UNIVERSITY *of* LIMERICK

OLLSCOIL LUIMNIGH

University of Limerick

CS4157 Software Quality

The Role & Importance of Testing in Facilitating the Development of Quality Software

Niall Dillane

CSIS

Submitted to

Dr. Valentine Casey
CSIS

1 May 2020

Abstract

Testing is an often neglected stage of the software development process, garnering less praise and compensation for its workers. Throughout this paper, we will research and demonstrate the various aspects of testing, as well as its importance in developing high quality systems.

Contents

1 Introduction

Software Testing is the investigation of software for the purpose of finding defects and providing feedback on the quality of the product (Kaner 2006). This typically involves running the program — either manually or automatically with some kind of script — and providing input with the intent of finding defects (colloquially known as bugs), or rather proving that there are no bugs. Of course, it is impossible to prove that a piece of software is entirely bug free (Pan 1999), at least if it has any degree of complexity, but testing aims to provide a reasonably complete picture of the state of the software and an avenue to reducing defects.

Testing has become more prominent in recent years (Tuteja et al. 2012), but I believe it still does not get the credit it deserves. Testing is a very broad area, which is key at each stage of the software development process: from specification, to implementation, to maintenance. As it consumes 40-50% of development efforts (Tuteja et al. 2012), it warrants significant consideration and planning from the beginning, so as to minimise costs and time needed later on.

2 Dynamic Analysis in Verification and Validation

Research and discuss the role and importance of dynamic analysis in the verification and validation of software and the essential part it plays in facilitating quality.

There are two different types of analysis in software testing: those are static and dynamic analysis.

Static analysis simply involves analysing a piece of software without actually running it (Wichmann et al. 1995). This can entail human review of code in a text editor, tools used to provide statistics e.g. lines of code per file, or the use of a compiler. This can be useful in certain circumstances, such as finding security vulnerabilities in a codebase (Livshits 2006) and is naturally quicker than walking through a program, but generally it is limited from a "user experience" point of

view.

Dynamic analysis is based on system execution, often using tools to automate the process (Ghahrai 2018). There are various stages, from individual unit tests of small portions of the system, to integrating these units and the system overall. Dynamic testing provides a more complete picture of the state of the system, and is a more "real life use" approach. This can be time consuming, and so more and more automation is sought out, but one must be wary of the false sense of security these tools can provide.

Verification and Validation are related concepts in software testing. In whole, they describe the process of: identifying if the system has satisfied the specification provided (verification); and determining if that specification is what the customer really wants or needs (validation). In short, verification answers the question: "Are we building the system right?", and validation answers "Are we building the right system?" (**?**).

Dynamic verification involves, as outlined previously, executing the code. At this point, developers wish to identify if the system has been built to specification, conducting a review of the program in various ways. These can be testing in the small — checking if inidividual functions perform as expected — up to testing the system as a whole, by executing modules and checking that they interact as expected. Non-functional requirements (discussed more in Section 3) may also be tested, such as stress-testing a server to see how many users it can handle.

Dynamic validation is an even higher level of testing (acceptance testing), wherein the development team and external stakeholders walk through the product and use it as a user would. This is black box testing (see Section 6), as the testers are performing these actions without looking at the internal workings of the system. A purely surface view.

It's important because.

3 Functional and Non-Functional Requirements

Clearly outline and define what Functional and Non-Functional Software Requirements are and in your discussion compare and contrast the differences between them

3.1 How to Test

Research and discuss how Functional Software Requirements and Non- Functional requirements can be successfully tested and define the differences in approach required by each

4 Software Testing Process

Outline and discuss the key stages in the software testing process

4.1 Unit Testing

4.2 Integration Testing

4.3 Regression Testing

4.4 System Testing

4.5 Acceptance Testing

5 Effective Test Strategy

Research and define the principles of an effective test strategy.

5.1 Documented Test Procedures

Consider and discuss the role and importance of having documented test procedures in place to facilitate effective testing

5.2 Test Plan

Research and outline the purpose, importance and content of a test plan

6 White Box and Black Box Testing

Define and discuss the importance of utilizing both white box and black box testing

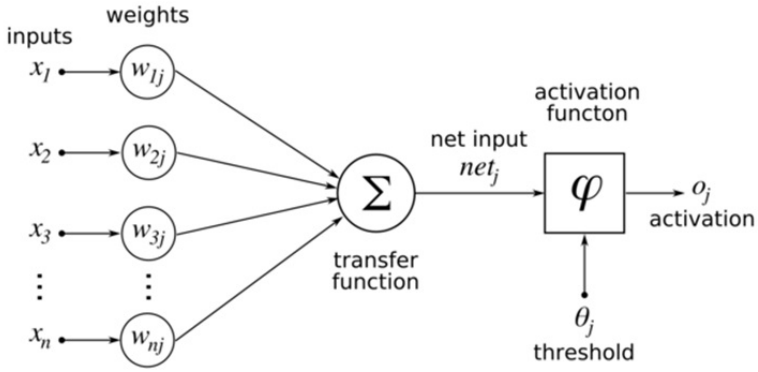
7 Sample Table

Table 1: Comparison p values

Attribute	***p-value***	**Significant**
Model A	0.0521	N
Model B	0.6171	N
Model C	<0.00001	Y

7.0.1 Sample Figure

Figure 1: Perceptron (Artificial Neural Network)



References

Ghahrai, A. (2018), 'Static analysis vs dynamic analysis in software testing', *Testing Excellence* .

Kaner, C. (2006), Exploratory testing, *in* 'Quality assurance institute worldwide annual software testing conference'.

Livshits, B. (2006), *Improving software security with precise static and runtime analysis*, Vol. 67.

Pan, J. (1999), 'Software testing', *Dependable Embedded Systems* **5**, 2006.

Tuteja, M., Dubey, G. et al. (2012), 'A research study on importance of testing and quality assurance in software development life cycle (sdlc) models', *International Journal of Soft Computing and Engineering (IJSCE)* **2**(3), 251–257.

Wichmann, B., Canning, A., Clutterbuck, D., Winsborrow, L., Ward, N. and Marsh, D. (1995), 'Industrial perspective on static analysis', *Software Engineering Journal* **10**(2), 69–75.