages. Because of lacking regulation and discipline among developers it is not very easy to follow any OSSDLC models truly. Also if we categorize the software according to Bohem'1982 [4][5] there are three types of software, Organic, Semi-detached and Embedded, according to size, complexity and resource required for development. Only first two types of software can imply any OSSDLC easily. Also on the basis of the size and complexity of the software some models may be proved useful and efficient than others. Without proper application of various OSSDLC on different OSS development worldwide and surveying thoroughly the effectiveness of different components can not be estimated and compare usefulness. With the already existing models, described in several research papers and case studies; some advantages of using our proposed model validating the basic features of OSS development methodologies are given below.

Quality Software	As collaborative development allows for multiple solutions, OSS features result in quality software. Also There is little tolerance for failure to adhere to the tacitly accepted norms [11][12]. A phase in our proposed model, 'Refinement of Concepts or Specification' truly helps out to achieve best solution and thus increase quality.
Development Speed	Reuse of code increase development speed. The more people are creating code and adding value to a project, the product is released quickly and it becomes valuable to a user group.[13] 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-organized mob with a fuzzy vision" [14] [12]. Our proposed model requires a very

	good project initiation and control and thus eliminates the problem of absence of formal management structure or irregularities if any.
User Involvement	Users are generally treated as a valued resource in the development process. Utilizing users as co developers leads to code improvement and effective debugging. Users can assist developers in finding system faults and improvements, thereby reducing the need (and cost) for extra developers to perform the same function [15][11]. However, involving users closely can become problematic as users tend to create bureaucracies or ego problem, which hamper development [14]. Proposed model emphasizes on controlled user involvement as the 'Testing/ validation' and 'Adoption' phase demands high quality review/feedback and quality control.
Access to existing code	Access to existing code is required for parallel coding, debugging and testing**.** Developers have access to the "open source toolset", a huge amount of other open source project codes which can speed up development. [16]. The proposed model is having these characteristics.
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 and largely without pay. Reasons for participation in open source projects are mainly due to lots of challenge, improving skills, motivated wish for human welfare, fun, as well as for financial reward [11] The proposed model ensures collaboration in large in all phases.
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 organization to receive the full benefit. It is very tough to decide for organizations whether these newer versions will continue to support business needs [12] Our model is having a new release all the time when it completes the 'Adoption' phase. The organizations will use and suggest improvements at general 'Support' or 'Maintenance' phase. That will definitely lead to newer kind formal requirement or concept and initiates another cycle of development.
Support issues	Wheatley [17] 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 [14] The 'Refinement of Concept or Specification' phase of the proposed model leads to documenting the requirements and modification history. Also the 'Validation' and 'Adoption' phase may generate high quality review report that can be used in future development or release. The developers also can do documentation with their coding to increase their code acceptance.

6. FUTURE SCOPE

The advantages and disadvantages of a proposed model will not be clear unless the model is used for developent of some software project. In this resaerch work a new model is proposed keeping in mind the laccunas of some already existing models and efforts have been made to remove those problems. Till it can not be said optimum as it is not used for the implementation of any software project.
In a future work we shall try to use this model for some Open Source Software developemnt to find its strong and weak features. Also we shall try to find the complexity of development, development speed, number of bugs present, best testing methodology etc. of a Open Source Software developed by this model.

7. CONCLUSION

Open source movement is a social movement. All people of the world must be blessed with the advancement of technology to improve their lifestyle. Technology must not be indoored in some companies for their business profit. With that it must be kept in mind that the developed software must be adequate to fullfill the requirement of the new technological advancement of the user. A universally accepted standard model must be there for the development of OSS, so that the developed software become a blessing for a lage number of people scattered in the whole world, not to a limited group of people. Research must go on, on the open source development life cycle to find a universally accepted model, so that user can get a standard software for their uses.

8. REFERENCES

[1] "Characteristics of OSS" http://www.opensource.org/docs/osd

[2] "Benefits of Using Open Source Software" http://open-source.gbdirect.co.uk

[3] "Free Software Movement" http://www.gnu.org.in

[4] Pressman,Roger S, "Software Engineering", McGraw Hill, pp79-88.

[5] Jawadekar, "Software Engineering- Principles and Practices", TMH, pp 18-27

[6] "Open Source Licence Proposal" http://cio-nii.defense.gov/sites/oss/ Open_Source_Software_(OSS)_FAQ.htm

[7] Beck, Kent; et al. "Manifesto for Agile Software Development". http://agilemanifesto.org/. Retrieved 2010-06-14.

[8] Cockburn. A. "Agile Software Development", Addison-Wesley, 2002.

[9] DeMarco, T., Boehm, "The Agile Methods Fray", IEEE Computer, Vol 35, no 6 June 2002, pp 90-92.

[10] Schwaber, K, "Agile Process and Self-Organization" http://aanpo.org/article/index , Agile Alliance, 2002.

[11] Rishab A. Ghosh, Bernhard Krieger, Ruediger Glott, Gregorio Robles, "Free/Libre and Open Source Software: Survey and Study" , International Institute of Infonomics University of Maastricht, The Netherlands June 2002

[12] Valloppillil, V. "Open source Initiative (OSI) Halloween I: A (new?) software development methodology" 1998

[13] Scacchi, W, "When is free/open source software development faster, better and cheaper than software engineering?" Institute for Software Research University of California, June 2004

[14] N. Bezroukov, "Open Source Software Development as a Special Kind of Academic Research (Critique of Vulgar Raymondism)," http://firstmonday.org/issues/ issue4_10/bezroukov/index.html

[15] Eric Steven Raymond, "The Cathedral and the Bazaar"http://www.tuxedo.org/~esr/] 2000

[16] Tom Adelstein, "How to Misunderstand Open Source Software Development", December 1, 2003, http://www.consultingtimes.com/ossdev.html

[17] M. Wheatley, CIO Magazine, "The Myths of. Open Source", March 2004, http:// /www.cio.com/archive/030104/open.html