

# Qualification of Open Source Software for e-Government

---

Bernhard K. Aichernig

---

e-Macao Report 28

Version 1.0, January 2006





---

**Table of Contents**

1. Overview .....	1
2. Background .....	1
3. Objectives .....	2
4. Results .....	3
4.1. OSS in e-Government .....	3
4.2. Quality Assurance of OSS .....	4
5. Applications – A Web Server Case Study.....	6
References.....	8



## 1. Overview

The objectives of this research task were to:

- better understand the status and maturity of OSS software in e-government,
- raise the awareness of the role OSS can play in e-government,
- investigate in methods and techniques to overcome the skepticism regarding OSS quality for critical applications and
- initiate a research program on certifying OSS.

To meet these objectives we:

- reviewed the current status of OSS in the domain of e-government,
- developed and gave a 3.5-hours seminar to the government on OSS in e-government,
- did a 9 month research project on model-based testing of the Apache web-server and
- established a network of international researchers to develop a research group on certifying OSS.

## 2. Background

In 2000, the computer network of the institute of the author at the TU Graz, Austria, was hacked. The intruder caused severe damage by deleting several essential files on various servers. Although no sensitive data could be accessed, it took several days until the network services were back to normal operation. As an immediate consequence a new security policy was enforced. The use of ftp and telnet was prohibited, since these programs send passwords unencrypted via the Internet. Instead the Secure Shell SSH was to be used. Everybody considered this as a reasonable measure. However, it was completely inadequate. A closer analysis of the attack revealed that the intruder got access by exploiting a fault in OpenSSH, an open source implementation of the SSH protocol. The program that was supposed to secure the network actually caused the trouble! Not the insecure protocol of ftp or telnet was exploited, but the incorrect implementation of OpenSSH.

We realize that little is known about the quality of the open source software used to build up our computing infrastructure. Though generally convinced of the high quality that can be achieved by opening the source code for inspection, no defensible argument regarding the security could be given for a particular piece of software.

This is a major obstacle regarding the acceptance of OSS in critical software, as found in e-government. Therefore, this research task was about to find out about the quality of the existing OSS software in e-government as well as doing research to develop new methods and techniques to ensure a certain degree of quality. The vision of this research is to establish a certification framework for OSS.

### 3. Objectives

Before targeting a certification process for OSS, one must review the current status of OSS and decide if OSS is an option for e-government. Hence, the first objective was to:

- Better understand the status and maturity of OSS software in e-government.

This analysis should highlight the strength and weaknesses of OSS in e-government. The IT experts in the government need to be informed about the options OSS provides. Arguments in favor and against OSS need to be collected. This includes a review of other OSS e-government initiatives around the world. The objective is to:

- Raise the awareness of the role OSS can play in e-government.

Having informed the stakeholders of OSS applications in e-government, one must address the technical objections relating OSS. The main obstacle is the unknown degree of quality and support for OSS applications in a critical domain like e-government. This needs to be addressed by foundational research. Therefore, the next objective was to:

- Investigate in methods and techniques to overcome the skepticism regarding OSS quality for critical applications.

This is a huge objective and realistically we will limit ourselves to techniques we already have an expertise in. The overall strategy has to combine several techniques, which need the collaboration of several researchers. Therefore we are aiming at:

- Establishing a network of international researchers to develop a research group on certifying OSS.

This is a long term aim, since it in fact establishes a new area of research.

## 4. Results

### 4.1. OSS in e-Government

Research has been done regarding the status of OSS in e-government. A 3.5 hours seminar has been given on 3 January 2005 as part of the eMacao seminar series, where we presented the results and our view regarding OSS. Find below the abstract of the talk:

Open source software (OSS) plays an increasingly important role in the IT sector. Successful OSS projects, like the Linux operating system and the Apache web-server, have demonstrated the strength of the OSS development process, where self-motivated users and developers share knowledge via the Internet. These projects have shown that the parallel inspection process involved can lead to a level of quality that outperforms commercial software. Furthermore, OSS is free or at least very cheap, and companies are beginning to reduce their development costs by integrating OSS into their products.

This talk addresses the question if OSS is suitable for e-government solutions. After an introduction to OSS, including its license policies and development process, we are discussing the prospects and obstacles involved using OSS in general, and within e-government in particular. In addition, we give a survey on European government policies related to open source software.

In this seminar, we:

- 1) explained the terminology involved, had a brief look at the history of OSS, gave an overview on the GNU and BSD projects, looked at the OSS development process, explained the motivations by discussing the GNU manifesto,
- 2) reviewed the market share of famous OSS applications, compared the reliability data available - performance data were analyzed, security studies and cost of ownership studies shown, and discussed non-quantitative issues,
- 3) presented common arguments against OSS and discussed the answers to this objections from the OSS communities,
- 4) reviewed the e-government situation relating to OSS in Macao,
- 5) presented the findings of a Danish study on OSS in e-government [1],
- 6) showed important government initiatives towards OSS in Europe and
- 7) presented our conclusion and discussed with the government people.

The research on the status of OSS clearly showed that OSS is a growing competitor to commercial applications. The main motivations of governments throughout the world for starting OSS initiatives are non-technical: the fear of vendor-lock-in and independence.

The talk highlighted the fact that in certain areas, like Desktops, Web-servers and operating systems, the quality of OSS has reached a point where it can be applied in critical application domains, like e-government.

## 4.2. Quality Assurance of OSS

A nine-month project on quality assurance of OSS has been carried out. The results have been published in [2]. The project investigated the use of fault-based testing on protocol models to test the Apache server.

Fault-based testing is a technique where testers anticipate errors in a system under test in order to assess or generate test cases. The idea is to have enough test cases capable of detecting these anticipated errors. This paper presents a theory and technique for generating fault-based test cases for concurrent systems. The novel idea is to generate test purposes from faults that have been injected into a model of the system under test. Such test purposes form a specification of a more detailed test case that can detect the injected fault. The theory is based on the notion of refinement. The technique is automated using the TGV test case generator and an equivalence checker of the CADP tools. A case study of testing web servers demonstrates the practicability of the approach.

The area of specification-based testing has advanced over the last couple of years and the results contributed to reconciliation between testing and formal verification communities. Testing is now commonly acknowledged as a complementary V&V technique, if carried out systematically and well-founded. Gaudel was the most prominent author who started this process. Since then, many techniques and tools have emerged that generate test cases from formal specifications and are based on complete and sound testing theories.

However, the field is far from being complete, as the growing number of publications in this area indicates. Non-classical testing paradigms have to be studied and incorporated into theories. The formal underpinnings allow a deeper understanding of the relationships between testing and other verification theories, like simulation and refinement. This will lead to further applications and tools, like model checkers and constraint solvers, to generate test cases.

In this research, we developed a method that aims to advance the field in the following directions:

- 1) Mutation testing, traditionally applied to program text is applied on the specification level.
- 2) A model checker is not used to generate the test cases directly, but to generate test purposes, a high-level description of the testing goal. The method is founded on the testing theories of labeled transition systems. Tool support comes from the TGV test case generator as well as from the CADP tools.
- 3) The technique serves to test an OSS application. The faults to be injected are partly inspired by vulnerability reports published on the Internet. The Apache web-server was the case study application.

In particular, we addressed the following problems.

### **Problem 1** - Lack of test selection strategy.

The early work on conformance testing in the area of distributed systems was mainly concerned with the soundness and completeness of the testing theory. Emphasis was given to develop a realistic conformance relation and a test case generation algorithm that was sound (no false negatives) and complete (no false positives). Since the models were finite labeled transition systems (LTSSs), the problem of how to select a manageable subset out of

the exhaustive test set was not a major concern. Abstraction was used to cope with the complexity. This lack of a test selection strategy limited the application domain to highly abstract protocol specifications.

**Problem 2 - Identifying test purposes.**

To overcome this shortcoming, test purposes have been introduced. Here, a test purpose is a special LTS that specifies the subset of test cases to be generated. With test purposes, a tester can steer a test case generator according to his strategy. However, the problem remains, how many and which test purposes to select. Thus, the problem has been lifted, but not entirely solved.

In our approach, we want to support the tester in formalizing test purposes, by turning his focus on possible faults. Possible faults can be anticipated by inspecting a specification, by using domain knowledge, or by heuristic mutation operators. In all cases, the fault is modeled at the specification level by altering the specification. We call this altered version a mutant. The idea is to generate test cases that would find such faults in the implementation.

**Problem 3 - Equivalent mutants.**

A common problem in this approach is known as the Equivalent Mutant Problem. Not all mutations represent actual faults that can be observed at the interface level. Thus, no test case exists that can distinguish the original from such an equivalent mutant.

On the specification level, equivalence checkers can be used to eliminate such equivalent mutants. The problem is which equivalence relation, based on simulations, is appropriate for our purposes. Once, the equivalence relation is fixed, the problem is solved.

**Problem 4 - Test generation automation.**

The technique should automatically generate test cases. Many use the counter examples (or witnesses) produced by a model checker as test cases. However, a counter example is not a test case in the traditional sense. A test case should provide the stimuli and the responses for a system. However, a counterexample exemplifies only one possible choice of computation (a path). In case of non-determinism involved this is not sufficient for a test case, since a test case should predict and take care of all possible responses, as well as reject wrong responses.

Therefore, we propose to use the counterexample as a test purpose. A test case generator, then, will generate a proper test case to cover the counterexample. Hence, our idea is to generate test purposes from injected faults, such that the generated test cases will discover these anticipated faults. Fault-prevention, not structural coverage is our testing strategy.

## 5. Applications – A Web Server Case Study

In the following we briefly report the case study on testing web servers that serves to demonstrate that our technique is applicable. The aim was to test the correct implementation of parts of the HTTP protocol in the Apache web server. We focused on the GET-Method responsible for retrieving pages and limited ourselves to single client-server connections.

The source for the LOTOS model was the Internet standard RFC 2616 (Request for Comments). RFC 2616 specifies the syntax of the HTTP protocol in BNF and describes the semantics in natural language (English). Our model consists of two LOTOS processes, the client and the server, running in parallel. The client is issuing a request message and then waits for a response message from the server process.

The request message contains three parts, each one modeled as an action:

- 1) a Request Line (with a Method - here GET, a URI and the HTTP-version),
- 2) a Request Header and
- 3) an optional Request Body.

The Request Header facilitates conditional requests, like e.g. header `If_Modified_Since` supports a restricted download of pages that have been recently updated. The Response message of the server contains three parts, too: a Status Line (with the HTTP-version, a Status Code and a Reason), a Response Header (with information about the web page) and a Response Body (which contains the web page in most cases).

The choice of the level of granularity of the actions is a pragmatic one. Which part of the protocol is modeled as an action depends on the actual testing strategy and how the actions are easily mapped to real interactions with the web server. For example, the three parts of the Request Line have been merged into a single action, since we were not interested in testing variations of URI's and HTTP-versions.

Almost 1500 mutants were derived from the HTTP model out of which we selected about 100 interesting mutations that were partly reflecting ambiguities in the HTTP standard. From these we generated the test purposes according to the process described in the previous section and used TGV to generate the test cases.

The implementation under test was our institute's Apache Web Server 2.0.40 for Red Hat Linux with HTTP/1.1 protocol. The tests have been carried out manually via a telnet session to Port 80 of the web server's URL. This is possible since the HTTP protocol is ASCII based.

We did not expect to find major flaws in the Apache Web Server, since it has been widely used since years. However, we found a case where Apache behaves unexpectedly. As mentioned above, conditional requests can be formed by adding header fields. They serve to control the caching done by a proxy server. The combination of several such header fields is underspecified in the standard. However, on page 56 of RFC 2616 the standard says:

``An HTTP/1.1 origin server, upon receiving a conditional request that includes both a Last-Modified date (e.g., in an If-Modified-Since or If-Unmodified-Since header field) and one or more entity tags (e.g., in an If-Match, If-None-Match, or If-Range header field)

as cache validators, must not return a response status of 304 (Not Modified) unless doing so is consistent with all of the conditional header fields in the request."

Entity Tags and Last-Modified Times are meta-information used to find out whether a cache entry is an equivalent copy of an entity. The description in the RFC is ambiguous, but it indicates that priority should be given to the If-Match, If-None-Match, and If-Range header fields.

The first tests showed that the Apache developers shared our interpretation:

Test Id	Header 1 satisfied?	Header 2 satisfied?	Response Status
1	If-Match = true	If-Modified-Since = true	OK (200)
2	If-Match = true	If-Modified-Since = false	Not Modified (304)
3	If-Match = false	If-Modified-Since = true	Precondition Fail (412)
4	If-Match = false	If-Modified-Since = false	Precondition Fail (412)
5	If-Match = false	If-Unmodified-Since = true	Precondition Fail (412)
6	If-Match = false	If-Unmodified-Since = false	Precondition Fail (412)
7	If-None-Match = false	If-Unmodified-Since = false	Precondition Fail (412)

Note that the test cases 3-7 respond with code 412 following the interpretation that the response of the Match header has higher priority. However, the following test cases suddenly deviate from this pattern:

Test Id	Header 1 satisfied?	Header 2 satisfied?	Response Status
8	If-None-Match = false	If-None-Match = false	Not Modified (304)
9	If-None-Match = false	If-None-Match = false	Not Modified (304)

This is a rather unexpected response of Apache which does not seem to be consistent with the If-Match cases.

To our present knowledge this is the first work on generating test purposes via specification mutation. Especially the application to OSS is new. It shows that even for such popular and frequently used applications like the Apache web-server, inspection alone is not sufficient. However, this is not an argument to reject OSS, but emphasizes the need for additional quality assurance techniques, when used in critical applications.

**References**

- [1] Danish Board of Technology. Open-source software in e-government – Analysis and Recommendations Drawn up by a Working Group under the Danish Board of Technology, October 2002.
- [2] Bernhard K. Aichernig and Carlo Corrales Delgado. From Faults via Test Purposes to Test Cases: on the Fault-based Testing of Concurrent Systems, in proceedings of FASE 2006, Fundamental Approaches to Software Engineering, Vienna (Austria), March 27-29, 2006.