Modelling the Dependability of Free and Open Source Software: A Quantitative Approach

Anas Tawileh, Jeremy Hilton, Steve McIntosh
School of Computer Science, Cardiff University
5 The Parade, Cardiff CF24 3AA, UK
{m.a.tawileh, jeremy.hilton, s.b.mcintosh}@cs.cardiff.ac.uk

Abstract – The increasing popularity and adoption of Free and Open Source Software (F/OSS) has fostered heated debates about the dependability of software developed in such an open, highly distributed context. Claims are made in favour of and against F/OSS as a viable alternative to proprietary software. Due to the complex nature of software dependability, and the large number of interacting factors involved, a wider systemic view is required to produce more defensible claims regarding dependability. In this paper, we propose a quantitative approach based on system dynamics to validate stated claims about F/OSS dependability. We sketch a first attempt towards a computer simulation model to test different hypotheses by using empirical data. The model applicability is demonstrated by an illustrative example; the preliminary results obtained are comparable to data reported in the literature. The example supports our belief in the validity of the system dynamics approach as a testing vehicle to explain observed phenomena and support or disprove argued hypotheses. The model is also useful for predicting future behaviour of F/OSS development projects.

I. INTRODUCTION

Growth levels of Free and Open Source Software (F/OSS) have been increasing at a substantial rate over the past few years [Whee05]. Users and decision makers are realizing the value of adopting F/OSS software and many organizations are seriously considering it as a viable alternative to proprietary, commercially distributed applications. While such growth and expansion could be attributed to multiple characteristics inherent in the F/OSS development products and process, dependability remains a critical aspect, and an area that needs further research and exploration if F/OSS is to make inroads into the business market.

The F/OSS development process is based on a set of rules and principles radically different from the traditions governing the organization of proprietary software development. Based on intense collaboration and communication, developers from all around the globe combine their efforts to produce high quality software. The final applications are built by incorporating the different contributions submitted by these developers. In order to cope with the complexity of managing and coordinating efforts on such a high scale, the community has invented novel mechanisms to maximize
the efficiency of its development activities. For example, F/OSS development relies on frequent release of software versions and a transparent, open peer review process to discover software defects and enhance quality. These mechanisms have proved to be highly successful in enabling the inherently diverse F/OSS community to produce high quality software applications, which are currently considered to be a serious rival to the established proprietary software applications. This is clearly evident in the higher levels of adoption of some F/OSS packages like the GNU/Linux operating system and the Apache web server.

Much effort has been invested in exploring and explaining the distinct characteristics of F/OSS and its open development process that enable it to compete against software which is developed in traditional commercial settings [MoFH02] [BLSN02]. The literature investigates many aspects of the F/OSS paradigm and how they compare against proprietary software development. Among these aspects, and an area that has attracted significant attention lately, is the dependability of F/OSS. Dependability is a broad term that covers many areas such as security, reliability and availability [LaJo02]. Also, the nature of dependability is perceived differently by different stakeholders [BLJM04]. Unfortunately, most of the work done so far has focused on specific issues. Although investigating these specific issues could generate a better understanding of the situation under observation, it tends to ignore the different factors affecting the situation and the complex interactions between them. This would result in explanations that are partial at best, and in some cases may lead to strange interpretations of the observed phenomena.

In this paper we propose a more elaborate, systems approach to the study of F/OSS dependability. By taking a systems perspective, a richer understanding of the problematic situation can be achieved, taking into consideration all the different involved factors and their interactions. Such an understanding would facilitate the inquiry process and integrate the results reported in the literature to examine their validity and applicability when exercised in a complete, systemic setting; largely similar to the situation in the real world.

The next section provides some background information about software dependability in general and how it applies to F/OSS in particular. It reviews the relevant literature in order to lay the foundation for the progression of the discussion. Next, justification for the systems approach to investigate the problem is presented, along with our vision of the purpose of the dynamic model. Afterwards, model development is described and the resulting model sketched. Some preliminary analysis of the model behaviour is discussed as well. The model is then validated by using data sets from real world situations, and the results compared against values reported in the literature. The utility of the model and its applicability in practice are also described and demonstrated by an
II. SOFTWARE DEPENDABILITY AND F/OSS

With the increasing reliance on computers and software in almost every field of human activity, the need for more responsive, trustworthy and secure software has become a priority. Dependability of software systems has therefore gained substantial attention. In order to create more dependable software, a clear understanding of what dependability means in the context of software development, and a proper appreciation of its impact on the software development process is required.

Dependability is a broad term that covers different characteristics of the software artefacts, including, but not limited to, security, reliability, safety and availability [Lapr92]. Neumann [Neum02] suggests a practical definition for dependability: "Dependable systems are systems where trust can be justifiably placed in the service the system provides". Therefore, determining the dependability of a particular software artefact should consider the different elements that promote higher confidence in this artefact. The process and methods used to produce the software artefact also have major influences on its dependability characteristics, including user confidence in claimed dependability and consideration of the nature of the development process in the context of software dependability is of prime importance.

The distinct nature of the development processes within the F/OSS community has fostered heated debate about their ability to produce highly dependable software. Some might argue that exposing the source code would stimulate the exploitation of more vulnerabilities in the software and the introduction of backdoors, and therefore, would negatively affect security. Advocates of F/OSS, on the other hand, claim that by releasing source code and facilitating more intensive peer reviews, software defects will be discovered and fixed much faster, resulting in much higher levels of security [Payn02]. Eric Raymond describes this feature of the F/OSS development process as: "Given enough eyeballs, all bugs are shallow" [Raym02]. Another argument has been made about the honest motives of source code reviewers. The thread is that although availability of source code would enable more reviewers to check its integrity and security, doubts are often raised that people who possess the requisite knowledge to undertake such review do not always have entirely honest motives [Levy00].

These arguments illustrate the highly subject nature of the claims made about the dependability of F/OSS. Both sides are using the same evidence to support their completely different thesis [ABGR02] [Payn02]. It is therefore very useful to investigate the different factors affecting the dependability of
software in general, and F/OSS in particular, in order to clarify the different perspectives for or against F/OSS. This was described in a study about the issues of dependability of F/OSS [ABGR02]. The study stresses that due to the multiple facets of dependability, generalized claims such as "F/OSS is inherently more dependable than proprietary software" are of little value. The study suggests that "there are different aspects of openness that may impact the dependability of software systems, aspects that may manifest themselves both in open source software and in non-open software projects" [ABGR02]. Littlewood and Strigini [LiSt00] confirm the need for an empirical method to clarify the advantages and disadvantages of the different aspects of the F/OSS development approach in influencing the dependability of resulting software.

Different studies concluded with fairly similar results [Payn02] [Levy00], suggesting that F/OSS is not inherently more secure or more dependable than proprietary software. However, there are different aspects of the F/OSS development process that would greatly affect its dependability. In an attempt to analyze the role of the development process in producing more secure software, Payne [Payn02] suggested a multilayered analysis model that incorporates different security and availability measures which are scored and averaged to form an overall indicator of the system's dependability. Another model was developed by Bosio el al [BLNS02] to determine the possible effects of certain factors on the dependability of F/OSS software. They propose a model to investigate and validate each claim made about F/OSS dependability based on empirical observations. The approach is fully described in [BLSN02].

III. A SYSTEM DYNAMICS APPROACH

Given the complex interaction of the different factors affecting software dependability, approaching the problem with a narrow focus on specific issues yields limited results. In an attempt to clarify the mechanisms governing a complex, multi-faceted phenomenon such as software dependability, the full picture might be lost, and the interactions among the different factors may be unintentionally left out.

We propose a different approach to evaluate and test claims made about the dependability of F/OSS. Initially, we wanted to obtain further insight into the situation by taking a more elaborate, holistic perspective. System Dynamics, first introduced by Jay Forrester [Forr68], is a computer aided approach for analyzing and solving complex, systemic problems.

Human thinking is limited by nature, and can be rapidly overwhelmed by complex formulae and large interacting factors. System Dynamics' modelling provides a set of tools and techniques to aid human brains in focusing on the big picture of the problem under study, without losing the details. By
making explicit the different factors and their interactions on an Influence Diagram, a richer understanding of the problem structure can be achieved, and behaviour patterns can be traced back to their real causes more easily. When supplemented by computer simulation, the model can be simulated to reproduce the behavioural patterns of the system in the real world. The simulation model can be used as a testing vehicle to analyze and interpret historical phenomena, and to predict future behaviour.

We aim to produce a System Dynamics model to represent the different aspects of software dependability. The developed model captures the multiple factors affecting dependability and provides a usable tool to test and validate different claims about F/OSS dependability. We can utilize the model to discover the specific effects of changes in the development environment that might have a visible impact on the dependability attributes of the developed software. This will make the validation of different hypotheses about software dependability and F/OSS much easier. Possible hypotheses to be tested may include: What feature of the F/OSS development process has the highest impact on the developed software dependability? Does a F/OSS application have fewer vulnerabilities than a comparable proprietary software product which has the same characteristics? Does the open peer review process adopted in F/OSS have any impact on reducing software defects?

The first attempt to produce a system dynamics model of the F/OSS development process was done by Antioniades et al [ASAB03]. They aimed to model the different interactions within the whole F/OSS development community in order to predict future trends. We believe that the scope of the proposed model was very large. This has led the authors to make many predictions and estimations for different elements and parameters within the model. The model was validated using one case study of the Apache web server. Although the reported results were fairly satisfactory, little work has been reported since then to further confirm the model's general applicability. We suggest that by restricting the scope of our model to the dependability aspects of software development, better clarity can be achieved in the model construction, and fewer assumptions would be required. The developed model, once validated well enough, can be extended to incorporate more aspects of the F/OSS development process.

IV. TOWARDS A DYNAMIC MODEL OF SOFTWARE DEPENDABILITY

We have developed a preliminary model to illustrate the interrelations between different factors affecting software dependability. This model serves as the basis for further research and expansion to incorporate the effects of more factors that may influence dependability. The model can also be extended to demonstrate the economics of software dependability by incorporating mechanisms such
as the iDave ROI model proposed by Boehm et al [BLJM04]. The current version of our model aims mainly to model the dynamics underpinning the security aspects of dependability, primarily the vulnerabilities that could negatively affect software dependability.

Software that has fewer residual vulnerabilities is considered to be more dependable from a security point of view. Understanding the mechanisms that affect the number of residual vulnerabilities is very important to clarify some aspects of software dependability. Vulnerabilities are a specific class of software defects [ScBL90]. During the development process, developers strive to discover and fix any defects in the code they are developing. However, because of the inherently complex nature of software applications, some defects will inevitably remain after the software is released. A subset of these defects are potential security vulnerabilities. Evidence in the literature indicates that a significant fraction of software vulnerabilities are discovered externally [AlMa05a].

Different attempts to model the software vulnerability discovery process have been developed [Fink98] [BLNS02] [AlMa05a] [ShDa05]. Most of these efforts use the same empirical techniques implemented in software reliability growth models to predict the potential residual defects or vulnerabilities. These approaches require statistical data about the defect detection behaviour in the specific development project in order to predict future trends. This requirement makes the validation of the proposed models fairly difficult in commercial proprietary software projects. Companies do not usually release data about the defects in their products, as a measure to protect reputation and to avoid negative publicity. In the F/OSS world, access to vulnerabilities reporting and removal data is much easier, as all the information relating to software defects and their fixes is left widely open in online repositories such as Bugzilla [Bugz06].

Another concern about the applicability of the current dependability models is the need for historical data about the project under consideration. This will lead to unstable estimations in the early stages of the project lifecycle where no sufficient information is available. A model was proposed [XiHW97] to rectify this issue by using historical data of similar projects to aid the prediction of residual defects and vulnerabilities. Our model can accommodate the different approaches for the estimation of software defects removal by enabling more flexible adjustment of the model's parameters.

The profile of vulnerabilities removal depends on many different factors. After the software is released, people start using it. Over time, defects and vulnerabilities in the software are discovered, reported, and eventually fixed. Therefore, the longer the software is used, the more defects will be detected and fixed, increasing the dependability of the software. In addition to usage time, the defect discovery rate will be influenced by the number of users of the software. More popular software will
attract larger numbers of users, who will diversify the utilization profiles of the software, improving the odds of discovering defects. However, an important factor in relating user population to software dependability is the willingness of users to report the defects they encounter. It can be argued that expert users are more willing and likely to report vulnerabilities than normal users. Alhazmi and Malaiya [AlMa05] investigated these dynamics and proposed two models to assist vulnerability assessment of software: a time based model and an effort based model. Both models attempt to predict the discovery trend of software vulnerabilities in relation to the time the software has been tested or put into operation for, or the effort invested in discovering vulnerabilities (which is proportional to the users' base of the software). Usually, software popularity changes over time. The system dynamics approach we propose will facilitate more flexibility in testing wider ranges of scenarios and possibilities by incorporating simultaneously the dynamics of both time and effort.

Interactions among these different factors and their effect on the dependability of the produced software are illustrated in the influence diagram presented in Figure 1. This list is by no means exhaustive of all possible influences related to software dependability. However, it provides the basis for further research and elaboration to incorporate more factors and dynamics. We assume that all the factors that are not included in this model (such as the link between dependability growth and user base size, the availability of formal requirements, changes of usage profiles over time, etc.) will be constant. This means that they will not have any negative or positive impact on software dependability.

![Influence Diagram for Software Dependability](image)
The influence diagram provides useful insights into the dynamics of vulnerability discovery and dependability of software development. Investigating the interactions between the model's elements explains different observable phenomena in software adoption and dependability growth. For instance, the influences in the model indicate that when the number of residual vulnerabilities in a given piece of software decreases, the software will be perceived as being more dependable. This will lead to an increase in the users' base. More users will encounter and report more vulnerabilities, and depending on the availability of qualified developers, the reported vulnerabilities will be fixed and the residual defects decreased. This is a reinforcing loop. The loop's growth is limited by the number of available developers and the willingness of users to report vulnerabilities. When the software project is perceived to have high dependability, it will attract more developers because of two reasons: firstly, developers are more likely to join a project with an established reputation for quality, and secondly, projects with higher popularity magnetize more developers. Clearly, this observation explains the failure of some F/OSS project because they could not recruit the required number of qualified developers to fix the reported vulnerabilities. When considering the second limit to dependability growth in the model, it can be argued that projects with higher percentages of active users would benefit from the higher reporting rates. This explains the considerably higher dependability of server side F/OSS application such as web servers (Apache) and database servers (MySQL) compared to desktop applications, because inherently, these projects command a population of more active (or expert) users.

However, the insights gained from analysing the influence diagram are mainly qualitative, in the sense that the model does not provide any measurable values of its elements. For the proposed model to become practically useful, more quantifiable data should be incorporated. A computer simulation version of the model was developed using the iThink™ simulation package to facilitate more empirical inquiry into the dynamics of software dependability.

Our main interest at this stage is the number of software vulnerabilities and how they can be affected by other factors in the development process. The vulnerability discovery models described in the literature use probabilistic distributions to predict the number of residual vulnerabilities or defects and the vulnerability discovery rate. The first attempt to produce a comprehensive dynamic model of F/OSS projects by Antoniades et al also adopts the same approach [ASAB03]. The values of these parameters can not be precisely known before the project starts. However, they can be predicted using historical data and information from similar projects to start the simulation. Upon progress of the project, the values can be readjusted as real data about the specific development environment become
available. This continuous readjustment of parameters (backward propagation) will enhance the results of the simulation and increase the predictions' accuracy [ASAB03].

V. MODEL VALIDATION AND UTILISATION

To validate the developed dynamic simulation model, we used vulnerability data collected for two versions of Red Hat Linux (6.2, 7.1) and reported in Alhazmi et al [AlMR05]. They propose vulnerability density (the number of vulnerabilities identified in the unit size of the software code) as a measure of software dependability. The results reproduced by our model were similar to those reported by Alhazmi et al [AlMR05], which is unsurprising as we used the same approach and mathematical reasoning to predict the number of residual vulnerabilities and the vulnerability discovery rate. Figure 2 shows the results produced by our dynamic model compared to the real values reported by Alhazmi et al [AlMR05].

![Figure 2: Cumulative Vulnerabilities in Red Hat Linux 7.1](image)

The model can be utilized by different stakeholders to achieve different goals. Maintainers of F/OSS projects may use the model to analyze the behaviour of their projects, and to identify the causes of possible trends or events. They can also apply it to determine the release date of their software when they are satisfied with the level of residual vulnerabilities. Users may exploit the model to evaluate the maturity and dependability levels of different F/OSS applications when they make their selection decisions. The model also can be utilized by researchers and practitioners alike to inquire into and explain observed trends in the evolution of F/OSS software, and to employ it as a test vehicle to validate different possible hypotheses about software development in general, and F/OSS processes in particular.
To illustrate the possibilities afforded by considering a systemic perception of the problem situation, aided by a computer based, quantitative dynamic model, we tested the behaviour in the model under different circumstances. We increased the value of the parameter incorporated to represent the market acceptance of the software, which also reflects its users' base. As shown in Figure 3, the increase in market acceptance resulted in faster detection and reporting of vulnerabilities. This reinforces the stated claim about F/OSS development that by "releasing early, releasing often" [Raym00], software defects and vulnerabilities will be discovered and fixed earlier.

![Figure 3: Change in Cumulative Vulnerabilities Detected Due to Higher Market Acceptance for Red Hat Linux 7.1](image)

VI. CONCLUSIONS

Software dependability is becoming an increasingly important issue with the increase in reliance on computer systems and software in all sectors of human activity. Many claims are made about the dependability of F/OSS and how it compares against proprietary software, with very different, sometimes conflicting conclusions. In addition, dependability is perceived differently by different stakeholders. Subjectivity of interpretations remains a significant issue. Therefore, there is a need to establish a coherent tool to test and validate the different hypotheses with empirical evidence. We have proposed a wider, systemic approach to incorporate the many factors affecting software dependability and the interactions among these factors. We have suggested system dynamics as an appropriate tool because it provides more justifiable models and the ability to develop a computer simulation with which to interpret quantitative data and test the effects of changing selected parameters. We have described our first attempt to sketch a basic model to represent some essential factors affecting software dependability. This model, if validated using sufficient empirical data and
other research, would serve as a basis for more elaborate, comprehensive models to accommodate and investigate other aspects of dependability thought to be relevant. The model was then investigated to derive qualitative explanations of observed phenomena in the F/OSS world. To facilitate the quantitative evaluation of our approach, we developed a computer simulation version of the model and validated its behaviour against work reported by other authors. Exploitation of the model was illustrated by testing the change of the model behaviour in response to adjustments in certain key factors. The results confirmed the expected utility of the model as a testing vehicle to validate different proposed hypotheses about dependability of F/OSS.

This model should only be considered as a first attempt to build a more comprehensive dynamic model for software dependability. Further research is required to validate the model using more extensive vulnerability data; this should include projects with different characteristics and development environments and processes. Moreover, work needs to be done to extend the model to include different aspects of software dependability and integrate the factors that affect them. The quantification of these factors and the relationships between them also requires further investigation. The model’s utility and applicability will be enhanced by incorporating appropriate metrics that can be obtained reliably.
VII. REFERENCES


