#04
MAY 2018

S.Q.mag

THE VOICE OF SOFTWARE QUALITY

INDUSTRY 4.0

Industrial Designer Raquel Ariza on the developments of the Industry 4.0 in Latin American Countries

ARTICLE

A CLASH OF MODELS

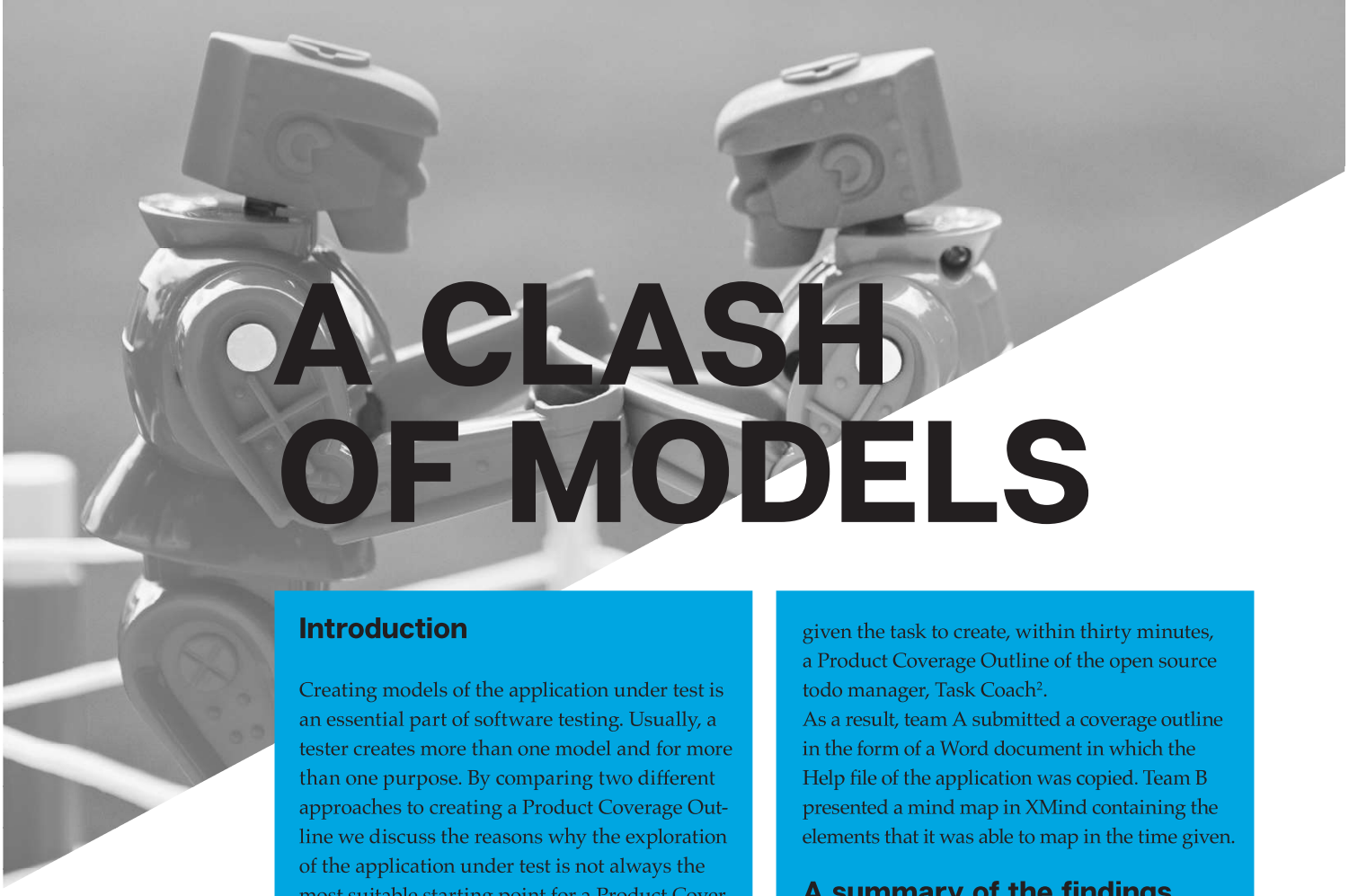
Test consultant Joris Meerts & testing practitioner Ruud Cox on creating Product Coverage Outlines

ARTICLE

THE CHANGE NEEDS TO BE BOTTOM UP AND TOP DOWN

Leanne Howard on the buzzword "Scrum" and how teams can work with it

INTERVIEW



A CLASH OF MODELS

Introduction

Creating models of the application under test is an essential part of software testing. Usually, a tester creates more than one model and for more than one purpose. By comparing two different approaches to creating a Product Coverage Outline we discuss the reasons why the exploration of the application under test is not always the most suitable starting point for a Product Coverage Outline. In our opinion some testers have a predisposition toward learning and understanding the application. They habitually apply a standard tool (a mind map) and a standard structure (the SFDIPOT tree structure) without questioning the feasibility of that approach.

We compare the outcomes of an exercise in which two teams participated and found that surprisingly different results were derived from the same mission, using the same heuristic on the same application. Both teams were asked to create a Product Coverage Outline; a model of the application which identifies aspects of the application that could be tested. The Product Elements heuristic – as mentioned in the Heuristic Test Strategy Model by James Bach[1] – puts these aspects into seven categories; Structure, Function, Data, Interface, Platform, Operation and Time (SFDIPOT).

A short description of the exercise

Our findings are based on an exercise in which the authors participated during a morning at the office of our employer, Improve Quality Services. Two teams (A and B) of two people each were given the task to create, within thirty minutes, a Product Coverage Outline of the open source todo manager, Task Coach[2].
As a result, team A submitted a coverage outline in the form of a Word document in which the Help file of the application was copied. Team B presented a mind map in XMind containing the elements that it was able to map in the time given.

A summary of the findings

Before we describe the particularities of the approaches of the teams, we present a summary of the findings, each of which lead to an impediment in the gathering of essential information.

Focus on engaging with the application. Team B, while building the mind map, was able to map only a small number of the product elements. Mapping the application by exploring it and at the same time recording its properties in the structure of the mind map turned out to be a time consuming task.

Restructuring the product elements. Team B started with a hierarchical structure with a branch for each category in the SFDIPOT mnemonic. This new structure forced the team to take the application apart into its constituent elements and to make decisions about grouping these elements in the right categories.

Predisposition to learning and understanding. For team B it felt that the specific characteristics of the mind map drew them towards learning and understanding the application and attempting to gain a deeper understanding of it, instead of creating the PCO.

1 Bach, J. (2015, May 20). "Heuristic Test Strategy Model" http://www.satisfice.com/tools/htsm.pdf
2 "Task Coach." http://taskcoach.org/

A more detailed view of the approaches

The first striking outcome was that within thirty minutes team A was able to produce a coverage outline – in the form of a Word document – in which it would be possible to locate a large set of product elements. Team B, while building a mind map, had been able to identify a substantially smaller set of product elements.

Focus on engaging with the application

Team A started its session engaging with the application to gather elements. A few minutes into the session it was noticed that the application contained a large Help file. After a quick assessment the team learned that it provided a useful overview of the application and that many of the product elements in various categories could be identified. The team then decided to copy the – lightly edited – contents of the entire file to a Word document as a first and good enough version of a coverage outline.

Team B also started its session engaging with the application trying to understand the various elements before mapping them. But the application contained many functions, nested deeply within the different menus, the contents of the screens, the Help file and even on the website of the application. So mapping the identified elements turned out to be a time consuming task which resulted in a map with a small set of elements.

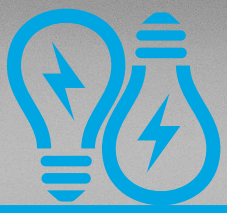
Restructuring the product elements

Team A skimmed the Help file and decided to copy the text in its entirety to a Word document. While skimming the text, the SFDIPOT heuristic was mentally used as a tool to discover product elements. It was also noticed that the Help file contained useful structures such as a table of contents and that the product elements were grouped by feature. The team decided to copy the structure of the Help file instead of creating a new structure. At the end of the exercise, team A suggested to add even more structure by highlighting each category of product elements with a specific color in order to improve the usability of the coverage outline.

Team B started with a mind map with a branch for each category in the SFDIPOT mnemonic. The Heuristic Test Strategy Model presents the product elements as a hierarchy, a list with bulleted sub lists. This hierarchical structure easily maps to the structure enforced by the common mind map tools such as FreeMind and XMind. They have a built in hierarchical structure of one central node and branches. This new structure however, forces the tester to take the application apart and to make decisions about grouping the product elements in the right category.

Joris Meerts,
is a test consultant for Improve Quality Services in the Netherlands. He is a passionate software tester who likes to explore the boundaries of the craft, looking for ways to do testing better. Joris is the author of the History of Software Testing, which can be found on his website Testing References (www.testingreferences.com). He likes to speak and publish on the topics of testing history, the practice of software testing and the use of test automation. He is a member of the Dutch Exploratory Workshop on Testing.

Ruud Cox,
is a testing practitioner and teacher with a special interest in context analysis and modeling. In 1992 he started out as a software developer in the embedded systems industry. He has worked in domains like consumer electronics, semiconductors, lighting, and healthcare. He became a tester because he got interested in the problems which result from what is perceived and what exists. He currently works as a test consultant for Improve Quality Services, a provider of consultancy and training. He speaks and teaches at national and international conferences. He is one of the founders of the Dutch Exploratory Workshop on Testing (DEWT) which is a group of passionate testers, trying to change the world. He is on twitter at @ruudcox.



Predisposition to learning and understanding
Which approach a team (or a person) applies is not always a conscious choice. Team A focussed on gathering elements. This approach was driven by a personal bias to create an overview, to see the big picture, before making a decision on which area of the application the testing effort should be spent. Team B's approach was a predisposition of the tester to focus on learning and understanding and to apply a standard tool (a mind map) and a standard structure (the SFDIPOT tree structure) without questioning the feasibility of that approach.

Discussion

A hefty debate followed the presentation of team A's coverage outline. Team B was surprised by the lengthy Word document. How could such a huge document be useful as a coverage outline? And how would the tester be able to understand the application without having explored it? Isn't learning about and making sense of the application what testing is all about? Of course, eventually the application has to be tested but if we examine what both teams produced in relation to the mission of the thirty minutes session, both teams produced what was asked for; a Product Coverage Outline.

One of the reasons why both teams produced such very different coverage outlines was the absence of a specific purpose. Team A asked questions about the purpose of the coverage outline but these questions were dismissed by the facilitator. Without a clear purpose, any set of product elements structured in whatever way will do.

A Product Coverage Outline can have multiple purposes; it can be used to learn about the application or to help a tester decide what to test. It is useful as a source for new test ideas and it can serve as a basis for test reporting. Team A's Product Coverage Outline provided a comprehensive set of elements. Team B's Product Coverage Outline only had a small number of elements.
Both sets of elements could have been used as a basis for further test planning but Team A's outline provided more information on what remains to be tested than Team B's outline. Team B's mind map created a two dimensional structure. In this map, it is easy to point and focus on a specific element – for example the 'Save' functionality in the File menu. It is easy to see its place in the larger picture and its relation to other elements which is an enabler for triggering new test ideas.

One objective of testing is to learn about the application under test. Because team A's focus was on gathering elements, there was almost no understanding of the (elements of the) application. Team B skipped the gathering of information, which resulted in a small set of elements. But because it focussed more on the processing of these elements it was able to describe to some degree what it had gathered.

We'd like to conclude with the remark that the result of both teams should not be considered final. It simply was what both teams were able to produce within the given amount of time. It is reasonable to assume that both models will evolve over time when the application is tested and new information becomes available. It even might be that both models will evolve in a set of coverage outlines, each with a specific purpose and structure. ▌