IJCSMS International Journal of Computer Science and Management Studies Volume 11, Issue 03, Oct 2011
ISSN (Online):2231-5268
www.ijcsms.com

Awareness of Open Source Software (OSS): Promises, Reality and Future

Anil (Student of M.Tech.(CSE))[1], Ashok kumar(Student of M.Tech.(CSE))[2] , Vikas Chahar[3] ,Nidhi Kandhil[4]

[1]Bhiwani Institute of Technology, Bhiwani-127021, Haryana, India
anilverma.bits@gmail.com

[2]Bhiwani institute of technology and science, Bhiwani-127021, Haryana, India
ashokdhirana@gmail.com

[3]Asstt. Prof. Vaish Institute of Management & Technology, Rohtak

[4]Research Scholar, Singhania University, jhunjhunu (Rajasthan)

Abstract

Open source is a development method for software that harnesses the power of distributed peer review and transparency of process. The Open Source Initiative Approved License trademark and program creates a nexus of trust around which developers, users, corporations and governments can organize open source cooperation. The promise of open source is better quality, higher reliability, more flexibility, lower cost, and an end to predatory vendor lock-in.

Key-Words: *Open-source software, freeware, toolbox/toolkit for research/education, simulation, computer science and engineering, education and training, innovation and issues, code ownership*

1. Introduction

Open source doesn't just mean access to the source code. The distribution terms of open-source software must comply with the following criteria:

I. Free Redistribution

The license shall 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 shall not require a royalty or other fee for such sale.

II. Source Code

The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost preferably, downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a pre-processor or translator are not allowed.

III. Derived Works

The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

IV. Integrity of the Author's Source Code

The license may restrict source-code from being distributed in modified form only if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.

V. No Discrimination Against Persons or Groups

The license must not discriminate against any person or group of persons.

VI. No Discrimination Against Fields of Endeavour

The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

VII. Distribution of License

The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

VIII. License Must Not Be Specific to a Product

The rights attached to the program must not depend on the program's being part of a particular software distribution. If 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 as those that are granted in conjunction with the original software distribution.

IX. License Must Not Restrict Other Software

The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.

X. License Must Be Technology-Neutral

No provision of the license may be predicated on any individual technology or style of interface.

Open source is a development method for software that harnesses the power of distributed peer review and transparency of process. The promise of open source is better quality, higher reliability, more flexibility, lower cost, and an end to predatory vendor lock-in. Starting in the early 2000s, a number of companies began to publish a portion of their source code to claim they were open source, while keeping key parts closed. This led to the development of the now widely used terms free open source software and commercial open source software to distinguish between truly open and hybrid forms of open source.

The Open Source Initiative Approved License trademark and program creates a nexus of trust around which developers, users, corporations and governments can organize open source cooperation. The open source movement has inspired increased transparency and liberty in other fields. The open-source concept has also been applied to media other than computer programs.

The promise of open source is better quality, higher reliability, more flexibility, lower cost, and an end to predatory vendor lock-in. Subsequently, the new phrase "open-source software" was born to describe the environment that the new copyright, licensing, domain, and consumer issues created.

Open source on the Internet began when the Internet was just a message board, and progressed to more advanced presentation and sharing forms like a Web site. There are now many Web sites, organizations and businesses that promote open source sharing of everything from computer code to mechanics of improving a product, technique, or medical advancement. The open source movement has inspired increased transparency and liberty in other fields.

Often, open source is an expression where it simply means that a system is available to all who wish to work on it. A main principle and practice of open source software development is peer production by bartering and collaboration, with the end-product, source-material, "blueprints," and documentation available at no cost to the public. This is increasingly being applied in other fields of endeavor, such as biotechnology.

Very similar to open standards, researchers with access to Advanced Research Projects Agency Network (ARPANET) used a process called Request for Comments (RFCs) to develop telecommunication network protocols. This collaborative process of the 1960s led to the birth of the Internet in 1969.

2. Architecture of OSS

In spite of the hype and hysteria surrounding open source software development, there is very little that can be said of open source in general. Open source projects range in scope from the miniscule, such as the thousands of non-maintained code dumps left behind at the end of class projects, dissertations, and failed commercial ventures, to the truly international, with thousands of developers collaborating, directly or indirectly, on a common platform. One characteristic that is shared by the largest and most successful open source projects, however, is a software architecture designed to promote anarchic collaboration through extensions while at the same time preserving centralized control over the interfaces. This talk features a survey of the state-of-the-practice in open source development in regards to software architecture, with particular emphasis on the modular extensibility interfaces within several of the most successful projects, including Apache httpd, Eclipse, Mozilla Firefox, Linux kernel, and the World Wide Web (which few people recognize as an open source project in itself). These projects fall under the general category of *collaborative open source software development*, which emphasizes community aspects of software engineering in order to compensate for the often-volunteer nature of core developers and take advantage of the scalability obtainable through Internet-based virtual organizations.

IJCSMS International Journal of Computer Science and Management Studies Volume 11, Issue 03, Oct 2011
ISSN (Online):2231-5268
www.ijcsms.com

Open Source software development, which is figured as nonlinear and self-organizing, from Closed Source, which is represented as hierarchical and authoritarian. The Open Source model has been characterized by some as representing a libratory politics for the information age.
An aspect of the software mentioned is that they are used by skilled users themselves – often developers. This is a fundamentally different model to the typical 'consumer' model which shoves a shrink-wrapped product down the 'luser's throat and expects them to pay for every upgrade, driven by features not stability. Of course, this improved model is a direct result of free software's for fundamental freedoms, and not merely because the source is accessible.

Flexibility

At the architectural level, experience shows that it is often best to pick tried and trusted standards for interworking. If that is done, then best-of breed solutions can be selected for particular components within the architecture. Provided that the solutions can interwork suitably, the business should be able to avoid lock-in to a particular supplier and over-dependency. This is notoriously hard to manage, requiring a real act of will from management. What happens most often is that a vendor will make a `feature sale', emphasizing something which cannot be done through the standard infrastructure. If they succeed then the business can become dependent on that particular solution and unable to choose alternatives at a later date. Any astute vendor will attempt to do this, only vigilant managers can avoid the lock-in that follows. Proprietary data formats are a particularly good tool for vendors to use. If they can establish a bridgehead, their competition will not only have to provide competing functionality, but also data conversion tools from a (typically) undocumented or even protected format.

3. Impact Factors on OSS

THE FACTORS THAT IMPACT ON OSS

The success of OSS has been mostly attributed the reliability, portability and scalability of the resulting software [1-6]. In turn, these qualities are attributed to three main issues, namely the fact that developers are usually also users of the software, the public availability of the source code, and the fact that developers are members of a community of developers.

Personal Need

Open Source Software often originates from a personal need [5, 6]. This approach to software offers some real benefits in the design process. Since developers are users of the software, they understand the requirements in a deep way. As a result, the ambiguity that often characterizes the identification of user needs or requests for improvement in the traditional software development process is eliminated: programmers know their own needs [7].

Open Inspection and Contributions

The personal needs attract the attention of other user-developers and inspire them to contribute. In OSS, the source code is open to inspection by and contributions from any interested individual. Therefore, users can also be developers. If they find bugs, they can fix them themselves rather than having to wait for the developers to do so; if a specialized feature is needed, it can be added, even if it is not one that the developers feel is cost-justified. As a result, OSS bugs can be fixed and features evolved more quickly.

Developers as a Community Part

Developers are part of a community. The OSS community represents a nexus of exchanges in which people report bugs expecting that other members will fix them. Similarly those who fix bugs expect other developers to contribute to other parts of the project [8]. Reputation is another important aspect — the community is in fact frequently described as being based on peer recognition and in some cases on a "cult of the personality". In particular, peer recognition is a value for the community that can sometimes lead to employment opportunities or access to venture capital [9]. In such an environment, developers may be motivated to do the best work they can, rather than anonymously finishing code so it can be shipped.

Commercial support

OSS with commercial support is relatively small. Luckily, there are lots of free resources out there: mailing lists, forums, wikis, and IRC (internet relay chat) channels, to name a few. The support available through these free resources is comparable to, and sometimes even better than, traditional commercial support.

Hardware compatibility

Hardware compatibility is another crucial factor when choosing a FOSS operating system. The system has to be capable of supporting your computer's parts and the types of devices in use. If you have a critically important computer part or device, it's often simplest just to check with the hardware maker for advice about which FOSS operating systems are supported.

Software compatibility

Software compatibility will likely be an important issue if you plan to use commercial software. As a general rule, most FOSS software will work with most FOSS operating systems. If there are particular programs you know in advance you'll need, then you should verify that they'll work with the operating system of your choice. One issue that may be overlooked is which version of any particular software application is installed on the operating system "out of the box". Server FOSS operating systems tend to come with older versions of applications, so if you prefer a more recent version of a particular application you might first have to uninstall the older version.

The main implication of the three characteristics described above is that OSS software engineering processes have evolved to develop software that meets developers' needs [10]. On the other hand, OSS, with its reliance on self-interested developers, may be less well suited for developing applications that address problems that developers tend not to face. We see very good OSS tools for software development and good end-user tools for issues faced by developers (e.g., email, word processing), for example, but would expect to see few OSS applications for problems developers rarely face (e.g., accounting, textual analysis).

4. Reliabilities of OSS Premises

Reliability mean the absence of defects which cause incorrect operation, data loss or sudden failures, perhaps what many people would mean when they use the term `bug'. it's hard to point to that as good way of defining what is a bug and what is a feature. Determining what constitutes a bug is usually by agreement amongst the developers and users of the software (an overlapping community in many cases). Obvious failure to perform is easily recognized as a bug, as is failure to conform to appropriate published standards.

Security related failings (exploits or vulnerabilities) are clearly bugs too. Each of these kinds of bugs is usually addressed with speedy fixes wherever possible and Open Source advocates will claim very rapid time-to-fix characteristics for software. White-box and black-box models are two approaches for predication of software reliability.

The white-box models attempt to measure the quality of a software system based on its structure that is normally architected during the specification and design of the product. Relationship of Software components and their correlation are thus the focus for software reliability measurement [1], [5], [22], [23]. In the black-box approach, the entire software system is treated as a single entity, thus ignoring software structures and components interdependencies. These models tend to measure and predict software quality in the later phases of software development, such as testing or operation phase.

The models rely on the testing data collected over an observed time period. Some popular examples are: Yamada S-Shape, Littlewood-Verrall, Jelinski-Moranda, MusaOkumoto, and Goel-Okumoto.

This study is concentrated on the black-box reliability approach to measure and compare the reliability of the selected OSS projects. Users of the software can choose whether to use the unofficial fix or wait for an `official' version. By `official' we mean a release blessed by the software team itself or a trusted authority such as one of the main distributors of Open Source packages. This mechanism clearly works very well in practice. Consequently much Open Source software becomes highly robust at a surprisingly early stage of its development, and mature Open Source products are setting new industry standards for bulletproofness.Figure 2 A portion of a bug report in XML at Bugzilla.

A. Black-Box Reliability Analysis

a) Bug-Gathering

A Java program portion of a bug report in XML stored at Bugzilla is developed to gather the relevant data from the XML format for further data filtering and analysis.

b) Bug-Filtering

The duration for which the failure data is collected for the five OSS projects are listed in Table I

	Project name	Start date	End date
1	Firefox	3/1999	10/2006
2	Eclipse	10/2001	12/2007
3	Apache 2	03/2002	12/2008
4	ClamWin Free Antivirus	03/2004	08/2008
5	MPlayer	09/2002	06/2006

IJCSMS International Journal of Computer Science and Management Studies Volume 11, Issue 03, Oct 2011
ISSN (Online):2231-5268
www.ijcsms.com

```
<bug>

 <bug_id>366101</bug_id>

      <creation_ts>2007-01-05 16:41
PST</creation_ts>

 <short_desc>nsIFile.initWithPath should
accept &amp;quot;c:/mozilla&amp;quot; as native path
(forward slashes should be treated as
backslashes)

</short_desc>

      <delta_ts>2007-01-05 16:57:22
PST</delta_ts>


<reporter_accessible>1</reporter_accessible>


<cclist_accessible>1</cclist_accessible>


<classification_id>3</classification_id>


<classification>Components</classification>

      <product>Core</product>

      <component>XPCOM</component>

      <version>Trunk</version>

      <rep_platform>PC</rep_platform>

      <op_sys>Windows XP</op_sys>

      <bug_status>NEW</bug_status>

      <priority>--</priority>

      <bug_severity>normal</bug_severity>

 …

      <who name="David
Hyatt">hyatt@mozilla.org</who>

      <bug_when>2000-04-13 16:16:07
PST</bug_when>

 …

      <bug_status>VERIFIED</bug_status>


<resolution>WORKSFORME</resolution>

</bug>
```

c) Bug-Analysis.

In the bug-analysis step, the frequency of bugs in two week periods is calculated. Therefore, the x-axis and y-axis represent the biweekly time and the corresponding bug frequency, respectively.The critical testing factors that determines the reliabilities of open source software premises:

1. System Environment
2. Emulator and Devices
3. Application Complexity

B. Quality Factors of OSS

Some interesting facts about open source quality, and in particular mentioned that open source software has an average defect density that is 50-150 times lower than proprietary software. As it stands, this statement is somewhat incorrect, and I would like to provide a small clarification of the context and the real values:

- Average: That is mentioned by Michael is related to a small number of projects, in particular the Linux kernel, the Apache web server (and later the entire LAMP stack), and a small number of additional, “famous” projects. For all of these projects, the reality is that the defect density is substantially lower than that of comparable proprietary products. [4]
- Other than the software engineering community, some results from companies working in the code defect identification industry also published some results, like Reasoning Inc. *A Quantitative Analysis of TCP/IP Implementations in Commercial Software and in the Linux Kernel*, and *How Open Source and Commercial Software Compare: Database Implementations in Commercial Software and in MySQL*. All results confirm the much higher quality (in terms of defect per line of code) of the academic research.
- Additional research identified a common pattern: the *initial* quality of the source code is roughly the same for proprietary and open source, but the defect density decreases in a much faster way with open source. So, it’s not the fact that OSS coders are on average code wonders, but that the process itself creates more opportunity for defect resolution on average. As Succi et al. pointed out: “*In terms of defects, our analysis finds that the changing rate or the functions modified as a percentage of the total functions is higher in open-source projects than in closed- source projects. This supports the hypothesis that defects may be found and fixed more quickly in open-source projects than in closed-source*

projects and may be an added benefit for using the open-source development model." (Emphasis Mine).

- Code Reusability: The general modularity and great reuse of components are helping developers, because instead of recoding something (introducing new bugs) the reuse of an already debugged component reduces the overall defect density. This aspect was found in other research groups focusing on reuse.
- As it can be observed from the graph, code originated from reuse has a significant higher quality compared to traditional code, and the gap between the two grows with the size (as expected from basic probabilistic models of defect generation and discovery).
- The second aspect is that the fact that bug data is public allows a "prioritization" and a better coordination of developers on triaging and in general fixing things. This explains why this faster improvement appears not only in code that is reused, but in newly generated code as well; the sum of the two effects explains the incredible difference in quality (50-150 times), higher than any previous effort like formal methods, automated code generation and so on. And this quality differential can only grow with time, leading to a long-term push for proprietary vendor to include more and more open source code inside of their own products to reduce the growing effort of bug isolation and fixing.

C. Economic Analysis

Open Source Software are available free of royalties and fees, leading to the confusion around the commonly used term `free software'. Regrettably the English language does not have separate concepts for free-of-charge and free as in unconstrained; other languages are better equipped to describe the difference between `freedom' and `free of charge' (libre vs.gratis). Proponents of free software licences tend to emphasise liberty over cost although in practice the main open source projects are free in both senses of the word.

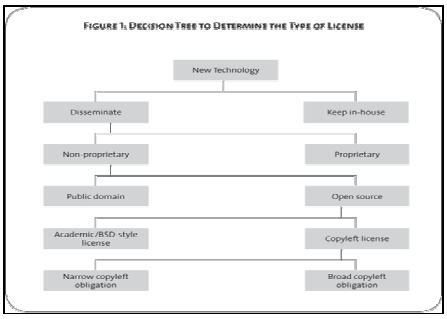

FIGURE 1: DECISION TREE TO DETERMINE THE TYPE OF LICENSE

From a business perspective the purchase cost of software is only one factor; total cost of ownership (TCO) is what really matters. Other things being equal, the solution with lowest TCO is usually the most desirable one. Arguments in favour of low TCO for open source software include:

- Possibly zero purchase price
- Potentially no need to account for copies in use, reducing administrative overhead
- Claimed reduced need for regular upgrades (giving lower/nil upgrade fees, lower management costs)
- Claimed longer uptimes and reduced need for expensive systems administrators
- Near-zero vulnerability to viruses eliminating need for virus checking, data loss and downtime
- Claimed lower vulnerability to security breaches and hack attacks reducing systems administration load
- Claimed ability to prolong life of older hardware while retaining performance

Some longer-term claims are more difficult to substantiate yet they need to be taken into account:

- Better adherence to standards permits competition in the market, reducing vendor lock-in and consequent monopoly pricing
- Availability of source code provides greater continuity and security against
- Financial collapse of vendors of key products
- Vendors choosing to withdraw support for unprofitable products
- Protection against being required to fit your IT strategy to the cash needs of your software supplier

Unfortunately in this area there are numerous claims and counter claims. Reliable TCO information is practically unobtainable, although the case studies which form part of this guide provide a large amount of circumstantial evidence in favour of the argument. Most businesses will have to choose the argument on its merits and choose to back the use of Open Source software where it seems most likely to provide either a clear cost win, or valuable leverage over entrenched suppliers.

Access costs also pose problems for authors who wish to create something based on another work but are not willing to pay the copyright holder for the rights to the copyrighted work. The second type of cost incurred with a copyright system is the cost of administration and enforcement of the copyright. These self-made protections free the general society of the costs of policing copyright infringement. Thus, on several fronts, there is an efficiency argument to be made on behalf of open sourced goods.

5. Comparison with free software

IJCSMS International Journal of Computer Science and Management Studies Volume 11, Issue 03, Oct 2011
ISSN (Online):2231-5268
www.ijcsms.com

The main difference is that by choosing one term over the other (i.e. either "open source" or "free software") one lets others know about what one's goals are. As Richard Stallman puts it, "Open source is a development methodology; free software is a social movement."

Critics have said that the term "open source" fosters an ambiguity of a different kind such that it confuses the mere availability of the source with the freedom to use, modify, and redistribute it. Developers have used the alternative terms *Free/open source Software* (FOSS), or *Free/Libre/open source Software* (FLOSS), consequently, to describe open source software which is also free software.

The term "open source" was originally intended to be trademarkable; however, the term was deemed too descriptive, so no trademark exists. The OSI would prefer that people treat Open Source as if it were a trademark, and use it only to describe software licensed under an OSI approved license.

OSI Certified is a trademark licensed only to people who are distributing software licensed under a license listed on the Open Source Initiative's list.

Open source software and free software are different terms for software which comes with certain rights, or freedoms, for the user. They describe two approaches and philosophies towards free software.*Open source* and *free software* (or *software libre*) both describe software which is free from onerous licensing restrictions. It may be used, copied, studied, modified and redistributed without restriction. Free software is not the same as freeware, software available at zero price.

The definition of open source software was written to be almost identical to the free software definition. There are very few cases of software that is free software but is not open source software, and vice versa. The difference in the terms is where they place the emphasis. "Free software" is defined in terms of giving the user freedom. This reflects the goal of the free software movement. "Open source" highlights that the source code is viewable to all; proponents of the term usually emphasize the quality of the software and how this is caused by the development models which are possible and popular among free and open source software projects.

Free software licenses are not written exclusively by the FSF. The FSF and the OSI both list licenses which meet their respective definitions of free software or open source software.

The FSF believes that knowledge of the concept of freedom is an essential requirement, insists on the use of the term *free*, and separates itself from the open source movement.

Limitations

Limitation as persistent open source software

Anyone with the skills can view the code and contribute to it. It is highly flexible due to source code access (third parties can customize it completely) and the requirements of a development model wherein the atomic contributions (as in small, not nuclear) of thousands of developers are organized within a single product.

There is a lack of concrete incentive to motivate developers to contribute to open source projects and the real problem, however, is that open source must rely on the willingness of programmers to contribute code without financial compensation. So after a time, the developer loses his interest and concentration in developing code for software.

Fulfilling the Promises of OSS

Open source sharing of information in virtual globes provide a means to identify economically and environmentally beneficial opportunities for waste management if the data have been made available.

1. Reduce embodied transport energy by reducing distances to recycling facilities.
2. Choose end of life at recycling facilities rather than landfills.
3. Establish industrial symbiosis and eco-industrial parks on known by-product synergies.

The Distribution Terms

Open source doesn't just mean access to the source code. The distribution terms of open-source software must comply with the following criteria:

Free Redistribution

The license shall 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 shall not require a royalty or other fee for such sale.

Source Code

The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed. The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost preferably, downloading via the Internet without charge.

Derived Works

The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

Integrity of the Author's Source Code

The license may restrict source-code from being distributed in modified form *only* if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.

No Discrimination against Persons or Groups

Copyright restriction then creates access costs on consumers who value the original more than making an additional copy but value the original less than its price. Thus, they will pay an access cost of this difference. Access costs also pose problems for authors who wish to create something based on another work but are not willing to pay the copyright holder for the rights to the copyrighted work. The second type of cost incurred with a copyright system is the cost of administration and enforcement of the copyright. The license must not discriminate against any person or group of persons.

License Must Not Be Specific to a Product

The rights attached to the program must not depend on the program's being part of a particular software distribution. If 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 as those that are granted in conjunction with the original software distribution.

Advantages of Open Source Software

Many people like Open Source for many reasons, here is an overview of some of the more important reasons. You can read through these if you're not sure you want to try Open Source yet, or you can continue to the pages describing actual programs you can use.

- Security: Open Source Software suffers from fewer security vulnerabilities than Microsoft products.
- Features: Open Source programs tend to have more advanced features and customizability than proprietary products
- Cost: Open Source Software is FREE. You pay nothing for a very high quality product.
- Community: In the Open Source development community, any skilled individual can contribute to projects in many ways.

6. Conclusion

The Open Source Initiative (OSI) is a non-profit corporation with global scope formed to educate about and advocate for the benefits of open source and to build bridges among different constituencies in the open source community.
One of our most important activities is as a standards body, maintaining the Open Source Definition for the good of the community. The Open Source Initiative Approved License trademark and program creates a nexus of trust around which developers, users, corporations and governments can organize open source cooperation.

References

[1] R.C. Cheung, “A user-oriented software reliability model”, IEEE Transactions on Software Engineering, vol. 6, no. 2, March 1980, pp. 118-125.
[2] en.wikipedia.org/wiki/Open-source software
[3] Apache Group, 1999. Apache FAQ http://httpd.apache.org/docs/misc/FAQ.html#support
[4] Succi, Paulson, Eberlein. *An Empirical Study of Open-Source and Closed-Source Software Products*, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, V.30/4, april 2004.
[5] Moody, G. Rebel code—Inside Linux and the open source movement. Perseus Publishing, Cambridge, MA, 2001.
[6] Vixie, P. Software engineering, in Open sources: Voices from the open source revolution, C. Di Bona, S. Ockman, and M. Stone, O’Reilly, Eds. San Francisco, 1999.
[7] Kraut, R. E., and Streeter L. A. Coordination in software development. Communications of the ACM, 38 (1995), 69–81.
[8] Moon, J. Y., and Sproull L. Essence of distributed work: The case of Linux kernel. First Monday, 5, (2000).
[9] Markus, M. L., Manville, B., and Agres, E. C. What makes a virtual organization work?. Sloan Management Review, 42 (2000), 13–26.
[10] Ousterhout, J., Free Software needs profit. Communications of the ACM, 42 (1999), 44–45.
[11] Mohagheghi, Conradi, Killi and Schwarz called “An Empirical Study of Software Reuse vs. Defect-Density and Stability”