Defining an Open Source Software Trustworthiness Model

Davide Taibi
Università degli Studi dell'Insubria
via Carloni, 78
22100 – COMO - ITALY
+39 31 238 6328
davide.taibi@uninsubria.it

ABSTRACT

In an ideal word, totally trustworthy software would provide an absolute guarantee that it will perform its required functions under all possible circumstances, will do so on time, and will never perform any actions that have hazardous consequences.

In the real world, this hardly ever happens, since different software products have different degrees of trustworthiness.

In this paper, we show how we set out to develop a trustworthiness model for Open Source Software, by identifying a few quality factors based on a GQM plan.

Based on a large number of interviews, the factors that are believed to determine the trustworthiness of Open Source Software will be analyzed. Then, a set of repositories and projects will be identified and tested so as to gather information about their intrinsic characteristics and check if it is possible to measure the previously identified factors. Then, a number of tools will be developed to measure the factors for which there are no tools available. Finally, we will build and validate a trustworthiness-based model.

Categories and Subject Descriptors

D.1.8 Distributed programming, D.2.2.c Distributed/Internet based software engineering tools and techniques, D.2.10.h Quality analysis and evaluation, D.2.18.f Quality process analyses, D.2.19.c Methods for SQA and V&V, D.2.m Miscellaneous

General Terms

Measurement, Documentation, Performance, Reliability, Legal Aspects.

Keywords

Software trustworthiness, Open Source Software

1. INTRODUCTION

Modern society depends on large-scale software systems of astonishing complexity. Because the consequences of their possible failure are so high, it is vital that software systems should exhibit a trustworthy behavior.

Trustworthiness is a major issue when people and organization are faced with the selection and the adoption of new software. Although some ad-hoc methods have been proposed (see for instance [5]), there is not yet general agreement about software characteristics contributing to trustworthiness.

Therefore, this work focuses on defining an adequate notion of trustworthiness of Open Source products and artifacts and identifying a number of factors that influence it to provide both developers and users with an instrument that guides them when deciding whether a given program (or library or other piece of software) is "good enough" and can be trusted in order to be used in an industrial or professional context.

2. WHAT TRUSTWORTHINESS IS

Although there is a good deal of research work on software trustworthiness, the traditional software trustworthiness assurance mechanisms mainly focus on security and dependability properties of software behavior.

Since Anderson [1] first proposed the concept of Trusted System in the 1970s, both academic and industrial circles have studied the trustworthiness of IT systems and have come up with definitions of trustworthiness. From the system perspective, ISO/IEC 15408 specifications [2] defines trustworthiness as by saying "A trusted component, operation or process is one whose behavior is predictable under almost any operation condition and which is highly resistant to subversions by application software, viruses and a given level of physical interference." The National Institute of Standards and Technology (NIST) defines trustworthiness as "software that can and must be trusted to work dependably in some critical function, and failure to do so may have catastrophic results, such as serious injury, loss of life or property business failure or breach of security." The Trusted Computing Group provides the following definition [3]: "something is trusted if it always behaves in the expected manner for the intended purpose.".

From the user's experience, Bill Gates provides the following definition [6]: "Trustworthy computing is computing that is available, reliable and secure as electricity, water services and telephony."

Since 2006, the concept of trustworthiness has also been investigated in the QualiPSo project (http://www.qualipso.eu), an

ongoing initiative funded by the EU, which proposes a coherent and systematic evaluation of the trustworthiness of OSS projects, and aims at promoting the diffusion of OSS by focusing on OSS trustworthiness.

In general, trustworthiness is a holistic property that encompasses security, safety and reliability. To define trustworthy software, we can draw upon conventional notions of trust in other contexts. Trust is the reliance by one party on the behavior of another party. Trustworthiness is not a quality that can be claimed without being proved. Trust is a matter of perception and implies finding answer to non technical questions like "why should people have confidence in my software?" or even "how can I make users confident in my software?" Trust is a relationship that involves two parties, the actual and the expected behavior of software. It is always conditional on the context and operational environment.

People may want to know useful key information about any software before making any commitment to use it and so, when users want to adopt new software, they have to trust it. Usually, during the selection of new software, users start to check if the selected program does exactly what they want and they collect information about the products from other users. In this respect, the web is clearly an extremely valuable and easily accessible source of information. In fact, many websites record a wide range of users' opinions and comments about every kind of product.

Of course, there are other quality related factors that should be verified. Measuring trustworthiness is possible only if there are specific attributes to measure. For example, in measuring reliability there are many useful attributes (such as mean time to failure of hardware or software).

The problem surfaces both in Closed and in Open Source Software, but, while in Open Source Software we can measure the code quality, in Closed Source Software we can only trust the producer company.

3. THE APPROACH

Organizations perceive software trustworthiness on the basis of the role that software plays with respect to the organization itself. For instance, an organization could be a software producer, a customizer, a value adder, etc.

To determine the trustworthiness factors, we need to take into account different users, from end users to developers, from system administrators to upper managers. The identification of the characteristics and the influencing factors with the subsequent derivation of measures, based on the business goals and the analysis of the software products and artifacts, will be carried out by our group by means of goal-oriented approaches such as the Goal/Question/Metrics paradigm [4].

This is a necessary step: in software measurement, there is too often a lack of agreement about the real meaning of a number of software qualities. Based on these factors, we are going to define a set of measures, so as to capture the various components of trustworthiness from different viewpoints, and they will be collected based on both static (i.e., based only on the analysis of the source code or artifacts) and dynamic measures (i.e., based on the execution of the software code or, wherever possible, the software artifacts).

Afterwards, based on the identified measures, we are going to test a set of relevant Open Source projects.

Finally, we will build a trustworthiness-based model on the back of the test results.

The whole process is explained next.

3.1 Trustworthiness Factors

The first step focuses on defining an adequate notion of trustworthiness of software products and artifacts and identifying a number of factors that influence it.

The definition will be driven by the specific business goals elicited in the GQM plan. To this end, we have carried out a survey to elicit these goals and factors directly from industrial players.

The survey was conducted via interviews supported by a questionnaire, partially derived from the existing literature. We interviewed several people with various professional roles, trying to derive the factors from the real user needs instead of deriving them from our own personal beliefs and/or only by reading the available literature.

The questions in the questionnaire were mainly classified in three different categories:

- Organization, project, and role. These questions are needed to profile the interviewed person.
- Actual problems, actual trustworthiness evaluation processes, and factors.
- Wishes. These questions are needed to understand what should be available but is not, and what indicators should be provided for an OSS product to help its adoption.

Based on the analysis of internal and external characteristics of software products, measures will be identified to quantify the factors. The results will be useful to check the quality of the first factors identified and in case if it will be needed to change some of its.

We believe that dynamic metrics will be most useful for the software trustworthiness definition. Instead of only relying on the results of static code analysis, based on the execution of a test set it will be possible, as an example, to assess how closely related two components are during their executions. The intrinsically dynamic qualities can only be an approximated through static analysis. So, dynamic metrics will be collected in addition to traditional static metrics.

Afterwards, results will be considered, so as to take out useless factors and measures and introduce new ones. If needed, other empirical studies will be carried out, to provide more support for their usefulness.

3.2 Analysis of Relevant Open Source Projects and Artifacts

That activity will be composed by three rounds: the first, before the first round of empirical studies; the second, between the first and the second rounds of empirical studies; the third, after the second round of empirical studies.

Based on the goals previously identified, we will analyze software products and artifacts, to identify commonalities and differences and gather information on their general intrinsic characteristics

and the way software is used by the software industry. With the aid of automated tools, a number of Open Source Repositories will be analyzed. In case of unavailability of tools aimed to measure a specific factor, new tools will be developed.

In the second and third rounds, in order to focus the intrinsic characteristics analysis that appear to be relevant for trustworthiness evaluation, the experimental results will be taken into account.

3.3 Definition of Standard Test Approaches, Test Suites, and Benchmarks

One of the biggest OSS problems is the lack of a comprehensive and widely adopted testing strategy allowing only limited potential for verification. Mainly, OSS is developed without considering complete coverage tests. That problem mostly relies on trustworthiness. People cannot really trust a program if they do not know if it will do exactly what it should do.

A common test approach, applicable to a wide range of OSS, may considerably facilitate OSS verification.

We will develop a common performances and functional tests approach aimed to produce a representative set of test suite and benchmark specially considering the project previously identified in Section 3.2 and the quality factors defined as described in Section 3.1.

Additionally, we will investigate the potential of Aspect Oriented Programming (AOP)[7] in measuring dynamic measures without the need for manual code instrumentation.

Test suites will be defined for a project and artifacts subset that will address and cover the quality factors and trust goals.

In addition, as testing will also be used to collect dynamic metrics for OSS qualities that are usually quantified via static metrics, it will be necessary to use special care in the definition of the ways in which that metrics are going to be collected.

It must be noticed that the dynamic metrics must be selected dependent on of the available technologies. That depends on code instrumentation issues that deeply depend on the programming language, technologies and environment. During this analysis we will start addressing a specific language (e.g. Java) and then we will extend code instrumentation to other languages.

Through the trustworthiness factors, different scenarios will be defined aimed at modeling the different system usage.

We will define dynamic metrics and benchmarks to be collected based on the execution of the scenarios.

Different benchmarks will be used for different qualities and for different types of scenarios, which will be used to profile users. A user can choose one or more scenarios that are akin to the usage he/she intends to do of the system. Dynamic metrics will be collected at run-time. In order to guarantee privacy and non disclosure, users will be always given the possibility to select which metrics to collect and which metrics not to collect.

The purpose of the dynamic metrics collected for Scenarios are the same as the dynamic metrics collected for test suites. Since dynamic metrics are relevant only for a subset of the trustworthiness qualities, benchmarks will address such subset. We will study commonalities and differences of the results across different scenarios.

In order to address the complexity of system deployment in relation to different business scenarios, a coherent methodology for defining test suites and scenarios will be developed. This methodology will be applicable also to systems and environments outside the project set.

3.4 Tool Definition and Building

In this activity, we will identify, specify, and develop a set of tools that support the measurement and assessment of the quality factors of the software products and artifacts that affect trust in open source software products.

The tools developed will aim at making the evaluation of OSS trustworthiness as easy and seamless as possible for developers, users and managers. In particular, the tools will automate as many activities as possible and will provide developers, users, and managers with information that are reliable and useful for taking decisions involving trust in OSS.

When possible, tools will be obtained by adapting, extending and integrating existing tools.

The tools will integrate a number of OSS tools that support the creation of a measurement plan, starting from the main actors' and stakeholders' objectives and goals (developer community, user community, business needs, specific users, etc.), down to the specific static and dynamic metrics that will need to be collected to fulfill the goals.

Specifically, the tools will support:

- the definition of a customized measurement plan
- the collection of static metrics
- the definition of test suites
- the execution of test suites
- the definition of scenarios
- the collection of dynamic metrics
- the analysis of the collected data
- the transfer and exchange of the collected data to be analyzed by one or more actors or stakeholders
- the building of models of various nature; these could be based on Statistics or Machine-Learning techniques

3.5 Model Building

In this activity we will develop a trustworthiness models through two rounds. The objectives will be achieved through an experimentation sub-activity and a model-building sub-activity.

The model will use a number of factors as its independent variables and an assessment of trustworthiness by OSS practitioners and users as its dependent variable. Therefore, it will be necessary to collect data from OSS practitioners and users about the trustworthiness of existing OSS products.

In the first round, empirical studies will be carried out in industrial environments. In this activity we need to take special care in designing and executing the experiments in order to minimize the risk of having incomplete or misleading information. The empirical studies must be as little invasive as possible for the industrial environments studied to disturb the observed environment as little as possible maximizing the chances that data

are actually collected from the industrial environment. To this end, the automated tools built as described in Section 3.4 will be used.

At any rate, questionnaires and interview may also be used to collect additional pieces of information that would not be possible to retrieve from the raw data. The collected information will be collected and stored in repositories.

In the second round of empirical studies, based on the results of the first round of experiments, some measures may be deleted, while others may be added. These tasks will clearly provide inputs to the tool building and will rely on the tools to be carried out effectively and efficiently. In addition to data on trustworthiness, data on the cost-effectiveness of and practicality of the approach will be collected to assess the overall impact that the approach may have on industrial environments.

The information collected in the second round of empirical studies will be used to confirm and enrich the models obtained in the first round of empirical studies.

Finally, the information collected will be analyzed to find out whether the factors influence the trustworthiness of the OSS products and artifacts and the results obtained in the two rounds of empirical studies will be compared.

The analysis will be carried out via a variety of different statistical (e.g., Ordinary Least Squares, Logistic Regression) and machine learning (e.g., Decision Trees, Random Forests) techniques will be used for data analysis, based on the specific independent and dependent variables involved and the objectives of the data analysis.

As a result, the analysis of the data will be packaged in models and lessons learned that will have been validated and interpreted by researchers and practitioners and will be subject to a confirmatory second round of empirical studies.

Models and lessons learned will be made explicit and will be presented to the practitioners in the European software industries that participated in the definition of the business goals and in the empirical studies.

4. ACKNOWLEDGEMENT

The research presented in this paper has been partially funded by the IST project “QualiPSo", sponsored by the EU in the 6th FP (IST-034763).

5. REFERENCES

[1] Anderson J. P. Computer Security Technology Planning study, ESD-TR-73-51, Vol1, AD758 206, ESD/AFSC, Hanscom AFB, Bedford, M.A., October, 1972

[2] ISO/IEC, Information Technology-Security Techniques-Evaluation Criteria for IT security, Pat1: Introduction and genera Model 2nd ed. 2005-10-01

[3] Trusted computing group. TCG Architecture overview. V1. 2, 28 April 2004

[4] Basili V., and Rombach H.D., The TAME project: towards improvement-oriented software environments, IEEE Transactions on Software Engineering, June 1988.

[5] D.Taibi, L.Lavazza, S.Morasca. OpenBQR: a framework for the assessment of Open Source Software, Open Source Software 2007, Limerick, June 2007.

[6] C. Mundie, B.Gates: How Microsoft Is Refocusing on Security, Reliability, Privacy, and More, as Part of Trustworthy Computing Initiative, February 2002 (http://www.microsoft.com/PressPass/features/2002/feb02/02-20mundieqa.mspx)

[7] http://en.wikipedia.org/wiki/Aspect-oriented_programming