

The Organization of Open Source Communities: Towards a Framework to Analyze the Relationship between Openness and Reliability

Ruben van Wendel de Joode & Mark de Bruijne
Delft, University of Technology
rubenw@tbm.tudelft.nl & markbr@tbm.tudelft.nl

Abstract

A number of open source communities have been able to create surprisingly reliable software. The popular claims to explain how and why certain open source packages have managed to become reliable are primarily focused on the openness of the communities and the development process. This paper describes our ongoing efforts to build a framework and define a number of propositions to guide our research effort in trying to understand the relationship between openness and reliability. Using an organizational focus on the issue of openness, we combine empirical evidence gained from research in a small-scale open source community (MMBase) with findings from two organizational theories that focus on the reliability of complex, large-scale technological systems.

In this paper we introduce three propositions, which are: i) the bigger the percentage of developers in an open source community who actually use the software, the more reliable the software; ii) the more transparent the flow of information in an open source community, the more reliable the software; iii) the more popular the open source software, the more reliable the software.

1 Introduction

A number of open source communities have been able to create surprisingly reliable software. Research shows that Apache for instance has 31 software defects in 58,944 lines of source code.¹ This results in a defect density of 0.53 per 1,000 lines of source code, which is said to be comparable to proprietary developed software programs, which have an average defect density of 0.51. In another research six operating systems were compared on their implementation of a key-networking component. It concluded that the Linux kernel performed better than the five proprietary developed operating systems. The study also showed that the networking component of the operating system had “8 defects in 81,852 lines”², which results in a defect density of 0.078 per 1,000 lines of source code. What can explain why software created in open source communities can become reliable? Much of the semi-scientific and popular writings debate this question and it is riddled with much rhetoric and story-

telling.

The popular claims to explain how and why certain open source packages have managed to become reliable is primarily focused on the *openness* of the communities and the development process.³ The following statement illustrates this point perfectly: “[w]ith enough eyeballs, all bugs are shallow.”⁴ This Linux axiom points to the fact that when a bug becomes an issue, many people have the source code, and it can be quickly resolved without the help of a vendor.⁴ Thus, one of the most popular claims to explain why some open source communities are able to create reliable software is based on the openness of the communities. Openness of the communities allows users who experience a bug in the software to locate that bug and to fix it accordingly.

On the other hand opponents of open source software, claim that sufficient ‘eyeballs’ will not automatically result in reliable software. On the contrary opponents of open source communities claim that openness introduces a whole new set of reliability issues: “The vulnerabilities are there. The fact that somebody in the middle of the night in China who you don’t know, quote, ‘patched’ it and you don’t know the quality of that, I mean, there’s nothing per se that says that there should be integrity that comes out of that process.”⁵ Although the argument is riddled with rhetoric and highly biased the point that is made is a valid one: openness alone cannot provide a sufficient answer to the question how open source communities are able to create reliable software.

The next section introduces an explorative case study and demonstrates that, although according to members of an open source community, openness plays a role, it is certainly not sufficient.

In this paper we want to explore the nature of the relationship between the openness of open source communities and the reliability of the product that results from this organizational form. The reason for describing it as an organizational form is because both the proponents and the opponents of open source software attribute the explanation to the organization of the communities. Both claim that the organizational structure and the way in which people are organized explain why open source software is inherently reliable or inherently unreliable.

¹ Based on:
http://www.infoworld.com/article/03/07/01/HNreasoning_1.html (March 2004).

² Based on: http://www.reasoning.com/newsevents/pr/02_11_03.html (August 2004).

³ In another part in this paper we will define what we mean with openness in more detail.

⁴ From an article on the Internet:
<http://techupdate.zdnet.com/techupdate/stories/main/0,14179,2907749,00.html> (last visited October 2004).

⁵ This quote is from an article on the Internet:
<http://www.microsoft.com/presspass/exec/steve/2003/10-21Gartner.asp> (last visited July 2004).

This leads to the research question that reads as follows: *How does openness of open source communities contribute to the production of reliable software?*

The questions: ‘is open source software reliable?’ and ‘is open source software more or less reliable than proprietary software?’ are relevant, yet they will not be answered in this paper. In this paper we want to focus on a different question. In our opinion it is very surprising that open source software is reliable, since we know that literature on software reliability tells us that creating highly reliable software is very difficult, and increasingly so [e.g. 3, 10, 13]. Arguably, judged on the debates, there are features in the organization of open source communities that allow them to create software that has reached a certain level of reliability.

The goal of this paper is not to give a definitive answer to the research question. Our current progress in this research does not allow us to do so. Instead we propose a framework and define a number of propositions to guide our future research efforts in trying to understand the relationship between openness and reliability.

First, we introduce a short case study that describes the MMBase community. After providing an introduction of the community, we describe the outcome of our analysis on the relationship between openness and reliability in the MMBase community.

Next we introduce two theories in which organizational conditions are identified to help structure the analysis on the relationship between openness and reliability. We specifically introduce a number of propositions that we want to verify in future case studies.

2 Analyzing the influence of openness on reliability: a first case study

We have undertaken a first case study to study the relationship of openness and the reliability of open source software. Although the focus of the case study, which studied the MMBase community, was not solely geared to understand the relationship between openness and the reliability of software, some of the questions were intended to get a better understanding of the potential relationship.

A main reason for analyzing MMBase can be found in the size of the community. MMBase is a relatively small community. The primary reason for choosing a small community is that we believe that openness does contribute to the reliability of software, but only because many people actually have access to the source code. In that sense we follow Raymond’s proposition that with many eyeballs, all bugs are shallow [23].

2.1 The MMBase community

The MMBase software was initially created as proprietary software in the 1990’s when a Dutch public broadcast corporation created a web Content Management System (CMS). In 1999, the corporation decided to

convert the CMS into open source software. There were two reasons to do so. First, the system had been created with public funds, which should in some way be returned to the public. Second, in the course of time the public broadcaster increasingly became dependent on the system. However, being the sole organization to use and maintain the software, it increasingly required dedicated know-how and costs. By opening up the software the corporation hoped to attract other users who could participate and subsequently share in the development and maintenance (costs) of the CMS.

When the MMBase community was established, the group of users initially consisted of the broadcast corporation that had developed the CMS software and a small number of other Dutch public broadcast companies. The first MMBase developers were hired by the broadcasters to exchange knowledge and to improve the software in the newly erected MMBase community.

The following years, the community enhanced the original CMS creating a far more robust piece of software, which gradually accumulated features that could be used outside the media-environment of the public broadcasts. In 2001 the community decided that the time had come to create the MMBase Management Committee. The committee would be responsible for the coordination of the future technological development of the system. Most of the committee members came from the group of developers who worked for the public broadcast companies.

In 2002 the community of users, which was still relatively small, decided that something needed to change to attract new users. According to the community, the focus on technology development alone had proven insufficient. Consequently, the community decided to create a foundation that could fulfill the roles of coordination, marketing, responding to requests for information (RFIs), setting up collaboration among users and facilitating knowledge exchange. This, it was hoped, would increase the user-base of the MMBase software.

The first action of the foundation was to appoint a director, which would be in charge of all these activities. At the time of writing, in June 2005, the MMBase community has been able to attract a fair amount of new users, of which the city of Amsterdam and mobile telephone multinational Vodafone are probably among the best-known [1]. Furthermore, MMBase has expanded its development base as software companies like IBM and Ordina have offered their support for MMBase.

2.2 How does openness of the MMBase community result in reliable software?

One of the primary focuses in the case study was to get a first understanding of the bug fixing process in an open source community, since this is an essential part of reliability in open source communities as witnessed by the quote from Raymond (...all bugs are shallow). Essentially we focused on two tasks in the bug fixing process: namely bug reporting and bug fixing. See [4] for a more detailed description of the tasks that are part of the bug fixing process in OSS communities.

In the case study we asked 12 people from the MMBase community – both developers and users – to react to the following proposition: “In MMBase vital bugs are resolved because many people have access to the source code.” Out of the 12 respondents 10 agreed to the proposition. They agreed to the fact that because people have access to the source code bugs are indeed solved.

However, when asked to explain their answer, 4 respondents stated that they were not sure whether the actual reason was because so many people have access to the source code. They claimed that only a few people actually take a look at the source code. Most users do not analyze the source code or try to report and solve bugs.

Another 4 respondents explained that openness of the communities might play a role, but it does not ensure that the software is reliable. They gave a very simple reason: in the MMBase community only one person does most of the fixing. One of these 4 respondents described it as follows: “half of the bugs are resolved by only one person.”

Further analysis of the bug tracking system that is used in the MMBase community has taught us that out of the 98 people who have an account in the MMBase bug-tracking system 59 people have reported zero or only one bug. In total 1158 bugs were reported. A staggering 71% of these bugs were reported by only 4 people. The 10 most active bug reporters are responsible for 85% of all bug reports.

One might be tempted to think that this is an exception and might only apply to the smaller open source communities like MMBase. However, research [15] reports similar findings. They found that in the Apache community only 15 bug reporters were responsible for reporting 83% of all bugs.

This provides some evidence that the common argument, which says that bugs are reported and solved because many people use their ability to access the software, to change the source code and solve the bugs, is not sufficient. Instead, other explanations may provide ideas how and why bugs are reported and resolved and why at least in some open source communities such high levels of reliability are achieved.

The next section introduces two theories on reliability to formulate a number of propositions as to how openness influences reliability in communities.

3 A theoretical perspective on openness and reliability

The take on openness and reliability will be based on two organizational theories, which are interested in the reliability, or lack thereof, of complex, large-scale technological systems. The first theory was developed to explain how major accidents could occur in large-scale complex systems and why accidents are in fact inescapable in certain technologies. The second theory, tries to explain how certain organizations apparently succeed in extracting themselves from the predictions about failures that the first theory makes. Both theories do not directly address the issue of “openness” with regard to

issues of safety or reliability. However, some of the conditions that these theories find to contribute to failure-proneness seem to touch upon issues of openness.

Before introducing both theories, we define openness and reliability in more detail. We finish this part with a discussion of the applicability of both theories on open source communities.

3.1 Defining reliability and openness

Reliability is often considered an important aspect of the much broader and more subjective concept of quality, and can be defined as:

[t]he probability that an item will perform a required function without failure under stated conditions for a stated period of time. [11, p. 72]

Thus, according to this definition the reliability of an item is based on the probability of a failure. Such outcome parameters, i.e. the number of failures that occur, don’t provide sufficient information about the real state of the system and thus the potential for failures.

The difficulty about reliability is that reliability may be described as a ‘dynamic non-event’ [34, p. 118]. Reliability is dynamic in the sense that it is an ongoing condition. Reliability indicators however are static and are only able to indicate that a system is *momentarily* under control. Reliability is something that necessitates constant attention, and to assure that a system is reliable necessitates continuous feedback.

On the other hand, reliability is a non-event because the outcome, i.e. the reliable provision of services, is virtually constant [34, p. 118]. Due to the very success of technology development, reliability is a ‘hidden characteristic’. An attribute that is only noticeable when it is absent, when the provision of services is interrupted or disrupted.

Consequently, we lack a single index or value to describe or define reliability, let alone to predict the future reliability of software. To nevertheless be able to make any judgements on reliability, a technology-independent performance indicator of reliability is used.

In this paper, considering the ability to individually report and change bugs as an essential aspect of openness, we use processes involving the reporting and fixing of bugs as primary measure of reliability. More concrete we propose a combination of the following indicators: (i) the number of defects per (1,000) line(s) of code, (ii) the number of reported bugs, and (iii) the speed with which the reported bugs are fixed.

The term ‘*openness*’ is logically connected to the name ‘open source communities’. But what exactly do we mean with openness? [2] writes:

What most distinguishes free software from off-the-shelf proprietary software is the openness of the source code, and thus the user’s freedom to use and distribute the software in whatever ways desired. Anyone with the expertise can “look under the hood” of the software and

modify the engine, change the carburetor or install turbochargers.

Bollier focuses on one aspect of open source, which is that anyone can have access to the source code of open source software. This is definitely an important aspect of openness, yet it is not the only one we can think of when using the term. Openness also refers to the developers in the communities who freely share their ideas and inventions amongst each other [9]. Furthermore, it refers to the fact that anyone can have a look at what others and, especially relevant in the context of this paper, developers in open source communities are doing. There are no artificially constructed 'walls' to keep outsiders from looking in.

3.2 Normal Accident Theory on openness

The so-called Normal Accident Theory (NAT) argues that large technical systems have characteristics that make them inherently prone to failure: they generate complex, unexpected interactions among system components and their components are tightly coupled [e.g. 30, 17, 18].

A number of so-called 'error-reducing characteristics' [18, p. 218; 21], further influence the frequency, spread and ability to manage failures and consequently the catastrophe rate within systems [19, p. 12]. One of them may be considered particularly important with regard to issues of openness and reliability.

NAT argues that there exists a correlation between the ability and willingness of organizations to avoid systemic failures and the eventual impact of system accidents. The amount of effort that is put into the maintenance of reliability in large-scale systems may be related to the extent that *elites* are affected by them. If elites are not directly affected by the consequences of unreliability, then chances are higher that the organizations managing these systems are not sufficiently encouraged to maintain high levels of safety or reliability [18]. Simply put: "few managers are punished for not putting safety first even after an accident, but will quickly be punished for not putting profits, market share, or agency prestige first" [20, p. 370]. Thus, according to NAT, failures in systems may often be described as "alarmingly banal examples of organizational elites not trying very hard at all" [18, p. 218].

Building upon Perrow's work, [30] research on nuclear weapons operations in military organizations goes one step further in tracing the origins of failures in large-scale and complex systems. Sagan argues that reliability or safety in systems may not be solely affected as a result of political elites who simply do not care, but because they are misinformed on the risks that are involved in employing large-scale technical systems. According to Sagan, the risk of misinformation increases as organizations are "closed".

Whereas Perrow argues that "closed" organizations with strong organizational control over members can potentially reduce errors with hazardous technologies as a result of their ability to intensively train those who deal with these technologies, Sagan argues that it creates

strong tensions with efforts to promote openness, an attribute that is considered essential to stimulate learning. Openness allows organizations to learn from mistakes. However, efforts to open up the organization conflict with tight organizational control over its members. Tight organizational control over members leads to efforts to protect the parochial interests and the reputation of the organization at the expense of exposing problems related to issues of reliability or safety. Under these circumstances, organizations are less open to admit or address potential failures or reliability issues as these may affect the goals of the organization.

In his historical study on nuclear weapons operations, Sagan finds that "the military's organizational power over information and interpretation led to a much more accident-prone nuclear system than American civilian leaders desired or understood existed" [31, p. 237].

3.3 High-Reliability Theory on openness

While recognizing the importance of NAT to consider certain large-scale technical systems as inherently accident-prone, a group of researchers pointed out that some of these systems nevertheless achieved remarkable levels of reliability, even though much seemed to conspire against them. The researchers were puzzled by organizations, which they labeled 'High Reliability Organizations' (HRO's), which could not be explained using conventional organizational theory [e.g. 25; 26; 27, 28].⁶ Like open source communities, HRO's seemed to defy predictions and conventional assumptions with regard to the reliability of their performance. Organizations that were chosen as the object of study included (nuclear) aircraft carrier flight operations and nuclear power plants [e.g. 26, 27, 32] and more recently also organizations that operate in commercial environments under "conditions such as increased competition, higher customer expectations, and reduced cycle time [that] create unforgiving conditions with high performance standards and little tolerance for errors" [33, 35]. This research evolved into what we now have come to know as High Reliability Theory (HRT).

HRT argued that HRO's have nurtured a number of conditions that result in organizations where people who work inside them manage complex systems remarkably well and continuously maintain high levels of reliability [29]. Among the factors commonly identified as reliability enhancing [10], HRT considers, a *strong presence of external groups* with access to credible and timely operational information. To maintain the focus upon the goals of the high reliability organization usually requires the presence and oversight of some external groups.

[12] specifically mentions the importance of independent public bodies, stake-holding interest groups and professional peer bodies, which would maintain the focus of the HRO on its reliability goals (p. 65). According to [12] the chances of HRO's maintaining or even enhancing their performance will be facilitated by

⁶ HRT is usually offset against NAT [e.g., 22, 24].

the “aggressive and knowledgeable oversight” of these groups. However, in order for external groups who have an oversight function to have any effect at all, these groups need to have the necessary and relevant information. And for this, the importance of accurate and timely information coming from the HRO’s they oversee is necessary.

3.4 Some thoughts about applying NAT and HRT to open source communities

At face value open source communities do not seem to compare well with the organizations that were studied in NAT and HRT. Consequently, a number of objections can be raised against applying these theories to open source communities.⁷ And indeed, we agree that, quite generally speaking, open source communities possess characteristics which are quite distinct from the more traditional type of organizations that were researched in NAT and HRT. Nevertheless, we feel there are a number of arguments to justify the application of these theories in the context of open source communities.

Firstly, HRT and NAT are the only theories that study reliability as a product of not only technological conditions but also organizational conditions. This makes HRT and NAT particularly relevant for our research, since (i) popular claims attribute reliability in open source to organizational characteristics and (ii) we are interested in the influence of openness on reliability, whereby we have defined openness primarily as an organizational characteristic of the communities.

Secondly, the application of HRT and NAT has not been limited to closed organizations such as military organizations. Instead, they have also been used to describe civil organizations such as private electric grid operators and air traffic control organizations, which offer more individual freedom to its organizational members than a military organization would [27, 32, 35]. Furthermore, both theories are increasingly used to study networks of organizations, including virtual organizations [6].

Thirdly, research on open source communities argues that open source communities do have organizational characteristics. For instance, [14] describes how the attachment of individuals to certain artifacts connects them to a particular community and gives them a feeling of belonging to that community. Examples are T-shirts and mascots, like the Linux penguin and the FreeBSD devil. [5] also argues that open source communities have boundaries to decide who is an ‘insider’ and who is an ‘outsider.’ These boundaries are informal and consist of, for instance, (i) the level of knowledge that potential participants need to understand the software and to be able to contribute to the development effort and (ii) the mastery of shared norms of conduct, as individuals who do not behave according to the norms will be sanctioned. The research like [5, 14] demonstrate that the

communities have organizational characteristics. In that sense we might find that the communities differ less from the more traditional organizations than we might suspect at face value.

4 Hypotheses on openness and reliability in open source communities

Based on organizational literature and on previously conducted extensive case study research [e.g. 36, 37], we propose a number of hypotheses to explain the relationship between openness and the ability of open source communities to create reliable software. Each of the hypotheses and their expected relationship with the organization of open source communities will be explored.

4.1 Hypothesis 1: The distinction between producers and consumers

NAT offers two potential explanations why failures occur in large-scale systems. First of all, NAT claims that the importance that organizational elites accord to safety or reliability depends on the measure in which they will be exposed to the consequences of potential failures. In other words, if they suffer from the consequences of failures, organizational elites (and subsequently organizations) will try harder to ensure the products are safe or reliable. The basic argument underlying Perrow’s claim is that organizational elites have the power to allocate resources to ensure the reliable and safe production of goods over other values such as profits or margins.

It is difficult to translate the concept of elites to open source communities. Who are the elites in the communities? Are they the project leaders of a community? Yet, irrespective whether or not it is difficult to translate the concept of elites, there is one thing in the description of social elites that appears to be relevant also for open source communities.

One of the most striking characteristics in open source communities is probably that many of the producers of the software are also its users. For instance, research on motivation asking the question: ‘why do people participate in open source communities?’, demonstrates that one of the most important reasons is to solve a personal need [7, 8]. In other words, people become involved in open source communities to make sure that the software they will use works according to their specific needs. This characteristic coincides with the presence of social elites in NAT theory.

Hypothesis 1: the bigger the percentage of developers in an open source community who actually use the software themselves, the more reliable the software.

⁷ We thank one of the anonymous reviewers for this valuable point of critique on a previous version of the paper.

4.2 Hypothesis 2: Control of information

The second claim in NAT concerning social elites is that the “closedness” of organizations that are involved in the production of a technology greatly contributes to the ability to be receptive to criticism and to learn from mistakes. The more the flow of information is controlled by the organization; the lower the chance that the technology will be reliable.

There are a number of indications that suggest that open source communities are very open and have a hard time in controlling the flow of information. Consider this statement: “...in the open source community monitoring the behavior of users is easy because the Internet gives full transparency...” [16, p. 15]. Furthermore, one respondent explained to us: “Online, a record is kept of who did what and when. Once you made a big mistake then it will become a ‘Hall of blame.’” To prevent this from happening, respondents described how they make sure to check every piece of source code before they add it to the repository.

Yet, certain open source communities are less open and try to shield their information to outsiders. In the case of MMBase for instance, a large portion of the developers uses IRC to communicate and discuss their ideas. Communication in this medium is much less transparent compared to communication on a mailing list.

Hypothesis 2: the more transparent the flow of information in an open source community, the more reliable the software.

4.3 Hypothesis 3: external oversight

HRT argues that the reliability in large-scale technical systems increases with the presence of aggressive and knowledgeable oversight from independent stake-holding interest groups and professional peers. Allowing for external oversight aims to prevent that organizations “close up” to such an extent that issues of safety and reliability will be ignored. However, in ensuring the success of external oversight, HRT specifies this should be aggressive and knowledgeable, thereby emphasizing the process in which oversight should take form.

There are a number of indications for the strong presence of external oversight in open source communities. Software has a number of different dimensions along which reliability can be measured. Some of them are relatively easy to quantify and measure. External groups frequently perform such measurements. Consider the examples from the introduction. The Reading consulting company measured the number of errors in both Linux and Apache modules. Or consider a very recent Forrester research that compared Microsoft and Linux on the number of bugs and their response rate to bugs.⁸ These research reports receive much attention

and publicity, both within and outside open source communities.

Another indication is the presence of a negative form of pressure to create reliable software. Anyone can report a bug or have a security flaw in software programs. However, this does not mean that the problem is automatically solved; it could be that no one has an interest in solving the problem. Thus, as long as the flaw or bug is not solved, everyone could make use of the flaw. All this creates a lot of pressure to solve the problem. One respondent reasons: “If they have spotted a bug that can cause security problems they will give it back and say that there is something wrong but they will not tell you how to fix it... I don’t need a 15 year old to own my site because he abuses a bug...”

Communities that produce software that is used by many people will attract more oversight from external stakeholders as compared to less popular open source software. Therefore, the chances are that the more popular the software the higher the reliability of the software.

Hypothesis 3: The more popular the open source software, the more reliable the software.

5 Future research

Based upon the propositions, we have constructed an initial research framework to guide our future research on the relationship between openness of open source communities and the ability to create reliable software (see Figure 1). This framework will have to be verified in subsequent research on open source communities. As a next step to achieve this goal, the authors will operationalize the variables in the framework into quantifiable or testable measures.

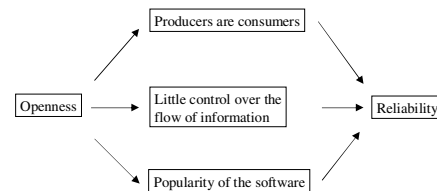


Figure 1: An initial framework

6 Conclusion

In this paper we have build a framework to begin to understand the relationship between openness of open source communities and their ability to produce reliable software. In doing so we hope to be able to verify and test one of the best-known and most-repeated axioms about open source communities, that with enough eyeballs, all bugs are shallow [23].

7 Acknowledgements

This paper is partly funded by a grant from the

⁸ The repost is available at:
<http://www.forrester.com/Research/Document/Excerpt/0,7211,34340,00.html> (August 2004).

Netherlands Organisation for Scientific Research (NWO). The reference number of the project is: 638.000.000.044N12.

8 References

- [1] Becking, J., S. Course, G. van Enk, H.T. Hangyi, J.J.M. Lahaye, D. Ockeloën, R. Peters, H. Rosbergen, & R. van Wendel de Joode. 2005. MMBase: An open source content management system. *IBM Systems Journal* **44**(2) 381-397.
- [2] Bollier, D. 2001. The cornucopia of the commons, *YES! Magazine*, Vol. 18, downloaded from the Internet: <http://www.yesmagazine.com/18Commons/bollier.htm> (December 2004).
- [3] Boots, F. P. 1995. No Silver Bullet: Essence and Accidents of Software Engineering. N. Heap, R. Thomas, G. Einon eds., *Information Technology and Society, A Reader*, Sage, London, U.K., 358-376.
- [4] Crowston, K. & B. Scozzi. 2004. Coordination practice within FLOSS development teams: the bug fixing process. Paper presented at the first International Workshop on Computer Supported Activity Coordination, Porto, Portugal.
- [5] Edwards, K. 2001. *Epistemic communities, situated learning and open source software development*. Paper presented at the 'Epistemic Cultures and the Practice of Interdisciplinarity' Workshop at NTNU, Trondheim.
- [6] Grabowski, M.R. & K.H. Roberts. 1999. Risk Mitigation in Virtual Organizations. *Organization Science* **10**(6) 704-721.
- [7] Hars, A. & S. Ou. 2002. Working for Free? Motivations for Participating in Open-Source Projects. *International Journal of Electronic Commerce* **6**(3) 25-39.
- [8] Hertel, G., S. Niedner & S. Herrmann. 2003. Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel. *Research Policy* **32**(7) 1159-1177.
- [9] Himanen, P. 2001. *The Hacker Ethic and the spirit of the Information Age*. New York: Random House.
- [10] Kling, R. ed. 1996. *Computerization and Controversy, Value Conflicts and Social Choices* (second ed.). Academic Press, San Diego, CA.
- [11] Landau, M., and D. Chisholm (1995), 'The Arrogance of Optimism: Notes on Failure-Avoidance Management', in: *Journal of Contingencies and Crisis Management* 3, No. 2, pp. 67-80.
- [12] LaPorte, T. R. 1996. High Reliability Organizations: Unlikely, Demanding and at Risk. *Journal of Contingencies and Crisis Management* **4**(2) 60-71.
- [13] Leveson, N. G. 1995. *Safeware, System Safety and Computers*. Addison-Wesley, Reading, MA.
- [14] Lin Y. 2004. *Epistemologically Multiple Actor-Centred System: or, EMACS at work!* Presented at 3rd Oekonux Conference, Vienna, Austria.
- [15] Mockus, A., R.T. Fielding & J.D. Herbsleb. 2002. Two Case Studies of Open Source Software Development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology* **11**(3): 309-346.
- [16] Osterloh, M. 2002. *Open Source Software Production The Magic Cauldron?* Paper presented at the LINK Conference, Copenhagen.
- [17] Perrow, C. 1984. *Normal Accidents, Living with High-Risk Technologies*. Basic Books, New York, NY.
- [18] Perrow, C. 1994a. The Limits of Safety: The Enhancement of a Theory of Accidents. *Journal of Contingencies and Crisis Management* **2**(4) 212-220.
- [19] Perrow, C. 1994b. Accidents in High-Risk Systems. *Technology Studies* **1** 1-20.
- [20] Perrow, C. 1999a. *Normal Accidents, Living with High-Risk Technologies with a New Afterword and a Postscript on the Y2k Problem*. Princeton University Press, Princeton, NJ.
- [21] Perrow, C. 1999b. Organizing to Reduce the Vulnerabilities of Complexity. *Journal of Contingencies and Crisis Management* **7**(3) 150-155.
- [22] Perrow, C. 1999c. *Y2K as a Normal Accident*, paper presented at the International Conference on Disaster Management and Medical Relief, Amsterdam 14-16 June, 1999, from the internet: http://europa.eu.int/comm/environment/civil/prote/cpactiv/dmmr-1999/papers_cluster1/perrow.pdf.
- [23] Raymond, E. S. 1999. *The Cathedral and the Bazaar: Musings on Linux and Open Source from an Accidental Revolutionary*. Sebastapol: O'Reilly.
- [24] Rijpma, J.A. 2003. From Deadlock to Dead End: The Normal Accidents-High Reliability Debate Revisited. *Journal of Contingencies and Crisis Management* **11**(1) 37-45.
- [25] Roberts, K. H. 1989. New Challenges in Organizational Research: High Reliability Organizations. *Industrial Crisis Quarterly* **3**(2) 111-125.
- [26] Roberts, K. H. 1990a. Managing High Reliability Organizations. *California Management Review* **32**(4) 101-113.
- [27] Roberts, K. H. 1990b. Some Characteristics of one Type of High Reliability Organization. *Organization Science* **1**(2) 160-175.
- [28] Roberts, K. H., G. Gargano. 1990. Managing a High-Reliability Organization: A Case for Interdependence. M. A. von Glinow, S. A. Mohrman eds., *Managing Complexity in High Technology Organizations*. Oxford University Press, New York, 146-159.
- [29] Rochlin, G. I. 1999. Safe Operation as a Social Construct. *Ergonomics* **42**(11) 1549-1560.
- [30] Sagan, S. 1993. *The Limits of Safety*. Princeton University Press, Princeton, NJ.
- [31] Sagan, S. 1994. Toward a Political Theory of Organizational Reliability. *Journal of Contingencies and Crisis Management* **2**(4) 228-239.
- [32] Schulman, P. R. 1993. The Analysis of High Reliability Organizations: A Comparative Framework. K. H. Roberts ed., *New Challenges to Understanding Organizations*. Macmillan, New York, 33-54.
- [33] Vogus, T. J., T. M. Welbourne. 2003. Structuring for High Reliability: HR Practices and Mindful Processes in Reliability-seeking Organizations. *Journal of Organizational Behavior* **24**, 877-903.
- [34] Weick, K.E. 1987. Organizational Culture as a Source of High Reliability. *California Management Review* **29**(2) 112-127.
- [35] Weick, K. E., K. M. Sutcliffe, D. Obstfeld. 1999. Organizing for High Reliability: Processes of Collective Mindfulness. *Research in Organizational Behavior* **21**, 81-123.
- [36] Van Wendel de Joode, R. 2004. Conflicts in open source communities. *Electronic Markets* **14**(2) 104-113.
- [37] Van Wendel de Joode, R., J.A. de Bruijn & M.J.G. van Eeten. 2003. *Protecting the Virtual Commons; Self-organizing open source communities and innovative intellectual property regimes*. The Hague: T.M.C. Asser Press.