Issues of Dependability in Open Source Software Development

Tony Lawrie and Cristina Gacek
Department of Computing Science
University of Newcastle, Newcastle upon Tyne, UK
A.T.Lawrie@ncl.ac.uk
Cristina.Gacek@ncl.ac.uk

Abstract

This paper presents issues raised by the articles, presentations, and discussions concerning Open Source Software, Trustworthiness, and Dependability at the Open Source Development Workshop held in Newcastle upon Tyne, UK, on the 25th & 26th of February 2002.

Introduction

We held a workshop on Open Source Software Development in Newcastle upon Tyne, UK, on the 25th and 26th of February 2002. The focus of this workshop was on dependability and open source software development. Dependability is a deliberately broad term which, among others, covers reliability, security, safety and availability [1][2].

Society's dependence on computer-based systems continues to increase and the systems themselves (embracing humans, computers and engineered systems) become ever more complex. Therefore, there is a current strong interest in developing improved means of specifying, designing, assessing, deploying and maintaining complex computer-based systems in contexts where high dependability is crucial.

Addressing the potential of the "open source approach" to contribute to aspects of dependability was the main objective of the February workshop. One key observation is that there are many, quite different, characteristics of projects which are described as "Open Source" [3]. The open source approach is sometimes characterised as "massively diverse human scrutiny": this both extends the idea of reviews or inspections and introduces a way of confirming final decisions about the inclusion of changes to a system. It poses interesting psychological, sociological and software engineering questions (http://www.dirc.org.uk/projects/dioss.html).

Examples of open source projects (e.g. operating systems, development tools, web and mail servers) indicate that a community can be built which can create software that is (claimed to be) highly dependable. It is not entirely clear what determines whether such a community can be built. Answering such questions requires interdisciplinary research involving people from various backgrounds, including (but not limited to) sociology and computer science.

We were fortunate enough to get Graham Button and Peter Neumann as keynote speakers, they added enormous value to the workshop. Graham Button is a sociologist, working for Xerox Research Centre Europe. He is well known for his work addressing Software Engineering and its development organisations. Peter Neumann is a computer scientist, working at SRI's Computer Science Lab. He has made a major contribution to the general problem of risks of computer systems. Of specific relevance to this workshop, Peter's more recent interest in Robust Open Source (RoS) has lead to a widely disseminated mailing list.

We received a considerable number of very good contributions to this workshop. The main areas addressed included: understanding open source, trust and dependability, community, and software engineering and open source. Paper submissions and conference attendees came from a variety of sources in industry, government and academia, some being personally involved in open source software development projects (i.e. Apache & Mozilla), others in using such systems, and still others in doing research on the topic. This was coupled with an interesting diversity of disciplinary backgrounds of the contributors, originating from several continents (America North and South, Europe and Oceania).

The discussions were lively and generated interesting insights. Both participants and organizers expressed having enjoyed the workshop's environment and discussions. In this short paper we share some of the ideas and issues raised and/or discussed during the workshop. The conference proceedings can be found at http://www.dirc.org.uk/events/ossdw/OSSDW-Proceedings-Final.pdf

Dependability and Open Source Products

The main focus of the workshop concerned attributes of Open Source Software (OSS) products and processes that promote dependability. Dependable systems are systems where trust can be justifiably placed in the service the system provides [4]. However, trust and trustworthiness can be different: trust may exist where there is no evidence to justify the reliance placed in a certain system, whereas trustworthiness suggests that there is assurance criteria to justify our confidence in a system [4]. To be a dependable and trustworthy[1] system, a computer system needs to embody certain attributes such as security, reliability, availability [5][1]. A number of the papers, presentations, and discussions, at the workshop, raised issues concerning not only the dependability of OSS products but also the dependability attributes of the OSS software processes that create them [4][6][5].

It was generally accepted, at the workshop, that OSS products are not *necessarily* more dependable than non-OSS products [7]. However, due to the influencing role of the software process and the increased openness of the OSS development paradigm, there seems to exist greater potential to actively and positively influence the eventual dependability of OSS products by influencing OSS processes [4][8]. For example, large U.S. government initiatives were presented that were primarily focused on promoting certain dependability attributes through influencing OSS project design

[1] The terms *Trustworthiness* and *Dependability* are equivalent. Trustworthiness is a U.S. term and Dependability is a European term.

goals and stimulating a more knowledgeable, disciplined, and principled approach to community-oriented software development in the future [8].

A potential problem particularly associated with OSS is the vulnerability to attacks by distribution of maliciously altered versions of software systems. How can OSS users be confident that the software version they have downloaded is trustworthy? The KeyMan software tries to avoid common pitfalls of simply using and checking PGP signatures by managing keys, certificates and signatures in a network of trust [9].

The breadth of applicability of the OSS approach was also considered in dependability terms [10][3]. It was argued, presented, and discussed that the OSS approach is largely driven by 'self-interest' [5][7] and this can result in the development of products that exist only in well established or well known product domains – such as systems-software or off-the-shelf-applications [3][5]. At the development stage, this may help in achieving dependability of such products through greater intuitive forecasting and anticipation of exceptions and faults that could occur during operational usage. On the other hand, this also begins to suggest the limitations of software products that can be developed using the OSS paradigm [3][5]. Such views were reinforced by the presentation of community support for OSS projects at Source Forge (http://sourceforge.net/) which indicated that very few OSS projects can generate enough support to be considered a sustainable success [11].

Dependability and the Open Source Process

The importance of architecture in composing trustworthy software systems was stressed by the keynote speaker Peter Neumann as being particularly suitable to collaborative development found in the OSS approach [4]. Whilst these aspects are of equal importance to both OSS and non-OSS development, it was believed that OSS approaches were considered to be less constrained by commitments to legacy applications and imposed reliance upon commercial obligations – such as time-to-market and budget cost-cutting [4]. Indirectly, these views were also further reinforced by the other keynote speaker Graham Button whose ethnographical social studies of traditional software engineering indicated that commercial, political, and schedule influences create barriers to open reviewing and sharing of source-code [10]. This can result in an increased need to improve the visibility through documentation of the development work [10]. Nevertheless, such contingencies are often thwarted through the prioritising of productivity over quality which then views documentation work being considered a less important overhead that does not help move the project forward to completion [10][7]. By contrast, the OSS approach focuses mainly upon the source-code as a critical co-ordination and evaluation device [12]. This, along with the inability to impose external process constraints of schedules and budgets, helps make the OSS development work less error-prone and more visible [7]. These influences may positively promote fault prevention strategies [5] and lead to higher levels of code reuse and increased knowledge acquisition during the OSS approach [10][13][5].

Increased dependability of computer systems rely also upon a range of software process characteristics. Firstly, effective tool/method support is considered vital for promoting higher quality assurance [8]. One aspect that became clear from two presentations [12][14] is that the stereotypical view of the "Bazaar" model is not as chaotic and ad-hoc as it first appears [7]. For instance, the ethnographic study of the Apache Cocoon project suggests that the work is carried-out along highly organised lines where individual developers orientate their effort towards advancing the project – as a whole [12]. Such findings were further reinforced by the case-study insights into the OSS process of the Mozilla Web-Browser project – where sophisticated process tool-support are used to enhance collaborative development, debugging, and reviewing of submitted code [14]. Secondly, the quality and experience of the people involved in software development were also considered, as the value of human intelligence, experience, and foresight in promoting trustworthy computer systems development and composition was particularly stressed by [4].

Therefore, the increased potential for diverse collaborative development and community bug finding were also discussed [7]. With respect to development, forms of human redundancy and diversity at the process level were presented and discussed. This was considered in terms of engineering diversity through differentiated non-functional design goals of developers to help assure dependable system composition [5]. There was some doubt whether the OSS paradigm does actually accommodate for human redundancy and diversity – at the development level [7]. However, there was general agreement that there exists the increased potential for both in the OSS approach [7]. In terms of community fault-detection, removal, and correction, formal probabilistic models were presented and discussed that considered the human diversity potential for individual and community usage profiles and its potential for increasing the reliability growth of OSS products via fault-finding, fault-reporting, and fault-removal over time [6]. Interest was shown whether the formal model could be applied to existing large OSS project tools (i.e. Bugzilla in the Mozilla project) to provide comparative research evidence for the increased potential for usage-diversity in the OSS process [7].

Issues and Implications for OSS and Dependability

A major consideration for dependability in OSS development concerns the need for research-based evidence to indicate which attributes of both the OSS and non-OSS processes can help assure dependability of the software products they produce. The open and public nature of the OSS approach offers lower confidentiality barriers of access for active influence and/or research involvement in OSS projects [8]. However, if adequate comparative research is not undertaken to measure the benefits of introducing and promoting formal software engineering initiatives into OSS projects, it will be difficult to objectively determine whether initiatives – such as the CHATS[2] programme, are responsible and justify increased trust in certain OSS products. It may be that the introduction of more traditional software engineering tools, methods, and techniques, may not result in dependability improvements and merely give the impression of OSS products being more dependable and trustworthy. These research issues raise long-standing contrasts of

[2] "CHATS" is an acronym for Composable High Assurance Trusted Systems. It is a U.S. Defence Advanced Research Projects Agency (DARPA) programme supporting high assurance in Open-Source operating system technologies [8].

the trustworthiness between the formal and informal approaches to software development. Therefore, the merits of both software development paradigms require comparative research based evidence to determine both the process attributes that result in increased system dependability [6] and their transferability from one paradigm to the other. For example, even if such formal influences may lead to increased trustworthiness of OSS products, can more disciplined traditional software engineering approaches be reconciled with the pragmatic approaches naturally, and culturally, adopted in OSS projects? It may be found that this only results in reducing product 'self-interest' and consequently support for such projects that are vital for leveraging the power of the OSS approach – in terms of increased diversity for collaborative development and fault-detection/correction. Consequently, it is not only dependability but also transferability that is important OSS and dependability research issues.

The nature of the types of products that can be developed successfully in the OSS approach is also an important consideration for dependability that was discussed at the workshop [7]. It has been discussed already that the OSS process may be restricted to developing only certain categories of software products – such as system-software [3][5][7]. However, dependable systems-software – such as operating systems, are considered a prerequisite for further composing and building on trustworthy and dependable systems [4]. As a result, the OSS approach may prove to be the most effective development approach for achieving a dependable system layer in IT infrastructures – leaving non-OSS approaches more suitable for the development of specific IT application domains [3] or where high levels of dependability are essential for initial system deployment (such as safety-critical systems) [6].

Business and Government attitudes vary also towards OSS. Product economics and functionality rather than system composition and dependability seemed to be the dominant commercial paradigm [7]. This is reflected in many of the strategic evaluation frameworks that have emerged to appraise the suitability of OSS product procurement [7]. Some focus solely upon the commercial advantages – in terms of viewing OSS as a cheap IT infrastructure alternative [7]. Others, however, are more encompassing, and incorporate required system-oriented properties relating to non-functional dependability and quality attributes (i.e. security, availability, reliability etc.) [15]. These frameworks are indicatory that system, quality, composition, and dependability, are considerations that are increasingly becoming more of an explicit infrastructure analysis and trade-off consideration in making strategic IT/OSS business decisions.

Finally, one other aspect that indicates the contrast between the informal OSS approach and the more formal software engineering processes are the methods and tool-support used in the respective paradigms. In OSS, tools are geared towards enhancing human collaboration and co-ordination during the development activities [12][14], whereas, traditionally, software engineering has typically been more oriented towards reducing and deskilling the human role of the developer through tool-support and methods that automate the software construction task wherever possible. This raises considerations also regarding trust issues of the respective OSS and non-OSS paradigms. Can OSS products be trusted if the OSS process itself is not trusted? In OSS, it appears that human innovation and creativity is actively promoted and encouraged, whilst the traditional software engineering paradigm appears to trust the methods, techniques and tools that seem to dominate in that paradigm. Therefore, trust issues connected with this centre upon that of increased human development freedom offered by the OSS approach.

Nevertheless, both paradigms still recognise that there is no substitute for human intelligence, experience, and foresight in achieving trustworthy systems and composition [4]. In this respect, there still appears to be little known concerning the true value and creative role of individual and collaborative design decision-making that may result in greater system dependability. Yet, it is clear from [12][14] that the increased openness and availability of OSS tool-support and source-code repositories begins to permit such investigations. Furthermore, associated academic initiatives – such as the GENESIS project, offer future opportunities to investigate and gain insight into what individual and collaborative development decisions promote or hinder design for dependability [16].

Conclusions

Whilst OSS products may be limited to the development of systems-oriented software, such systems are vital for further trustworthy composition and building of dependable systems.

At the process level, the OSS approach is not subjected to the same level of negative external process constraints of time and budget that can often subtly undermine the development of dependable systems within an organisational setting. Furthermore, despite the characterisations of the OSS approach as being highly ad-hoc and chaotic, OSS projects appear to be highly organised, in many cases, and provide tool-support focused upon enhancing human collaboration, creativity, skill, and learning – considered vital in developing trustworthy systems.

Nevertheless, the drive to improve the quality assurance of the OSS process and influence them through the introduction of tools, methods, and techniques from traditional software engineering raises issues concerning "how" trustworthy the OSS process, itself, is often perceived by organisations and government departments with a high dependency upon IT infrastructures.

It is clear that there is as much variation in attributes among OSS projects as among non-OSS ones. Several of these attributes are not restricted to either class of projects [3]. Hence dependability should be dealt with at the project attribute level and not by using such broad terms as OSS and non-OSS. Comparative research is therefore required – not only to provide evidence of which process attributes from both paradigms can improve the dependability of future software products, but also to determine whether such attributes can be successfully transferred from one paradigm to the other.

Acknowledgements

This workshop could not have been successful if it wasn't for its excellent set of participants, each of which openly shared his/her own personal insights and experiences on various matters. Hence we thank them all for playing their pivotal role so well.

Another extremely important success factor was the support and

dedication of several people involved in various aspects of the workshop organisation. We are very thankful to all of them, including: Joan Atkinson, Budi Arief, Denis Besnard, Angela Birrell, Diana Bosio, David Greathead, Cliff Jones, Mark Rouncefield, Carles Sala-Oliveras, Claire Smith, and Tim Smith.

We are grateful for the support of our sponsors, the Department of Computing Science and the Centre for Software Reliability at the University of Newcastle, as well as the UK EPSRC project on Dependability Interdisciplinary Research Collaboration (DIRC – http://www.dirc.org.uk/).

References

[1] Laprie, J.C. (Ed.) (1992): Dependability: Basic concepts and terminology — in English, French, German, Italian and Japanese. In *Dependable Computing and Fault Tolerance*, Vienna, Austria. Springer-Verlag.

[2] Randell, B. (2000): Turing Memorial Lecture: Facing up to faults. In *The Computer Journal*, Vol. 43, No. 2, pp. 95-106.

[3] Lawrie, T., B. Arief, and C. Gacek (2002): Interdisciplinary Insights on Open Source. In *Proceedings of the Open Source Software Development Workshop, Newcastle upon Tyne, UK, February 25-26, 2002*, ed. C. Gacek and B. Arief. pp. 68-82.

[4] Neumann, P. (2002): Developing Open Source Systems: Principles for Composable Architectures (keynote speech). In *Proceedings of the Open Source Software Development Workshop, Newcastle upon Tyne, UK, February 25-26, 2002*, ed. C. Gacek and B. Arief. pp. 2-19.

[5] Lawrie, T. and C. Jones (2002): Goal-Diversity in the Design of Dependable Computer-Based Systems. In *Proceedings of the Open Source Software Development Workshop, Newcastle upon Tyne, UK, February 25-26, 2002*, ed. C. Gacek and B. Arief. pp. 130-154.

[6] Bosio, D., B. Littlewood, L. Strigini, and M. J. Newby (2002): Advantages of Open Source Processes for Reliability: clarifying the issues. In *Proceedings of the Open Source Software Development Workshop, Newcastle upon Tyne, UK, February 25-26, 2002*, ed. C. Gacek and B. Arief. pp. 30-46.

[7] This reference refers to prominent issues raised by participant delegates during the discussion sessions at the workshop.

[8] Murphy, R. and D. Mauhgan (2002): Trusted Open Source Operating Systems Research and Development. In *Proceedings of the Open Source Software Development Workshop, Newcastle upon Tyne, UK, February 25-26, 2002*, ed. C. Gacek and B. Arief. pp. 20-29.

[9] Laurie, B. and M. Byng-Maddick (2002): KeyMan: Trust Networks for Software Distribution. In *Proceedings of the Open Source Software Development Workshop, Newcastle upon Tyne, UK, February 25-26, 2002*, ed. C. Gacek and B. Arief. pp. 197-203.

[10] Button, G. (2002): Organisational Considerations in the Work of Software Engineering (keynote speech). In *Proceedings of the Open Source Software Development Workshop, Newcastle upon Tyne, UK, February 25-26, 2002*, ed. C. Gacek and B. Arief. pp. 1.

[11] Hunt, F. and P. Johnson (2002): On the Pareto Distribution of Open Source Projects. In *Proceedings of the Open Source Software Development Workshop, Newcastle upon Tyne, UK, February 25-26, 2002*, ed. C. Gacek and B. Arief. pp. 122-129.

[12] Mackenzie, A., P. Rouchy, and M. Rouncefield (2002): Rebel Code? The Open Source 'Code' of Work. In *Proceedings of the Open Source Software Development Workshop, Newcastle upon Tyne, UK, February 25-26, 2002*, ed. C. Gacek and B. Arief. pp. 83-100.

[13] So, H., N. Thomas, and H. Zadeh (2002): What is in a Bazaar? A Model of Individual Participation in an Open Source Community. In *Proceedings of the Open Source Software Development Workshop, Newcastle upon Tyne, UK, February 25-26, 2002*, ed. C. Gacek and B. Arief. pp. 101-121.

[14] Reis, C., R. Pontin, and M. Fortes (2002): An Overview of the Software Engineering Process and Tools in the Mozilla Project. In *Proceedings of the Open Source Software Development Workshop, Newcastle upon Tyne, UK, February 25-26, 2002*, ed. C. Gacek and B. Arief. pp. 155-175.

[15] Kenwood, C. (2002): A Business Case Study of Open Source Software. In *Proceedings of the Open Source Software Development Workshop, Newcastle upon Tyne, UK, February 25-26, 2002*, ed. C. Gacek and B. Arief. pp. 47-67.

[16] Boldyreff, C., D. Nutter, and S. Rank (2002): Architectural Requirements for an Open Component and Artefact Repository System within GENESIS. In *Proceedings of the Open Source Software Development Workshop, Newcastle upon Tyne, UK, February 25-26, 2002*, ed. C. Gacek and B. Arief. pp. 176-196.