Here goes the front page

Presentasjon av resultat fra automatisk testing av software

Jon Arne & Mats

February 21, 2016

The title page should include the title of your thesis, author, date/year, field of study, and name of institution (UiS and collaborating company/institution, if any). Remember to fill out the form "Title page – Master's Thesis" and insert this as page 1.

1 Abstract

The abstract must be understandable independently of the thesis itself. It gives a brief presentation of the problem(s) and the work that has been carried out. Main results and important conclusions should also be included. A good deal of work ought to be spent on writing a good abstract because this is what most people will read. The abstract should be short.

2 Acknowledgments

If you wish to thank institutions and/or persons who have been of great help during the project, this may be done in the acknowledgements. You may use the first- 12 Student guide for bachelor's and master's thesis Faculty of Science and Technology Decision made of the Dean June 25th 2013 person mode ("I") in this section. The use of first person is normally avoided in the actual report.

Contents

This shows the chapters and subchapters with page numbers. If there are symbols and abbreviations they can be listed after the table of contents.

3 Introduction

The introduction could consist of several sections. First comes a short presentation of the background for the thesis, e.g. why it is important to examine this problem. Furthermore, you should describe what the thesis is about, what has been done and how the report is built up.

Testing is used in every aspect of programming. It is the key way to achieve stable and working programs. As tests are run for every attempted solution, their efficiency is vital to increase productivity of workers. Not only are tests useful in software development, they are pivotal in making sure components that are already deployed, works as intended.

In larger systems, one component's fault can make delays and unnecessary downtime occur. Therefore it is important to identify a fault as fast and efficiently as possible. If a scheduled test reports errors, it should be intuitive for a programmer to find out where the underlying fault is. If a programmer spends more time going through logs and reports than necessary, the test process is inefficient and faulty.

Currently it's simple to recognize if everything is as it should, but in the event of a test failure, log-files often turn out thousands of lines long and result files hundreds, making precious time wasted scrolling through and searching for what exactly went wrong.

The thesis will try to define what a good result design is, and discuss what is necessary to recreate a better design based on already existing result files. It will discuss different methods of gathering test data, whether the data should come from the source, or be parsed from reports already generated.

There are several ways to present these reports, this thesis will focus on two of them, in PDF, and HTML. They are both powerful ways to gather lots of information into one, consistent file. Both having different pros and cons. They will both be used in order to give a flexible experience.

The Python(2.7) language will be used almost exclusively. The library support is extensive, which gives many alternatives and ways to solve the tasks. Python is used at ABB, and therefore the script/program will easily be compatible and give simple implementation in their test-framework. In league with Python, is Visual Studio. The thesis will discover how the synergy is between those two, and how they work together with the Team Foundation Server.

This thesis is written in English, and there are several reasons to this. First, because the code at ABB is written in English, it makes sense that the code produced along with this thesis also is written in English. Also, writing about programming involves many definitions and expressions not easily described in Norwegian. Computer Science in general is best suited for English, therefore it makes sense to write both code and thesis in English.

3.1 Subsection name about how this is built up but a more fancy word

Chapter 4 explains underlying concepts in which the thesis is built upon. The Python language, Visual Studio and the Team Foundation Server, how this is integrated at ABB, and how the current report generating solution works.

Chapter 5 discusses the designs made through the process. How the HTML was made, and the PDF. Also a possibility to use Zip to gather all the files neatly.

Chapter 6 focuses on the implementation of said solution. The decisions made throughout the project in cooperation with ABB. And how flexibility is an important factor.

Chapter 7 consists of results that has been made. How decisions have affected the process of the program. Also evaluating what could have been done better or differently.

Chapter 8 finally concludes the thesis, wraps it up, and discusses what has been learned.

4 Background

Description of the work that has already been done in this field. Normally one evaluates alternative methods and components/equipment and argues for the choices one has made. This section also outlines the theories, methods, models, equations, etc. that are relevant to the thesis. Remember to tell the reader of the paper where you found the information. Write down the sources both in the text and in the reference list.

This section will represent the foundations we have to work on. These are all important to consider, since choices made later, will be affected by what we have to work on beforehand.

4.1 Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics [1]. It is widely used, highly flexible, and fully modifiable. This means that there are several solutions to solve problems not only in python, but in the extensive range of external libraries available. Considering HTML parsing alone, there are several alternatives to chose from. All with varying functionality and efficiency. It will later be considered whether an external HTML parser is meaningful to implement, or if the default python parser does the job well enough.

An important thing to consider, is the flexibility of adding python code to a large framework. This will be important when the parser is implemented fully into ABB's systems. This also means useful information can be added to the HTML reports generated without much complication.

Python is available to use in Microsoft's Visual Studio. Visual Studio's complete suite of IDE, a Test Center and continuous integration will be considered further on.

4.2 Visual Studio

4.2.1 Team Foundation Server

4.3 Current solution & ABB

At ABB, tests are run at different times and intervals. Tests are run in parallel, which makes the test process vary in the amount of time they take. Tests are presented in Microsoft Test Center, which lists a summary test file, and makes it possible to navigate through the test files. Given that a test has failed, one can browse through these files, and find the corresponding HTML, and through to a log file.

5 Design

Good design makes for an easy and intuitive experience. Considering good design, one must also examine who the end-user is. In this case, the end-user is a highly competent developer who knows what to look for, and where to find it. A main goal can therefore be made: for the end-user to view information in a way which is easily accessible and intuitive.

While the end result has some requirements, one of these, "To display the result in an easy-to-read structure" is of relative importance.

Therefore we decided to create multiple designs in multiple filetypes and discuss pros and cons.

Test results must be quickly understandable while still providing useful deep level information. The design has to pull the readers attention to what he might find important, while not being annoying.

5.1 HTML

HTML's markup language is simple to work with, and produces quick results for fast viewing. Starting with several designs, the feedback was immediately that there was too much text. A more simplified solution with the looks of the test center already in use was sought after. A summary at the top would show the result of the test as a whole, while the details follow further down the page. A pie chart shows a visual representation of the test results, while a large symbol represents wheter the test has failed or passed.

The details here consists of all the tests from the whole testing process. In order to display these in a convenient fashion, every tests' result is grouped together until a new test result is put into the table. This way, interesting test results gets their own table entry which is easily available and stands out from the rest.

Inside each test entry, a new table with subtests appears. These are named the subtest's result. Clicking a subtest will reveal a console log, where outputs from the log appears at the according timestamp.

It is important to consider what is important and relevant for the end-user here. Not every passed test needs to show the details as to why it passed, it simply only needs to be read as passed. However, further down in the tables of tests, the user is obviously more interested in details. Therefore more details emerge the lower the user go. Ending up at the logical endpoint of the log file.

5.2 PDF

While more advanced to produce than PDF. The viewing experience can end up being more pleasant. A PDF document supports all the functions needed for the design. Using ReportLab, producing such a document in python is a manageable task.

5.3 Zip

Considering a flexible output, one may wish to have a combination of results stored. To deliver this, the files can be zipped together into one file containing all relevant data.

6 Implementation

The process of implementing our software with ABB.

The generation of the test report is called by a method call. This way implementation and testing of functionality can be tested at site. Thus comes a goal of making software that can be used both at home for testing and development, and at ABB to be implemented and further tested.

An early in-progress software was sent to be implemented to test basic functionality. It is important to get feedback for what works and what doesn't. The early product needed some fine tuning in order to work at ABB. Furthermore, the program was too dependent on the output. The gathering of data was done at the same time as the HTML file was written. A better solution to this is to divide the process into parts, where one method gathers data, and another displays it.

6.1 Integration with ABB

6.2 Flexibility

7 Results

All results should be presented. What was the outcome of the experiments, analyses, literature search and interviews? What kind of method or model did you arrive at? Calculations and estimates (if any) of uncertainty should be included at this point. Use tables (with text above) and figures (with text below). In the text all tables must be referred to with necessary explanations. It must appear logical to the reader why these tables/figures have been included. Reference to formulas, models and literature should be made in the theory section. It is often more appropriate to discuss and comment on the result(s) as you move along. In that case, the results and discussion should be dealt with in the same chapter.

8 Discussion

This is where you evaluate and interpret your results. Compare your findings to other available results. Discuss possible sources of error and the implications of these. A good thesis is recognized by a reflective discussion. The discussion must focus on significant results and observations. Avoid making the impression that you have solved every single detail. Be honest – do not try to cover up errors and simplifications that you have made on the way and later found to be unfortunate. Explain instead your choice of simplification and comment on it.

9 References

To avoid all suspicion of cheating by copying, it is important to quote references correctly. This also applies to the internet when it is used as a source.

References

[1] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *What is Python? Executive Summary*. Available https://www.python.org/doc/essays/blurb/.