Codmon: A multi-platform modular test environment.

Berend van Veenendaal

March 2, 2014

TODO:Abstract

Preface

TODO: Preface,acknowledgements

Contents

1 Introduction

This chapter will give an introduction about the Codmon 2.0 project by giving a brief decription of background of my research and of the previous version of the Codmon project. It also describes the structure of the reminder of this thesis.

1.1 Background

In times when software projects become more and more complex, testing of this software becomes more and more important. Many software related problems are caused by lack of testing of the software [6]. One of the challenges of software engineering is to make sure that the software behaves in the same way on different platforms. Even when software is written in such a way that it can run on multiple platforms, there are still issues that must be dealt with, before one is able to run and test the software. Think about the configuration of the test environment or finding and installing all the prerequisite libraries etc etc.

1.2 Problem indication

Now days there are numerous test frameworks and test environments available. For example there is *Junit*[3] for Java-unit testing and *NUnit*[4] for C#-unit testing. There are also different environments like Hudson[1], [5], Jenkins[2] which can build a project and run a series of (unit) tests against this project. All of the frameworks and environments have both their advantages and disadvantages. One of the advantages of unit testing is that a software developer easily can add new *functinal* unit tests. One of the disadvantages is that standard unit testing ignores non-funtional tests like performance testing and the deployment of the software. Jenkins and Hudson,like Unit tests, also have their disadvantages. One of them is, althoug they both run on different platforms, in their usage they are not realy platform independent. For example, If you want to make a connection form Hudson or Jenkins to a remote machine you do this by executing a shell script to be able to do this, a user must know in advance on which platform this script has to run. With this in mind you can see that, althoug it is possible to connect to different machines it is not a 100% platform independent environment.

1.3 Problem statement

The test frameworks and test environments mentioned in section 1.2 can be criticized on one or more aspects. What we are looking for is infact, a combination of the positive aspects of the described frameworks and environments, without getting the undesirable aspects for free. So the big question is, is it possible to design a multi-platform, modular test environment? If the answer to this question is yes, if we manage to design such a test environment, is it also possible to design is in such a way that is user friendly and maintainable? To be able to answer the second question we first have to define what is meant with *"user friendly"* in this thesis, with user friendly we

mean that it must be possible to add easily both new test cases and software under test to the test environment. In other words a user must be able to add both new test cases and software to the test environment without nowing anything about the internal mechanisms of the test environment.

This thesis describes a multi-platform, user friendly modular test environment called Codmon 2.0. The Codmon 2.0 project provides users with a set of virtual machines, in which Codmon 2.0 is aleady installed and preconfigured. The purpose of the virtual machines is to make it as user friendly as possible. By doing it this way the only things a Codmon 2.0 user has to do are 1) add their project to the init.xml file. ant add the tests to a so calles wrapper file. Both of these files will be discussed in more detail in sections 3.2.1 and 3.2.2 .

1.4 Thesis outline

Section 2 first describes the current Codmon framework. It starts with telling why the original Codmon was built. Then it continious with a description of the design of Codmon.

2 Codmon

Before we start describing the Codmon 2.0 we start with a description of the current Codmon framework. We first describe How Codmon works and why it is not good enough for the purposes mentioned in section 1.3.

2.1 Codmon Design

2.2 Codmon problems

3 Codmon 2.0

3.1 the road to Codmon 2.0

//TODO: Think of better subsection title!! //TODO: Explain ideas and road to solution

3.2 Codmon 2.0

//TODO: adapt Codmon 2.0 to new project name. //TODO: 2) Explain Solution, including why it's different then existing solutions (compare with solutions described in the problem indication)

3.2.1 init

3.2.2 wrapper

4 Conclusion

4.1 conclsuion

TODO: give answers to the questions from section Problemstatement TODO: Discus related future work

References

[1] "Hudson Documentation". http://www.hudson-ci.org/docs/.

[2] "Jenkins Documentation". http://jenkins-ci.org/.

[3] "Junit". http://junit.org/.

[4] "Nunit". http://nunit.org/.

[5] "Winston Prakash". Introducing hudson.

[6] "Masato Shinagawa" "Toshiaki Kurkowa". Technical trends and challenges of software testing. *Science and technology trends*, 29, 2008.