

Aalto University  
School of Science  
Master's Programme in Computer, Communication and Information Sciences

Jani Saarijärvi

# Improving User Experience through Consistency

Master's Thesis  
Espoo, March 27, 2017

Supervisors: Professor Marko Nieminen  
Advisor: Anri Friman M.Sc., University of Helsinki  
Markus Laakso M.Sc. (Tech), Helsinki University of Technology

|  |  |                    |
|--|--|--------------------|
| <b>Author:</b>   | Jani Saarijärvi  |                    |
| <b>Title:</b>  | Improving User Experience through Consistency  |                    |
| <b>Date:</b>   | March 27, 2017   | <b>Pages:</b> 99   |
| <b>Major:</b>  | Usability and User Interfaces  | <b>Code:</b> T-121 |
| <b>Supervisors:</b>  | Professor Marko Nieminen   |                    |
| <b>Advisor:</b>  | Anri Friman M.Sc.<br>Markus Laakso M.Sc. (Tech)  |                    |
| <p>This thesis studies how the concept of consistency applies to user experience, how consistency of user experience could be measured, and whether a consistent user experience is measurably better than an inconsistent one. Consistent interfaces offer many potential benefits, such as being faster to learn, more efficient to use, and knowledge of consistent interfaces being transferable to other similarly consistent interfaces.</p> <p>Consistency of user experience is defined in this thesis by abstracting it into dimensions of internal and external consistency over conceptual, communicational and physical components. Pragmatic and hedonic aspects of user experience are used in measuring the effect of consistency on user experience.</p> <p>An empirical study was performed on a web frontend that was being refactored to a new platform with the plan to concurrently add improvements. The focus was to fix inconsistencies in the frontend and gauge whether that improves user experience and usability.</p> <p>Two usability evaluations (both N=13) were conducted: one for the original version of the frontend, and one for a frontend where some of the consistency-related findings of the first evaluation had been addressed. Talk-aloud process and observation were used to collect qualitative findings during the task-driven evaluation sessions, while quantitative user experience and usability data was measured with post-evaluation AttrakDiff and SUS questionnaires.</p> <p>The results, however, did not show any clear change. Qualitative findings from the second evaluation showed positive development, but they conflicted with negative but statistically insignificant quantitative findings from the same evaluation. Possible explanations for this turn of events are presented, including the possibility that the evaluation participants graded the frontend as relatively "too good" on the first evaluation.</p> |  |                    |
| <b>Keywords:</b>   | user experience, usability, consistency, internal consistency, external consistency, hedonic, pragmatic, attrakdiff, sus, system usability scale |                    |
| <b>Language:</b>   | English  |                    |

Aalto-yliopisto

Perustieteiden korkeakoulu

 Master's Programme in Computer, Communication and  
 Information Sciences

 DIPLOMITYÖN  
 TIIVISTELMÄ

|  |   |                   |       |
|--|---|-------------------|-------|
| <b>Tekijä:</b>   | Jani Saarijärvi   |                   |       |
| <b>Työn nimi:</b>  | Käyttöliittymän konsistenssi käyttäjäkokemuksen kohennuksessa   |                   |       |
| <b>Päiväys:</b>  | 27. maaliskuuta 2017  | <b>Sivumäärä:</b> | 99    |
| <b>Pääaine:</b>  | Käytettävyys ja käyttöliittymät   | <b>Koodi:</b>     | T-121 |
| <b>Valvojat:</b>   | Professori Marko Nieminen   |                   |       |
| <b>Ohjaaja:</b>  | Filosofian maisteri Anri Friman<br>Diplomi-insinööri Markus Laakso  |                   |       |
| <p>Tämä diplomityö tutkii konsistenssin - eli johdonmukaisuuden tai yhdenmukaisuuden - käyttöä käyttäjäkokemuksen yhteydessä, kuinka tätä konsistenssia voidaan mitata, ja onko konsistentti käyttöliittymä epäkonsistenttia parempi. Konsistentit käyttöliittymät tarjoavat monia potentiaalisia etuja: niitä on muun muassa tehokkaampi käyttää, niitä on nopeampi oppia, ja tämä opittu tietämys on siirrettävissä muihin vastaavasti konsistentteihin käyttöliittymiin.</p> <p>Käyttäjäkokemuksen konsistenssi määritellään tässä työssä sisäisen ja ulkoisen konsistenssin dimensioiden sekä käsitteellisen, viestinnällisen sekä fyysisen osamäärittelyn avulla. Käyttäjäkokemuksen käytännöllisiä ja hedonisia aspekteja käytetään mitattaessa konsistenssin vaikutusta käyttäjäkokemukseen.</p> <p>Tutkimuksen empiirinen osa suoritettiin web-käyttöliittymälle jota oltiin uudistamassa. Tavoitteena oli korjata epäjohdonmukaisuuksia ja mitata paransivatko nämä toimet käyttäjäkokemusta ja käytettävyyttä.</p> <p>Käyttöliittymälle suoritettiin kaksi käytettävyysarviota (molemmissa N=13): alkuperäiselle versiolle, sekä versiolle, jossa ensimmäisen arviokierroksen konsistenssiin liittyviä löydöksiä oli paranneltu. Laadullista dataa kerättiin tehtäväpohjaisissa arvioissa äänenajattelu-metodilla ja havainnoinnalla. Määrällistä dataa käyttäjäkokemuksesta ja käytettävyydestä kerättiin AttrakDiff- ja SUS-menetelmillä.</p> <p>Lopputuloksena ei kuitenkaan saatu selkeää eroa. Toisen arviokierroksen laadullinen data osoitti positiivista kehitystä järjestelmän parannuksissa, mutta tämä data oli ristiriidassa negatiivisten, mutta tilastollisesti ei-merkitsevien määrällisten löydösten kanssa. Mahdollisia selityksiä tälle annetaan työssä, päällimmäisenä mahdollisuus että osallistujat arvioivat käyttöliittymän suhteellisesti ”liian hyväksi” ensimmäisessä arviokierroksessa.</p> |   |                   |       |
| <b>Asiasanat:</b>  | käyttäjäkokemus, käytettävyys, johdonmukaisuus, konsistenttius, konsistenssi, sisäinen konsistenssi, ulkoinen konsistenssi, hedoninen, pragmaattinen, attrakdiff, sus, system usability scale |                   |       |
| <b>Kieli:</b>  | Englanti  |                   |       |

# Acknowledgements

I studied for a while and now I'm done.

As much as I'd like to leave it at that, several people were instrumental in making this thesis possible and they deserve my utmost gratitude. Above all I would like to thank Anri Friman for tirelessly advising and encouraging me throughout this thesis process. I also wish to thank my supervisor professor Marko Nieminen for his guidance and expertise. My gratitude also to my second thesis advisor Markus Laakso for his input and support.

A big thank-you also to all my colleagues at work who were involved with this thesis in any capacity, particularly the brave ones who volunteered as usability test participants. Extra warm thanks to my friends and family for their support throughout my studies and this thesis process.

And finally, thank you to Finnish governments prior to year 2003 for their pro-education stances, enabling in their part the creation of this thesis.

Espoo, March 27, 2017

Jani Saarijärvi

# Abbreviations and Acronyms

|                     |   |
|---------------------|---|
| AJAX                | Asynchronous JavaScript and XML: A set of web development techniques used to asynchronously retrieve data from server to client (web browser) without stopping, freezing or reloading the user interface.                     |
| ASP.NET             | A Microsoft-developed server-side web application framework that can be used to create dynamic web applications.  |
| AUIT System         | The "old" AUIT (Automated Unit and Integration Testing) system was a cloud-based environment with a rudimentary web frontend, used for executing automated tests at Varian. It is detailed further in section 3.2 on page 31. |
| Backend             | The backend of a web application runs on the server machine and usually handles some or all of the actual business logic of the application, and is used to store and retrieve permanent data related to the application.     |
| Client, client-side | In web development the "client" is generally the user's web browser. When code is executed "client-side" it usually refers to JavaScript code being run on user's browser. See also: frontend.                                |
| DOM                 | Document Object Model: An internal hierarchical model of a web page that the user's web browser constructs and manipulates.   |
| Execution Framework | The "new" cloud-based environment for executing automated tests at Varian, it superseded the old AUIT system. Introduced a dynamic ASP.NET web frontend.  |

|                   |  |
|-------------------|--|
| FAST              | Framework for Automated Software Testing. An umbrella term for automated software testing tools at Varian. Execution Framework is a subset of FAST.  |
| Frontend          | The frontend of a web application is the graphical user interface presented to user on their web browser. Modern web frontends are combinations of HTML markup for structuring data, CSS markup for styling the page, and JavaScript code for enabling dynamic interaction on the page. See also: client, client-side. |
| JQuery            | JQuery (or jQuery) is a popular JavaScript library used for DOM manipulation in client-side JavaScript code.   |
| JSON              | JavaScript Object Notation: A human-readable format used for transmitting data on the web as attribute-value pairs. Modern web platforms generally use JSON in favour of XML for data transfer between server and client.  |
| ReactJS           | ReactJS (or just React) is a JavaScript library used for creating dynamic web user interfaces.   |
| Spinner           | A spinner (sometimes also referred to as throbber) is an animated graphical element used for representing an on-going progress, such as a long-running calculation operation, without conveying how much of the progress has been completed.   |
| System under test | An artefact of the Execution Framework, a system under test is a configuration object. It tells the system where to find new builds and contains information on what sort of environment settings are required by the tests that are run from this system under test.  |
| Web API           | In simple terms, a web API (Application Programming Interface) is a server interface returning data when its specified access points are called.   |
| Worker cloud      | The worker cloud is a distributed network of PCs that the Execution Framework uses to run tests in parallel.   |
| Workflow instance | An artefact of the Execution Framework, a workflow instance is a concrete, build-specific version of a <i>workflow template</i> where the actual running of tests happens.   |

Workflow template    An artefact of the Execution Framework, a workflow template is an abstract configuration object that is used to run repeatable sets of tests for new builds of a specific system under test according to a schedule. A *workflow instance* gets instantiated from a workflow template when the template's trigger conditions are matched - that is, when a new build appears into the Execution Framework system that matches the scheduling and system under test criteria configured in the template.

# Contents

|  |           |
|--|-----------|
| Abbreviations and Acronyms                                 | 5         |
| <b>1 Introduction</b>                                      | <b>11</b> |
| 1.1 Background and Motivation . . . . .                    | 12        |
| 1.2 Research Questions, Goal, and Scope . . . . .          | 14        |
| 1.3 Structure of the Thesis . . . . .                      | 15        |
| <b>2 Consistency of User Experience</b>                    | <b>16</b> |
| 2.1 Defining Consistency through User Experience . . . . . | 16        |
| 2.1.1 User Experience . . . . .                            | 16        |
| 2.1.2 User Experience and Usability . . . . .              | 17        |
| 2.1.3 Consistency . . . . .                                | 18        |
| 2.1.4 User Experience and Consistency . . . . .            | 20        |
| 2.2 Measuring User Experience and Usability . . . . .      | 21        |
| 2.2.1 AttrakDiff . . . . .                                 | 22        |
| 2.2.2 System Usability Scale . . . . .                     | 22        |
| 2.2.3 Think-aloud Process . . . . .                        | 23        |
| 2.3 A Model for Measuring Consistency . . . . .            | 23        |
| <b>3 The ASP.NET Frontend</b>                              | <b>25</b> |
| 3.1 Design . . . . .                                       | 25        |
| 3.1.1 Home Page . . . . .                                  | 25        |
| 3.1.2 Test Browser . . . . .                               | 25        |
| 3.1.3 Workflow Manager . . . . .                           | 27        |
| 3.1.4 The Workflow Instance Page . . . . .                 | 28        |
| 3.1.5 The Test Executions Page . . . . .                   | 28        |
| 3.1.6 Workflow Template Editor . . . . .                   | 30        |
| 3.1.7 Systems under Test Pages . . . . .                   | 30        |
| 3.2 Old AUIT System . . . . .                              | 31        |

|          |   |           |
|----------|---|-----------|
| <b>4</b> | <b>Evaluating the ASP.NET Frontend</b>  | <b>34</b> |
| 4.1      | Evaluation Goals . . . . .  | 34        |
| 4.2      | Participants . . . . .  | 34        |
| 4.3      | Procedure . . . . .   | 35        |
| 4.4      | Scenarios . . . . .   | 36        |
| 4.4.1    | Scenario 1: Viewing Front Page . . . . .  | 36        |
| 4.4.2    | Scenario 2: Queueing Test Runs . . . . .  | 36        |
| 4.4.3    | Scenario 3: Scheduling a Workflow . . . . .   | 37        |
| 4.4.4    | Scenario 4: Investigating a Failing Workflow . . . . .                                      | 37        |
| 4.4.5    | Changes to Scenarios . . . . .  | 38        |
| 4.5      | Collecting Findings . . . . .   | 39        |
| 4.6      | Pre-Test and Post-Test Questionnaires . . . . .   | 39        |
| <b>5</b> | <b>Results of First Evaluation</b>  | <b>40</b> |
| 5.1      | Overview . . . . .  | 40        |
| 5.2      | Quantitative Findings . . . . .   | 40        |
| 5.3      | Qualitative Findings . . . . .  | 41        |
| 5.3.1    | Home Page . . . . .   | 43        |
| 5.3.2    | Test Browser . . . . .  | 44        |
| 5.3.3    | Workflow Manager . . . . .  | 45        |
| 5.3.4    | Workflow Instance . . . . .   | 46        |
| 5.3.5    | Test Executions . . . . .   | 47        |
| 5.3.6    | Workflow Template Editor . . . . .  | 47        |
| 5.3.7    | Systems under Test . . . . .  | 49        |
| 5.3.8    | Other Findings . . . . .  | 49        |
| <b>6</b> | <b>Building the React Frontend</b>  | <b>50</b> |
| 6.1      | Architecture . . . . .  | 50        |
| 6.1.1    | React As the New Frontend Library . . . . .   | 51        |
| 6.1.2    | Focus on Workflow Pages . . . . .   | 51        |
| 6.2      | Improvements . . . . .  | 53        |
| 6.2.1    | No Overall Result Summary for Workflow Instances . . . . .                                  | 53        |
| 6.2.2    | No Result Summary Available in Workflow Manager . . . . .                                   | 55        |
| 6.2.3    | Workflow Instance Page Does Not Auto-Refresh With Latest data . . . . .                     | 57        |
| 6.2.4    | Workflow Manager Page Has No Column Titles . . . . .  | 58        |
| 6.2.5    | Workflow Manager Has Three Different Ways of Conveying Workflow Instance Progress . . . . . | 59        |
| 6.2.6    | Considerable Lag When Sorting Tables . . . . .  | 60        |
| 6.2.7    | Tables Do Not Look Sortable . . . . .   | 61        |
| 6.2.8    | Misinterpreting Auto-Refresh Spinner . . . . .  | 62        |

|           |  |           |
|-----------|--|-----------|
| 6.2.9     | Run Time for a Workflow Instance Is Displayed With Too Much Accuracy . . . . . | 63        |
| 6.2.10    | Undo Function In Table Is Confusing . . . . .                                  | 64        |
| <b>7</b>  | <b>Evaluating the React Frontend</b>   | <b>66</b> |
| 7.1       | Evaluation Goals . . . . .   | 66        |
| 7.2       | Participants . . . . .   | 66        |
| 7.3       | Procedure . . . . .  | 67        |
| 7.4       | Scenarios . . . . .  | 67        |
| 7.4.1     | Scenario 1: Current State of the System . . . . .                              | 67        |
| 7.4.2     | Scenario 2: Investigating a Failing Workflow . . . . .                         | 68        |
| 7.5       | Pre-Test and Post-Test Questionnaires . . . . .                                | 68        |
| 7.6       | Post-Test Questions . . . . .  | 68        |
| <b>8</b>  | <b>Results of Second Evaluation</b>  | <b>69</b> |
| 8.1       | Overview . . . . .   | 69        |
| 8.2       | Quantitative Findings . . . . .  | 69        |
| 8.3       | Qualitative Findings . . . . .   | 70        |
| 8.3.1     | Workflow Manager . . . . .   | 71        |
| 8.3.2     | Workflow Instance . . . . .  | 72        |
| 8.3.3     | Other Findings . . . . .   | 73        |
| 8.4       | Other Feedback . . . . .   | 74        |
| <b>9</b>  | <b>Analysis of Results</b>   | <b>75</b> |
| <b>10</b> | <b>Discussion</b>  | <b>79</b> |
| 10.1      | Reflection on Results . . . . .  | 79        |
| 10.2      | Reliability and validity . . . . .   | 82        |
| <b>11</b> | <b>Conclusion</b>  | <b>84</b> |
|           | <b>Bibliography</b>  | <b>87</b> |
| <b>A</b>  | <b>Questionnaires</b>  | <b>91</b> |
| <b>B</b>  | <b>Final React Frontend Designs</b>  | <b>97</b> |

# Chapter 1

## Introduction

In the foreword to the 2002 reprint of *Coordinating User Interfaces for Consistency* Jakob Nielsen writes:

Today, almost all large and mid-sized companies have an intranet on which employees spend a large percentage of their time accessing information and services. Much of that time is a complete waste due to inconsistent design that reduces employee productivity. World-wide, the losses are in the hundreds of billions of dollars per year. [23, p. viii]

While Nielsen does not back this statement with data, the anecdote likely rings true for anyone with experience in the corporate world. Consistency is a core concept in human-computer interaction, going as far as getting an entire item devoted to it in Nielsen's own famous set of ten usability heuristics:

*Consistency and standards:* Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions. [21]

Nevertheless, the concept of consistency in human-computer interaction is defined vaguely at best in literature, and its relation to user experience instead of just usability or user interface design is not really discussed. Furthermore, there seems to have been no attempts made to measure the effect consistency has on user experience, nor has the question been answered whether improving consistency of a system or a product would also improve its user experience.

This thesis attempts to answer these questions by defining consistency within the context of user experience and by introducing methods on how the effect of consistency on user experience can be measured. A case study is

presented, in which an internal company system is evaluated and improved based on consistency-related aspects.

## 1.1 Background and Motivation

This thesis work has been produced for Varian Medical Systems, a company in the medical device field. The Helsinki site of Varian designs *Eclipse*, a computer-assisted design (CAD) software used to design cancer treatment plans for radiotherapy treatment. Each night, as new builds of the Eclipse software are created, the software is automatically tested by over 15,000 automated unit and integration tests, run on a local cloud system.

This automated testing environment is called the *Execution Framework*. It consists of a number of core backend services, a worker cloud, and a web-based frontend for accessing Execution Framework functionality through a standard web browser. A high-level view of architecture of Execution Framework is presented in figure 1.1.

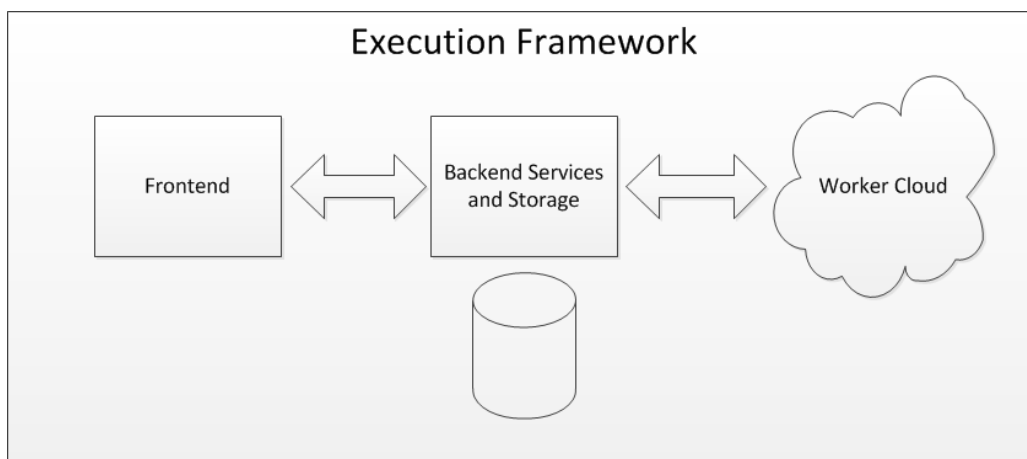


Figure 1.1: Overview of the Execution Framework architecture.

The backend services handle, among other things, discovering new builds automatically and importing them into the Execution Framework system. After builds have been discovered, they can be viewed and used for test runs through the frontend. The services also handle prioritising and queueing automated tests, and set up and execute the actual test runs in the worker cloud.

The worker cloud is a local network of both physical and virtual test machines (or "workers"). The backend services perform actual test runs by

picking a worker machine from the cloud, installing required components on the machine, running a single automated test (or a small set of tests) on the machine, reporting the end result back for the backend services, and finally cleaning up the worker machine before marking it as free for reuse.

The web frontend, or the web GUI, is a web site that is used to view builds and tests discovered in the system, schedule automated test runs with workflows, launch automated tests manually, view test results, and view the current state of test runs in the system. The frontend is built on Microsoft's ASP.NET MVC technology.

While the entire Execution Framework can be split into a "backend" - the services and the worker cloud - and a "frontend" - the web GUI frontend -, the web frontend can also be split into a backend and a frontend: the ASP.NET MVC web application running on Microsoft's IIS server is the GUI's backend, collecting data and rendering it into web pages that it can send to user's web browser. The web browser forms the GUI's frontend, using JavaScript executed in the browser for dynamic interaction.

As historical background, the Execution Framework was preceded by an older test execution environment system called AUIT (for *Automated Unit and Integration Testing*). The AUIT system is described briefly in section 3.2 on page 31 of this thesis. Later parts of this thesis describe how the Execution Framework's ASP.NET-powered frontend, the "ASP.NET frontend", was replaced by another frontend implementation using ReactJS technology, referred to as the "React frontend". The timeline of this in-house test execution system and frontend evolution is shown in figure 1.2.

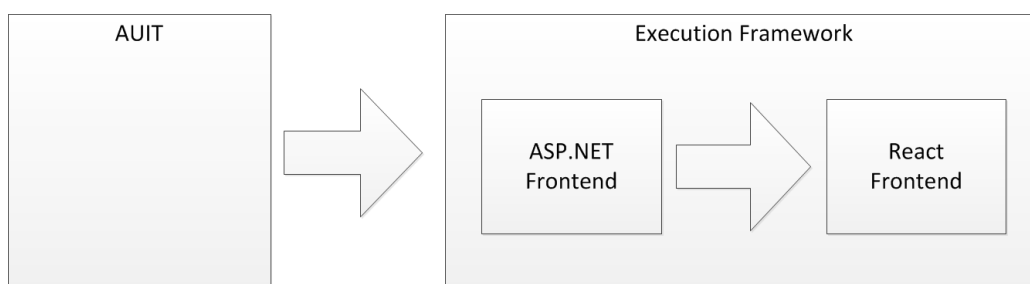


Figure 1.2: Evolution of test execution environments at Varian. The old "AUIT" system was replaced by Execution Framework. The initial ASP.NET frontend of Execution Framework was to be replaced by a new React frontend. This process is detailed in this thesis.

The subject of this thesis is this web GUI frontend of the Execution Framework. The current frontend has been built with several thousand

lines of hand-built JQuery code over two years, without using any actual pre-existing frontend framework or committing to any single style of coding practice. This has not only resulted in an inconsistent code base, but also inconsistencies in the user-facing frontend, such as some pages loading content dynamically with AJAX whereas some pages require a manual refresh, and some data tables being sortable and some not.

The hypothesis of this thesis is that these inconsistencies in the GUI also cause a worse user experience, and re-constructing the GUI by taking advantage of an existing frontend framework, and by adding further consistency improvements on top, would result in a more consistent and better user experience and usability.

## 1.2 Research Questions, Goal, and Scope

The main research question that will be studied in this thesis is:

*Is a consistent user interface better from user experience point of view than a heterogeneous one?*

While it seems logical to assume that a more consistent user interface will result in a better user experience, the real question is how much user experience is improved and, considering business realities, whether or not this improvement is worth the effort. Thus the research goal is to measure the change in user experience, and also to gauge whether this change is significant or not.

To aid in answering this question, two sub-questions are also looked into in this thesis:

1. *What is a consistent user experience?*

First, before the main research question can be answered, a definition of what is exactly meant by consistency must be reached, and its connection to user experience must be specified.

2. *How can consistency of user experience be measured?*

Second, a method for measuring consistency of user experience must be devised.

Constructing a new frontend for the existing Execution Framework system has multiple side benefits in addition to the hopeful improvement in user experience. Using a ready-made frontend framework should result in smaller code footprint overall, meaning there would be less code to maintain. A well-maintained existing framework has also been tested better, making it a more reliable pick. Blending code into an existing paradigm and using well-defined coding conventions should also make code maintenance easier compared to maintaining separate components that have each been programmed

with wildly differing logic.

Possible drawbacks of using an existing frontend framework include inflexibility, in that the framework may railroad design of user interface elements so much that it may not be easy to recreate current features from the old frontend as they are, and that the coding conventions of the chosen framework may be overly complicated and difficult for non-experts.

However, although the promise of lessened maintenance burden was a major factor in initialising the work covered in this thesis, these factors are out of scope of this study. This thesis focuses only on the changes in user experience and usability caused by the new frontend.

### 1.3 Structure of the Thesis

This thesis begins in chapter 2 with the definitions of usability, user experience, and the hedonic/pragmatic model of user experience. The concept of consistency is also introduced and defined. Benefits of consistency are explained. The power of consistency is examined through mental models. Benefits of inconsistency are also touched on. The chapter also presents methods for measuring user experience and usability, ending with a description of a method for measuring consistency using concepts and tools presented earlier in the chapter.

Next, to present a baseline for the upcoming changes, chapter 3 introduces the design of the initial "ASP.NET frontend" of Execution Framework.

Chapter 4 describes how the ASP.NET frontend was usability tested in order to discover consistency issues that would be improved. Results of this usability evaluation, both quantitative AttrakDiff and SUS numbers and qualitative findings, are presented in chapter 5.

Following this, chapter 6 shows how the old ASP.NET frontend was refactored and improved into the "new" frontend, or the "React frontend" as per the chosen web technology. Some of the findings from the first usability study are also presented here in more detail, along with information on how they relate to consistency, and how they were fixed in the React frontend.

After this, chapter 7 presents how the React frontend was usability tested. Evaluation results follow in chapter 8.

Results of both usability studies are summed and analysed in chapter 9.

The results and the entire study are discussed in chapter 10.

Finally, the thesis is concluded and possible future research avenues are presented in chapter 11.

## Chapter 2

# Consistency of User Experience

This chapter begins by defining user experience and usability and comparing these definitions. Definition and description of consistency within human-computer interaction follows. After that, methods for measuring user experience and usability are presented. Finally, a method for measuring consistency of user experience and usability with the methods presented earlier is described.

### 2.1 Defining Consistency through User Experience

This section begins by defining user experience and usability. Section 2.1.3 then introduces and defines consistency within human-computer interaction.

#### 2.1.1 User Experience

An archetypical definition of *user experience* comes from ISO standard 9241-210 which defines it as “person’s perceptions and responses resulting from the use and/or anticipated use of a product, system or service” [6].

Zimmermann [36, pp. 11-12] notes the dilemma of finding a generally accepted definition of user experience within literature of the field, and describes user experience as a holistic, all-encompassing concept that takes characteristics of the user, the product and the usage situation into account.

These broad definitions, however, make measuring user experience difficult. To make measurement feasible this paper uses a framework described by Hassenzahl [11]: a hedonic/pragmatic model of user experience, where the vast scope of user experience is reduced into hedonic and pragmatic aspects.

Using this model user experience can be divided into elements, and those elements in turn can be measured.

The *pragmatic* aspects of the model refer to a product's perceived ability to support achievement of *do-goals*, such as "making a telephone call" or "ordering take-away food". These pragmatic aspects fall more in line with what is traditionally understood with plain "usability".

The *hedonic* aspects of user experience, on the other hand, refer to an interactive product's perceived ability to support achievement of *be-goals*, such as "being competent" or "being trendy". Hassenzahl's hedonic/pragmatic model divides hedonics into further three different subdimensions: stimulation, identification and evocation: *Stimulating* products offer new impressions, opportunities and insights, enabling users to achieve personal development, that is, to gain new knowledge and develop skills. In addition, as people express themselves through objects, products should allow users to *communicate identity* through them. Finally, products can also *evoke memories*, by representing past events, relationships or thoughts that are important to an individual. [10, 11]

These distinct constructs of the hedonic/pragmatic model compose an obviously reduced model of user experience. Yet they model user experience as a concept that is manageable, and, more importantly, measurable. [11]

### 2.1.2 User Experience and Usability

ISO standard 9241-210 defines *usability* as the "extent to which a system, product or service can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use" [6]. Contrasted against the definition of user experience, usability is a more narrow, more context-bound view into interaction between a user and a product. This is mirrored by Diefenbach & Hassenzahl [5] who state:

Assessing a product's pragmatic quality calls for a focus on functionality and usability in relation to a potential task at hand. Assessing a product's hedonic quality calls for a focus on the Self and its needs, that is, the question of why someone owns and uses a particular product.

According to this, expanding measurements from purely pragmatic – or usability-centered – to include hedonic qualities allows a potentially deeper focus into the user and their wants and needs.

Hassenzahl [11] also states that user experience differs fundamentally from usability: where usability is more concerned with negative aspects of products

– what is broken, sub-optimal, irritating – user experience has more of a focus in positive aspects of products: enjoyment, beauty, satisfaction. User experience also tackles the subjective side of user–product interaction, rather than having a focus on objective performance metrics. [11]

### 2.1.3 Consistency

There is no widely accepted definition for consistency in the field of human-computer interaction [1, 13, 26]. Wolf [35, p. 89] says that consistency “means that similar user actions lead to similar results” and that it “allows the user to transfer skills from one area to another”. According to AlTaboli & Abou-Zeid [1] researchers generally mean by interface consistency that “all items in the interface with the same information should have the same design (format, style...) within a display (or system) or among displays”. Kellogg [14] says consistency is an inherently relational concept, and that interfaces and parts of interfaces are generally consistent in relation to some aspects, and inconsistent to other aspects. Relatedly, Nielsen [20] reports of a workshop of 15 HCI experts that was unable to arrive at a single definition for consistency, defining it instead as a sum of its dimensions.

Perhaps it is best to approach consistency through these dimensions of consistency? The dimensions can roughly be divided to *internal* and *external consistency* [8, 13, 28]. Internal consistency is the consistency of an interface in regards to itself, such as using the same terminology across the application or system, using a similar colour theming throughout, or displaying system messages to the user in identical fashion in different cases. External consistency is the consistency of an interface in the context of external influences, such as other applications or systems, current trends and standards, and user’s expectations and familiarity with other software. Wolf [35] mentions consistency across different applications (such as the ubiquitous hamburger menu found in many current mobile interfaces), consistency over time (in regards to previous versions of the same software), consistency within a platform (staying true to platform’s user interface guidelines, such as implementing the order of “ok” and “cancel” buttons in dialogs depending on whether the software is designed for Windows or Macintosh environment), and consistency across platforms (aspects of the user interface not within the purview of platform guidelines should be guideline-independent and consistent across platforms). Ozok & Salvendy [26] define the difference between internal and external consistency as the former meaning consistency within a task and the latter consistency among different tasks.

Some researchers [8, 28] also refer to a dimension of *metaphorical* or *analogical* consistency, relating to use of metaphors in interfaces - that is,

this dimension is the consistency of an interface in comparison to concepts outside of the domain of computers, such as presenting a single email item in an email application as a graphical item resembling a real-world letter.

Kellogg [13, 14] on the other hand considers this dimension of metaphorical consistency a part of external consistency. Instead, she proposes a framework where the dimensions are taken as conceptual, communicational and physical components of interface, and these are placed in a matrix across internal and external sources of consistency (see table 2.1). In this framework the conceptual component refers to the metaphor used in a task or an action in the interface, the communicational component refers to the interaction, the input and the output of the interface, and the physical component refers to the visual representation of the interface. [13, 14]

Table 2.1: Kellogg’s [13] proposed framework for defining consistency (simplified).

|                        | <b>Internal consistency</b> | <b>External consistency</b> |
|------------------------|-----------------------------|-----------------------------|
| <b>Conceptual</b>      |                             |                             |
| <b>Communicational</b> |                             |                             |
| <b>Physical</b>        |                             |                             |

As for advantages of consistency, they seem intuitive: users learn consistent systems faster [28] and can transfer that knowledge to other parts of the same, or different, system [13, 22]. Users can work faster and more efficiently on consistent systems [13, 15, 22], although perhaps only during multitasking situations [18]. They make less errors [1, 22, 26, 28] and seem generally more satisfied with consistent systems [1, 13, 22]. Consistency can lower training costs [15, 22, 30] and consistent systems require less support [22]. And finally, users complain less about consistent systems [15].

Mental models are a useful tool for abstracting how and why consistency can work. Satzinger & Olfman [28] describe mental models as a user’s knowledge of a computer application. The forming of a user’s mental model, or just User’s Model [30], seems to be aided by a consistent interface [28]. An interface designer’s job is to help users in forming their mental model to match that of the designer’s Design Model [30]. An inconsistent interface may make the User’s Model and Design Model clash.

On the other hand, Grudin [8] states that “a ”fully consistent system” is

not achievable". He goes on to argue that trying to design a truly, completely consistent system is a folly, and that some degree of inconsistency is in fact very beneficial for usability, and that the strive for consistency should always be subservient to a better understanding of user's tasks and work context [8, 9]. He also gives an example [8] where the most obvious solution is to flat-out reject consistency. He describes a system where a "print" operation can be executed on a folder. Research indicated that most people would have expected this command to print all the files within the folder. However, the system's designers argued that the print command should have printed a list of the files in the folder instead, as this would have been consistent with the system's internal architecture. As far as the system was concerned the folder was merely a collection of pointers to the files within, so if executing a print command on a file should print out its contents, shouldn't printing a folder also print its contents: the list of file names? Clearly this would have been the more unintuitive option of the two.

Koritzinsky [15] gives a related example where internal and external (or metaphorical) consistency clash against each other:

For example, if all of the icons of a user interface are square, it may seem inconsistent to add a new icon that is a circle. However, if that new icon is to represent a clock, and the user thinks of most clocks as circular, the circular shape may be quite appropriate. In other words, the icon is consistent with the user's vision and needs.

A 1998 study by Satzinger & Olfman [28] noted increase in task accuracy when visual inconsistency was introduced. This was theorised to be due to distinctive visual appearance aiding with users creating a more accurate mental model, at least when users are simultaneously learning multiple applications.

#### **2.1.4 User Experience and Consistency**

Consistency of user experience feels like an intuitive concept, but is not entirely straightforward to define. In the previous section consistency of interfaces was defined through dimensions and components: dimensions of internal and external consistency, and components of conceptual, communicational and physical consistency. These same concepts, used to describe consistency of interface design, can also be used to define consistency of user experience: a system's user experience can be considered consistent or inconsistent either internally (compared to the user experience within the same

system or product) or externally (compared to user experiences offered by other systems). The system's user experience can also be considered consistent across components of conceptual (metaphor that the user experience may convey), communicational (the interaction between the user and the interface) and physical (audiovisual and physical representation, the look of the product or system) consistency.

To give some examples, an external conceptual inconsistency in user experience of a car could be that the direction (clockwise or counter-clockwise) a user would need to flick the turn signal lever would not match the direction they would then rotate the steering wheel to actually turn the car. This would break the metaphor where the direction of the turn signal lever matches the direction of user's eventual steering wheel motion, causing conceptual inconsistency. It would also not be how users would generally expect cars to behave by their prior experiences, causing external inconsistency.

An internal communicational inconsistency in a car's user experience would occur if some of the door handles on the same car would require users to pull on them to open the door, and some of them would require turning the handle instead.

When a user initiates a turn signal on their car, a ticking metronome is generally heard. Varying the pitch of this ticking depending on whether the user is turning left or right would cause both internal and external physical inconsistency, as the audio feedback of the interaction would differ without a clear, well-established justification as to why turning left would be pitched higher or lower than right.

These categories certainly aren't always clear-cut – the first turn signal lever example could also be categorised as communicational inconsistency as the direction the user needs to move the lever is related to interaction as much as it is to metaphors.

## 2.2 Measuring User Experience and Usability

This section describes methods for measuring user experience and usability. Two questionnaires are presented: AttrakDiff for measuring user experience, and SUS for measuring usability. They are both used to collect quantitative data, and are both post-evaluation questionnaires that participants of usability evaluations are asked to fill after the participant has used the product or system that is being evaluated. The think-aloud process, used for collecting qualitative findings during usability evaluations, is also described.

### 2.2.1 AttrakDiff

AttrakDiff (or as it alternatively appears in some other variants or with a slightly different name in some research papers: AttracDiff, or AttracDiff 2) is a post-evaluation questionnaire. It collects quantitative data with a 7-point semantic differential scale.

The questionnaire is based on Hassenzahl’s hedonic/pragmatic model of user experience, as described in section 2.1.1 on page 16. It assumes that perceived hedonic and pragmatic qualities describe separate dimensions of user experience [27]. The questionnaire presents 21 opposite adjective pairs that are meant to capture pragmatic and hedonic aspects of quality. In addition, the hedonic aspects can be divided into identification and stimulation [32], although the subdimension of evocation, present in Hassenzahl’s original model, is left out.

Seven of the 21 opposite adjective pairs are meant to capture pragmatic quality: *human–technical*, *simple–complicated*, *practical–impractical*, *straightforward–cumbersome*, *predictable–unpredictable*, *clearly structured–confusing* and *manageable–unruly*. The fourteen other pairs capture hedonic quality: *connective–isolating*, *professional–unprofessional*, *stylish–tacky*, *premium–cheap*, *integrating–alienating*, *brings me closer to people–separates me from people*, *presentable–unpresentable*, *inventive–conventional*, *creative–unimaginative*, *bold–cautious*, *innovative–conservative*, *captivating–dull*, *challenging–undemanding* and *novel–ordinary*. [12]

A study by Walsh et al. [33] found the AttrakDiff questionnaire a useful tool for measuring user experience repeatedly over a long-term study. They conclude that AttrakDiff was simple to implement and their participants found the questionnaire easy to fill, but they point out that AttrakDiff results alone do not explain causes of user experience changing over time.

### 2.2.2 System Usability Scale

System Usability Scale (SUS) is a widely used tool for measuring usability. It is a cost-effective, 10-item Likert scale giving “a global view of subjective assessments of usability” [4]. The questionnaire was created as a solution to the need of having a quick, simple, cost-effective and a reliable way of measuring user performance changes between software versions. [4]

Participants are asked to fill the SUS form directly after using the product or system that is being evaluated, before any further debriefing or interview should occur. A fully-filled form is scored according to SUS scoring rules, resulting in a number between 0 and 100, with higher numbers denoting better usability. [4]

According to Kühnel [16, p. 98] there is a strong correlation between SUS scores and the pragmatic aspect of AttrakDiff. Kühnel, Westermann, Weiss & Möller [17] also came to conclusion that SUS is useful if a mere comparison of systems or different versions of the same system is wanted. For gaining further insight into the collected usability evaluation scores Kühnel et al. recommend AttrakDiff instead as it specifically distinguishes between hedonic and pragmatic aspects.

### 2.2.3 Think-aloud Process

In think-aloud process the participants are encouraged to share their mind while using the product or the system [3, p. 19]. By externalising their thoughts the participants allow the session moderator to know what they are thinking instead of the moderator having to guess [29, pp. 335-337]. The process can be used to organically collect qualitative evaluation findings from participants, using their own words.

Combining quantitative task completion time/performance measurements with think-aloud process is possible. Studies seem to suggest that at least in some cases asking the participant to think aloud during a usability evaluation does not have a significant effect on the time used to perform tasks in the test [3, 25].

## 2.3 A Model for Measuring Consistency

Literature on human-computer interaction does not offer any previous models for measuring the effect of consistency on user experience or usability. In this thesis the consistency of user experience and usability is measured as the effect consistency-related improvements have had on an iterative user experience and usability study.

This measurement of consistency is done by first measuring user experience and usability in a study involving an initial version of the software. After this, the software is iterated on by tackling consistency-related findings gathered from the first study. Another study is conducted, now with the second, improved version of the software. The same user experience and usability measurement tools that were used in the first study are then used in the second one. Finally, the measurements from these two studies are compared.

The idea is that by focusing on consistency-related fixes between the two studies the effect the improvements to consistency had on the software can be collected directly as the change in measured user experience and

usability. For this work, however, these consistency-related fixes must be identified beforehand as being, indeed, consistency-related. The definition of consistency as dimensions over components as described in chapter 2.1.3 can be used for this identification. As the fixes are identified as being consistency-related, more certainty is gained for the measurements being caused by the changes to consistency in the software.

When comparing results from two iterative designs it's advantageous to collect quantitative results in favour of qualitative ones as quantitative measurements are easier to compare against each other. In the study presented in this thesis two quantitative methods were used: AttrakDiff and SUS questionnaires. These are both described in section 2.2.

## Chapter 3

# The ASP.NET Frontend

This section describes the initial "ASP.NET" version of the frontend to Execution Framework. The test execution environment system prior to the Execution Framework, referred to as "AUIT", is also briefly touched on.

### 3.1 Design

This section describes the design of the ASP.NET frontend used for the first round of usability tests.

#### 3.1.1 Home Page

The home page (see figure 3.1) was essentially a collection of navigational links to the main functionalities of Execution Framework. It offered links to the Test Browser, the Workflow Manager, configuration of Systems under Test, and the Keyword Test Designer (the last one being an external tool which is not a part of the Execution Framework). These links were presented as dashboard tiles on the page, with short descriptions of the pages these links represent. See figure 3.2.

On top of this, and every, page was a navigation bar with quick links to these same pages.

#### 3.1.2 Test Browser

Aim of the Test Browser page (see figure 3.3) was to offer a place for the user to see all tests in a build in one place, as opposed to viewing subsets of them in workflows in the Workflow Manager. The tests themselves were organised hierarchically, with tests being inside test paths, and these branches of tests

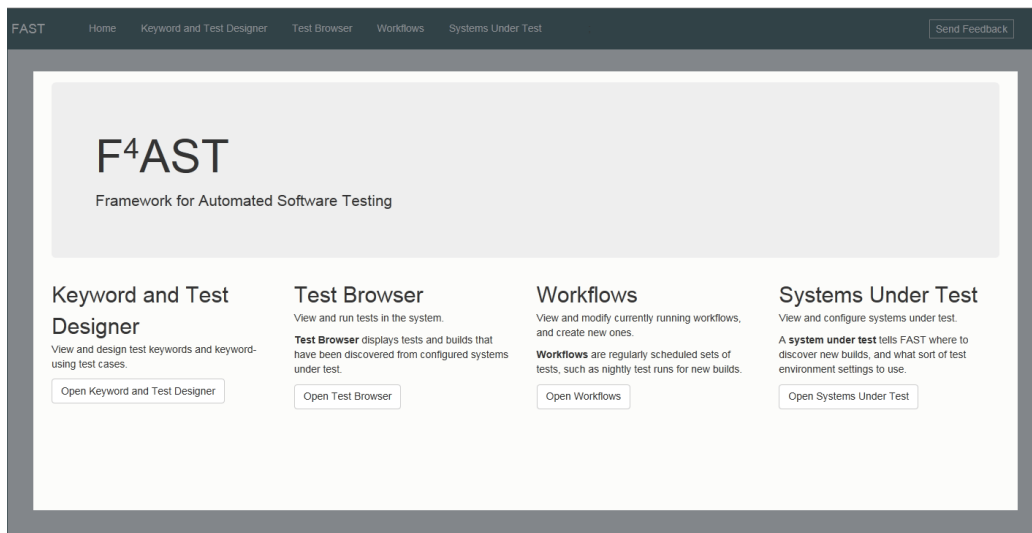


Figure 3.1: ASP.NET frontend home page. The navigation bar is also visible at the top of the page.

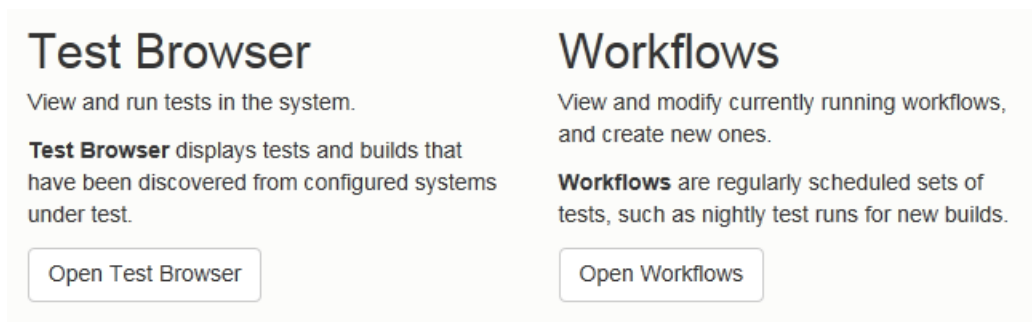


Figure 3.2: Flat dashboard tiles on the home page.

paths inside physical test files. This structure was represented in the Test Browser by displaying tests in a tree resembling how a file explorer might display folders and files. The page itself only displayed one build at a time - user had to first choose a system under test, then a build under it to display all tests in the build.

The user interface allowed user to select multiple tests at once and tell the system to run them, in which case the tests would be sent to the test queue before eventually being run in a worker machine in the worker cloud. Run results of these tests, if there were to be any, were also displayed. Clicking on a test would bring user to the Test Executions page for the test in question.

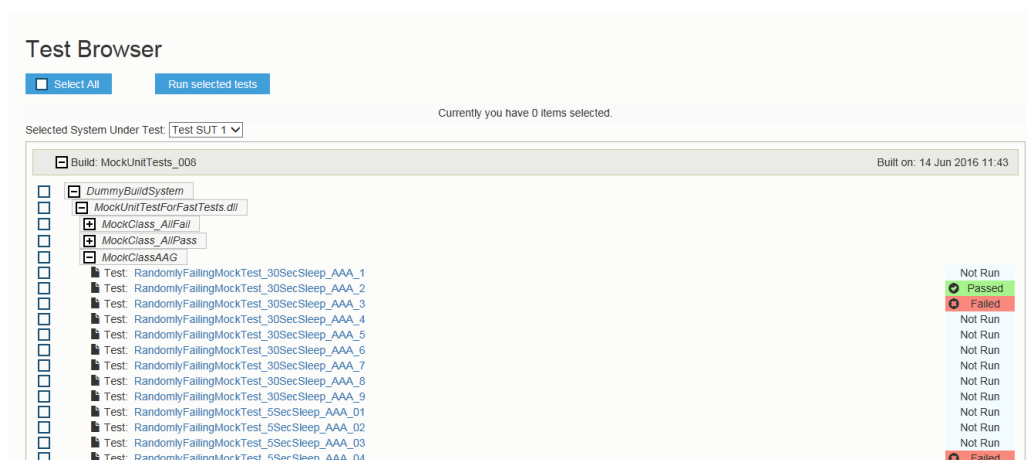


Figure 3.3: ASP.NET frontend Test Browser page.

### 3.1.3 Workflow Manager

The Workflow Manager page (see figure 3.4) was a hub for all workflow-related pages. It collected all workflow templates in the system, displayed related workflow instances under them, and had a button for adding new workflow templates into the system.

Workflows themselves were meant as the main tool for users to configure reoccurring sets of tests for new builds: nightly test runs and continuous integration test runs.

The Workflow Manager page displayed all workflow templates in the system, sorted by them being either active, inactive, or "manually queued", the last one being used to collect tests queued directly from the Test Browser into one place in Workflow Manager. Under each workflow template was a table of all the workflow instances in it. Some basic information was shown for each template and instance: their names, their test run progress, their current activity state, when they had been launched, and so on.

Certain activity thresholds were used to hide workflow templates and instances that hadn't seen any recent activity, and clicking a "show all items" button was required to view them in these cases. This was to make the interface slightly less cluttered when the system had numerous workflows.

This was one of the few pages in the ASP.NET frontend that updated itself automatically, refreshing the workflow template items with latest progress data several times per minute. The idea was to have the page function as a dashboard for the most frequent daily use cases of the site, where users could leave the site open in a browser tab and check in occasionally to see

the progress and state of test runs in the system.

The screenshot displays the 'Workflows' section of the ASP.NET frontend. At the top, there is a '+ Create a new Workflow' button. Below this, three workflow cards are listed:

- Test SUT 2 Primary Tests**: Priority: 1, Primary Tests for Test SUT 2 branch., Every day, for every new build. Shows 1 running and recent instances of this workflow. Edit this workflow.
- Test SUT 2, all**: Priority: 2, Nightly runs for every Test SUT 2 build., Every day, for every new build. Shows 1 running and recent instances of this workflow. Edit this workflow.
- Test SUT 1 workflow**: Priority: 3, Test runs., Every day, for every new build. Shows 1 running and recent instances of this workflow. Edit this workflow.

Below the workflow cards is a section titled 'Manually queued work' with a card for 'Manually queued tests' with Priority: 1, showing 0 running and recent instances of this workflow.

Figure 3.4: ASP.NET frontend Workflow Manager page.

### 3.1.4 The Workflow Instance Page

The Workflow Instance page (see figure 3.5) was used to show details of and interact with a single workflow instance. The top of the page was reserved for a small summary of the workflow instance, including which build the instance corresponded to. Below it were all test runs contained in the instance, listed in a table. Related information such as test result, test run state and name of the worker machine the test was run on was also shown. Link to the Test Executions page of each test was also available.

Users were able to run any test in the workflow instance again from this page. The entire workflow instance could also be cancelled, which removed any pending tests from worker queue and stopped any on-going test runs.

### 3.1.5 The Test Executions Page

The Test Executions page (see figure 3.6) was used to show all test runs of a single test of a specific build. The test runs, or test executions, were

Workflow instance: Test SUT 1 workflow

Build: MockUnitTests\_008    System under test: Test SUT 1    Requested by:

State: Active    Finished tests: 170/441

Start time: 5.8.2016 11:50:58

Description:

[Cancel this workflow instance](#)

Tests in this workflow instance

Select All    [Re-run selected tests](#)    Filter:

Currently you have 0 items selected.

| Test name   | State               | Verdict timestamp | Worker      |
|---|---------------------|-------------------|-------------|
| <input type="checkbox"/> MockUnitTestForFastTests.dll/MockClassAAA/RandomlyFailingMockTest_5SecSleep_ABA_11 | Test result: Passed | 5 Aug 2016 13:31  | HE4D1RKK212 |
| <input type="checkbox"/> MockUnitTestForFastTests.dll/MockClassAAC/RandomlyFailingMockTest_5SecSleep_ABC_19 | Test result: Passed | 5 Aug 2016 13:31  | HE4D1RKK212 |
| <input type="checkbox"/> MockUnitTestForFastTests.dll/MockClassAAE/RandomlyFailingMockTest_30SecSleep_AAA_2 | Test result: Passed | 5 Aug 2016 13:31  | HE4D1RKK212 |
| <input type="checkbox"/> MockUnitTestForFastTests.dll/MockClassAAG/RandomlyFailingMockTest_30SecSleep_AAA_2 | Test result: Passed | 5 Aug 2016 13:29  | HE4D1RKK212 |
| <input type="checkbox"/> MockUnitTestForFastTests.dll/MockClass_AllFail/FailingMockTest_5SecSleep_ABA_12    | Test result: Failed | 5 Aug 2016 13:28  | HE4D1RKK212 |
| <input type="checkbox"/> MockUnitTestForFastTests.dll/MockClassAAA/RandomlyFailingMockTest_5SecSleep_AAA_09 | Test result: Passed | 5 Aug 2016 13:28  | HE4D1RKK212 |
| <input type="checkbox"/> MockUnitTestForFastTests.dll/MockClassAAD/RandomlyFailingMockTest_5SecSleep_ABA_17 | Test result: Failed | 5 Aug 2016 13:28  | HE4D1RKK212 |
| <input type="checkbox"/> MockUnitTestForFastTests.dll/MockClassAAF/RandomlyFailingMockTest_5SecSleep_ABA_13 | Test result: Failed | 5 Aug 2016 13:27  | HE4D1RKK212 |
| <input type="checkbox"/> MockUnitTestForFastTests.dll/MockClass_AllPass/PassingMockTest_5SecSleep_AAA_10    | Test result: Passed | 5 Aug 2016 13:27  | HE4D1RKK212 |
| <input type="checkbox"/> MockUnitTestForFastTests.dll/MockClass_AllFail/FailingMockTest_5SecSleep_ABA_11    | Test result: Failed | 5 Aug 2016 13:27  | HE4D1RKK212 |

Figure 3.5: ASP.NET frontend Workflow Instance page, with tests in workflow instance sorted by their verdict timestamps.

displayed in a list along with each test's result and a link to an external test log. Because of technological limitations (due to the nature of the network Execution Framework was deployed in all the test log links pointed to local network drives, but for security reasons most internet browsers prevent users from opening links that point to local network drives) users had to manually copy and paste these links into new browser tabs in order to open them.

Test executions for  
*MockUnitTestForFastTests.dll/MockClassAAG/RandomlyFailingMockTest\_5SecSleep\_AAA\_01*

Build: MockUnitTests\_005    System under test: Test SUT 1    Build source: local build source

Component: DummyBuildSystem    Platform:    Flavor:

| Name                                     | Timestamp         | Verdict | Executing Machine | Assert Message                              | Test Log   |
|--|-------------------|---------|-------------------|---|--|
| RandomlyFailingMockTest_5SecSleep_AAA_01 | 5 Aug 2016 15:44  | Passed  | HE4D1RKK212       |   | jsaarja_HE4D1RKK212 2016-08-05 15_44_06.trx <a href="#">Copy link to clipboard</a> |
| RandomlyFailingMockTest_5SecSleep_AAA_01 | 5 Aug 2016 15:42  | Failed  | HE4D1RKK212       | Assert.Fail failed. Randomly forced failure | jsaarja_HE4D1RKK212 2016-08-05 15_42_35.trx <a href="#">Copy link to clipboard</a> |
| RandomlyFailingMockTest_5SecSleep_AAA_01 | 5 Aug 2016 15:41  | Passed  | HE4D1RKK212       |   | jsaarja_HE4D1RKK212 2016-08-05 15_41_10.trx <a href="#">Copy link to clipboard</a> |
| RandomlyFailingMockTest_5SecSleep_AAA_01 | 10 Jun 2016 18:54 | Passed  | HE-UITB6          |   | he-bmuser_HE-UITB6 2016-06-10 18_54_30.trx <a href="#">Copy link to clipboard</a>  |

Figure 3.6: ASP.NET frontend Test Executions page.

### 3.1.6 Workflow Template Editor

The Workflow Template Editor page was used to both create new workflow templates and to modify existing ones (see figure 3.7). The editor required user to enter some basic details for a new template (name, chosen system under test), the tests that should be run in any instances of the template (presented as test paths, essentially wildcarded test folders under which everything would be run), and the schedule on when and how often the template should trigger and generate new instances from itself.

Test paths were chosen with a modal widget that resembled the Test Browser, but was made to look slightly dissimilar to emphasise the difference of the user now choosing paths from a hierarchy instead of selecting individual tests (see figure 3.8).

The screenshot shows the 'Workflow Editor' page with the following details:

- Name:** Test SUT 2 Primary Tests
- Description:** Primary Tests for Test SUT 2 branch. (Optional)
- State:**  Enabled,  Disabled. Note: If the workflow is set to disabled, it will remain in the system but no scheduled runs will be instantiated from it.
- Priority number:** 1. Note: Priority number of the workflow. Smaller numbers mean higher priority. Workflows are run using round-robin style.
- Work configuration:**
  - Target system under test:** Test SUT 2. Note: The system under test that this workflow will run against.
  - Target build source:** Test SUT 2 build source. Note: This workflow will run against builds found from this build source.
- Test selector:**  Test path. Note: When Test Path selector is used, the tests to be run are chosen with a test path-based wildcard selection. E.g. path selector "test.dll/unit tests" would run tests under "test.dll/unit tests/component tests" and "test.dll/unit tests/nightly/calculation tests", but not "test.dll/integration tests". Other types of selectors are currently not supported.

Figure 3.7: ASP.NET frontend Workflow Template Editor page.

### 3.1.7 Systems under Test Pages

The Systems under Test pages were several relatively simple pages displaying all the systems under test artifacts in the system (see figure 3.9), offering

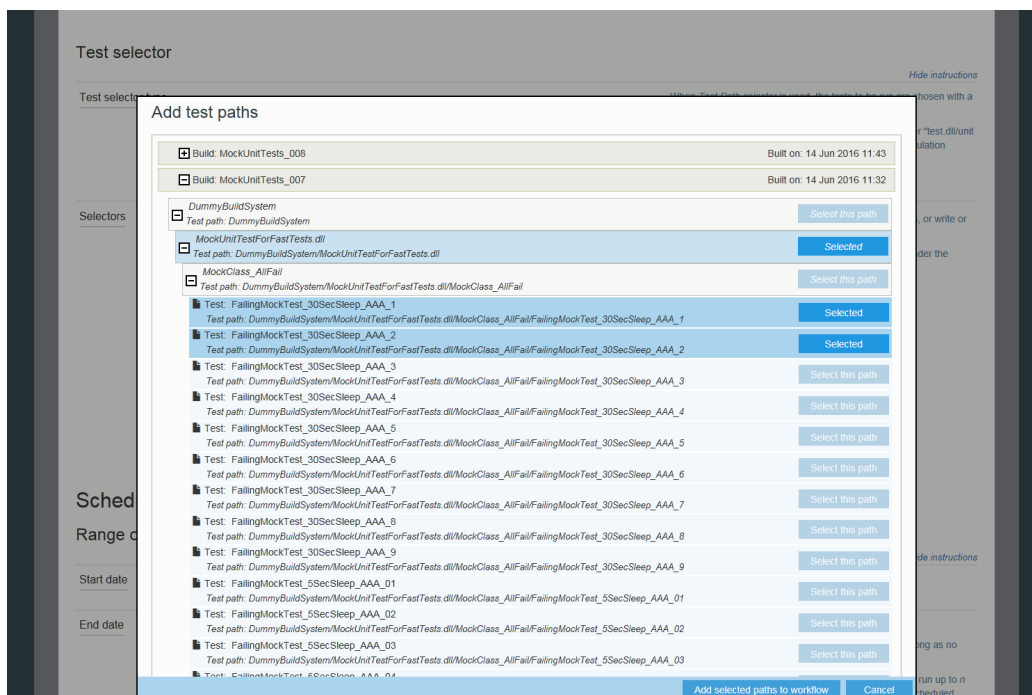


Figure 3.8: The test path picker widget for Workflow Template Editor.

users the ability to perform basic CRUD (*create, remove, update, delete*) operations on them. Users were able to configure system under test artifacts in these pages (see figure 3.10), including giving them one or more build sources from which these systems under test would scan and discover physical test files (usually .DLL library files).

## 3.2 Old AUIT System

The old AUIT (*Automated Unit and Integration Testing*) system was a distributed test execution environment that was used for several years prior to launch of the first version of the "FAST" Execution Framework. Like Execution Framework, the AUIT system also offered a graphical user interface, as a set of mainly tabular, static ASP.NET pages with little to no JavaScript code. The user interface was considered crude and rudimentary, but got the job done at the time.

The old AUIT version of the frontend had no concept of workflows. Instead, if users wanted to modify which test sets were being automatically launched for new builds they had to manually edit an SQL database. Test

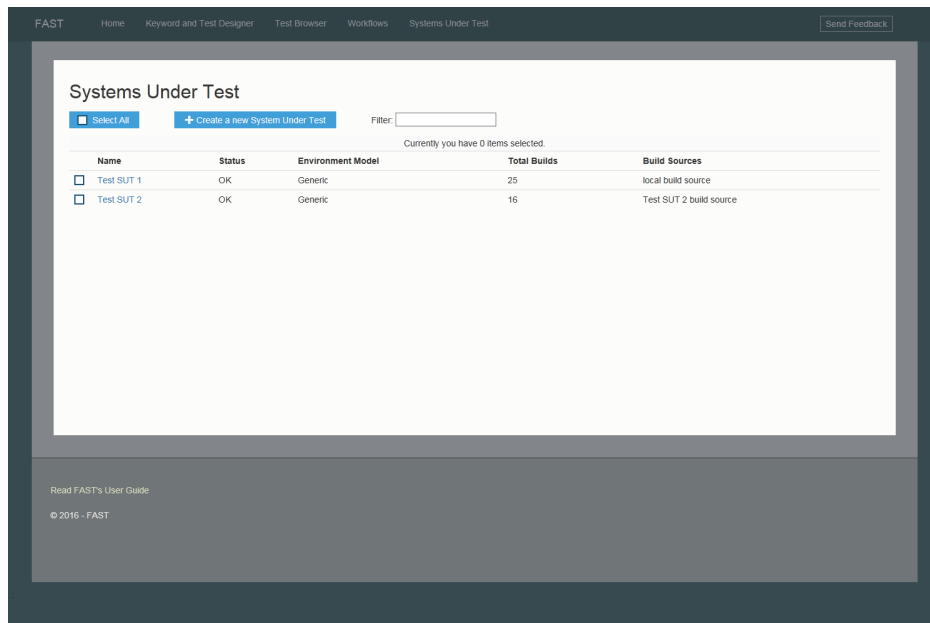


Figure 3.9: ASP.NET frontend Systems under Test index page.

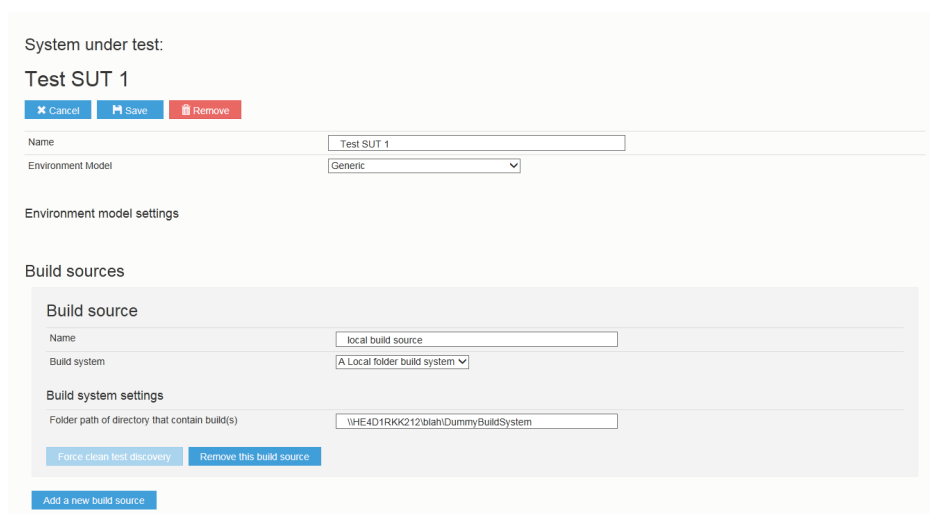


Figure 3.10: ASP.NET frontend individual system under test configuration page.

results were also not available from the frontend. Instead users had to open test result log files manually from a different computer on the network.

The AUIT frontend had some features that were not implemented in Execution Framework's frontend, however. In AUIT users were able to change priority of tests on the fly, being able to prioritise certain test runs to be picked into the worker cloud as soon as possible. The worker cloud itself was more transparently available for users than it eventually would be in the first version of Execution Framework. Users were able to view all worker machines in the cloud in AUIT's frontend and see which tests, if any, the machines were currently running.

The old AUIT frontend is described here because of its historical significance – many of the current and future users of the Execution Framework have used the old AUIT system for varying amounts.

## Chapter 4

# Evaluating the ASP.NET Frontend

This section explains how and with whom the ASP.NET Execution Framework frontend was usability tested, and which methods were used and the reasoning behind them.

The gathered evaluation data is presented in chapter 5, starting on page 40.

### 4.1 Evaluation Goals

The main goal during the first round of usability tests was to gather measurements of current state of usability and user experience in the ASP.NET frontend. These measurements would be compared against measurements from the second round of usability tests later, as reported in chapter 7 on page 66.

Secondary goal of these usability tests was to find qualitative issues in the frontend that could be taken into account when designing the new "React" frontend, especially pertaining to consistency of user experience.

### 4.2 Participants

Participants for the test were recruited from different teams in-house. A total of 13 participants were used.

Participants were predominantly male, with 10 of the participants being native Finnish speakers. All participants spoke fluent English. The test materials were in English, but the test sessions themselves were held in Finnish for the Finnish-speaking participants, and in English for everyone else.

11 of the participants mentioned considerable experience with the old "AUIT" system, with only 2 participants mentioning having very little to no experience with the old framework. This information was collected from the pre-test questionnaire that the participants were asked to fill.

### 4.3 Procedure

Test sessions were held in a closed conference room to avoid outside distractions.

The software was tested on a laptop computer and accessed with the Google Chrome web browser. The frontend web service itself was hosted on local intranet. The server of the Execution Framework that was used was running on a separate development server that wasn't used by other people, thus closing it off from any other users than whoever was currently testing it in the usability session. This was to keep the state of the system predictable and to avoid any external users accidentally causing noticeable system-wide effects during testing.

The state of the Execution Framework was reset into an initial state after each test session. This initial state contained some basic pre-filled data in the system, with several builds, tests and workflows. This was to not only make it easier for participants to jump directly into scenarios, but to also make the tests seem more realistic. It is quite unlikely that most users of Execution Framework would ever see the environment as "empty", without any data configured by the system administrators.

The sessions themselves were recorded on video, with the video camera pointed on the laptop's monitor and recording all speech between the participant and the moderator. As a backup, a smartphone was also used to record audio of the session in case something would've gone wrong with the video recordings. Notes were also taken during the session, which then formed the backbone of qualitative findings gathered from the usability study.

A pilot session was held before the rest of the sessions. Rest of the sessions were held over 6 working days in April 2016, with up to three sessions per day. An hour was reserved for each session, plus fifteen minutes of setup time before each session. The sessions generally lasted between 30 and 50 minutes, with a couple of sessions going over time.

## 4.4 Scenarios

Scenarios were designed so that they could be re-used during the second round of usability tests with the React frontend in order to keep the studies comparable. They were also designed to make participants interact with areas of the user interface that had been pre-analyzed to contain potential clashes in user experience consistency.

The scenarios were presented to the user verbally by the moderator, and participants were told that they could ask the moderator to repeat the scenario at any time. Participants were also instructed to go back to the front page of the FAST frontend after each scenario task.

Four scenarios were used in total. They consisted of one scenario to capture initial impressions, followed by three scenarios with focus on three main use cases of the software: investigating individual test failures and running individual tests; investigating failures in scheduled nightly test runs; and configuring nightly test run schedules. The scenarios are presented and explained below.

### 4.4.1 Scenario 1: Viewing Front Page

The scenario text for the first scenario was as follows:

Take a moment to look at the FAST front page without clicking on anything. What are your first impressions? What do you think these links and tabs on the page mean? What do you think would happen if you clicked on them? Are there any words or labels on the page that you are unsure about?

The motivation for this scenario was to capture the participant's initial impressions of the frontend when they see it for the very first time [3, pp. 131, 135, 153-154]. Of particular interest was also whether the language and terminology used in the front page matched with what participants were expecting or were familiar with. Likewise it was interesting to know if users would be able to naturally figure out what the different functionalities and sub-pages branching from the front page would actually be, just by their titles and short descriptions, even if unclear descriptions would generally just be a one-time nuisance after user would learn what sort of functionalities the frontend would support.

### 4.4.2 Scenario 2: Queueing Test Runs

A colleague has set up automated tests in FAST for the Component X branch. You have been told that there have lately been issues with "Unit test set Y" tests, and you now want to run these tests for the latest Component X build to see how they go.

This scenario was used to introduce the concept of running automated tests from specific builds, a practice which the participants should have already been familiar with from the old testing system. An interesting aspect was also to find out where participants would start with this task without any other prodding — whether they would first click on the Test Browser component, as was intended, or whether they would wander off into other pages or run the tests from a workflow from the Workflow Manager page.

### 4.4.3 Scenario 3: Scheduling a Workflow

You've been asked to schedule Component Y's all core and external component-related tests each Tuesday and Thursday night.

The third scenario was designed to test the usability and UX aspects of creating a new scheduled test run. A key point was to see if participants would understand what "workflows" meant in the system and that "scheduled work runs" were related to workflows.

The specifics of the schedule – the multiple test sets and the non-standard weekly schedule of running only on Tuesdays and Thursdays – were designed to force the participant to interact further with the workflow creation dialog, and in particular to get them to attempt to select multiple values from where they would choose which tests to schedule to see if that UI component would feel natural to them.

### 4.4.4 Scenario 4: Investigating a Failing Workflow

A colleague has previously configured another workflow of "Unit test set Y"-related tests for Component X to run automatically every night. However, you've been told that some of these tests have started to fail lately, and you've been asked to investigate the issue, and then to attempt if running the failing tests again from the workflow would solve the failures.

The fourth and final scenario was used to gauge how participants would approach investigating failures that occurred in tests that are in scheduled

workflows, and whether the slightly different use case in scenario 2 would help or hinder in this task.

Participants were also asked to run the tests again, just to simulate and evaluate a scenario where a user would naturally run tests from a workflow again and then view results of the re-run.

#### 4.4.5 Changes to Scenarios

Two major changes were made to the scenarios during the study.

Originally the last two scenarios, the "Scheduling a Workflow" and "Investigating a Failing Workflow" scenarios, were in different order, so that the participant was asked to investigate a failing workflow before they were asked to create a new scheduled workflow. The motivation for this was that testing the scheduling dialog was, as far as broad usability strokes were concerned, the least interesting of all the scenarios, and was thus left as the last scenario so that it could be dropped out in case there would not have been time to run it during an evaluation session.

However, the participant of the pilot session had some trouble with where to begin with investigating the failed workflow, but had no issues with creating a new scheduled workflow in the final scenario. The scenarios were swapped after the pilot session to hopefully better introduce the concept of workflows for the participants and to make the usability sessions smoother so they would cover more ground and find more issues.

Scenario 4, "Investigating a Failing Workflow", did also not originally include any references to the term "workflow". Instead, phrases like "set" and "test set" were used, with the motivation that it would be interesting to see how naturally participants would mentally map these abstract concepts of scheduled test sets to the "Workflows" component visible on the front page. The first four sessions (including the pilot session), however, had definite trouble distinguishing between the usage of the Test Browser and the Workflow components for this task. As this confusion between the components was starting to take a lot of time from the sessions, and as there was a risk that due to this the Workflow component would not receive much actual evaluation, it was decided to simply mark this as a usability finding and to introduce the word "workflow" into the scenario description.

After this change the final 9 of the 13 session participants had no real further difficulties between distinguishing these two components as far as the scenario focus was concerned. It is noteworthy, however, that after the change the scenario description is much more leading, and it may have had an effect on the post-test questionnaire rankings.

## 4.5 Collecting Findings

Qualitative findings were gathered by both observing participants as they performed the given scenario tasks as well as by asking them to think aloud. Participants were also encouraged to ask questions and share their thoughts after each scenario.

Quantitative findings were collected only by measuring participant "satisfaction levels" of user experience and usability with AttrakDiff and SUS questionnaires, elaborated more in the next section. Task performance was not measured, although scenario completion times could have been retroactively extracted from the recorded session tapes. This was decided to be unnecessary, however, since these completion times would not have been comparable to completion times from the future second usability evaluation round as participants ended up being more familiar with the system by then.

## 4.6 Pre-Test and Post-Test Questionnaires

Before and after each session the participants were presented with printed-out questionnaires that they were asked to fill in. All of the questionnaires used in the study are available in appendix A.

Participants were asked to fill a short pre-test questionnaire before the test that asked their previous experience with the new "FAST" Execution Framework that they were going to soon try, their previous experience with the old "AUIT" system as described in chapter 3.2 (page 31) and their previous experience with automated testing in general. Participants were asked to circle in their answer for each of these three questions from the four options of "not at all/not sure", "very little", "some" and "quite a lot". See figure A.1 on page 92.

After the test the participants were asked to fill AttrakDiff and SUS questionnaires. Both of the questionnaires were presented as paper copies. The used two-page AttrakDiff questionnaire is available as figures A.3 and A.4 on pages 94 and 95. The used SUS questionnaire is available as figure A.2 on page 93.

## Chapter 5

# Results of First Evaluation

This chapter discusses findings collected from the first round of usability evaluation studies conducted on the ASP.NET version of the Execution Framework frontend.

### 5.1 Overview

A total of 13 evaluation sessions were held, including the pilot session. Three more sessions had been booked but the participants had to cancel beforehand.

As explained in section 4.4.5 on page 38 the scenarios were changed slightly after the pilot session (the order of the last two of the four scenarios in the script was swapped) and again after the fourth session, when the leading word "workflow" was introduced into briefing of the final scenario.

### 5.2 Quantitative Findings

Quantitative findings were collected using SUS and AttrakDiff forms as explained in section 4.6 on page 39.

Answers to AttrakDiff questionnaire were scored by converting each participant's adjective-pair answers on the 7-notch semantic differential scale into an integer number between -3 and 3, with negative numbers from -1 to -3 corresponding to the left-side adjective of the adjective pair, and positive numbers from 1 to 3 corresponding to the right-side adjective of the adjective pair. 0 was used to convey an option where participant found the system to correspond equally to both adjectives, or to not correspond to either of them. Thus, a total of 21 numbers were extracted for each participant, one number for each adjective pair. Mean averages of these numbers for the first

usability study can be found from figure A.5 on page 96 as the graph "Old frontend".

Mean averages were also calculated for each of the four qualities represented by AttrakDiff: pragmatic quality, hedonic quality - identity, hedonic quality - stimulation and attractiveness. When calculating means for qualities some of the adjective pairs were flipped so that each adjective pair had the adjective generally interpreted as "worse" in the negative number area and the "better" one in the positive number area. The calculated mean averages, along with standard deviations and confidence intervals calculated at the 95% confidence level, can be found from table 5.1.

SUS values were scored according to the questionnaire's instructions [4]. Mean average of 79.62 was calculated for SUS scored. See table 5.1.

Table 5.1: Mean AttrakDiff and SUS results from the first usability study. (AttrakDiff: Min. = -3/Max. = 3; SUS: Min. = 0/Max. = 100; N = 13; CI = 95%. SD: standard deviation, CI: confidence interval, Lower: lower confidence bound, Upper: upper confidence bound.)

| Scale                                     | Mean  | SD    | CI   | Lower | Upper |
|---|-------|-------|------|-------|-------|
| AttrakDiff: Pragmatic quality             | 1.04  | 0.91  | 0.55 | 0.49  | 1.60  |
| AttrakDiff: Hedonic quality - identity    | 1.01  | 0.33  | 0.20 | 0.81  | 1.21  |
| AttrakDiff: Hedonic quality - stimulation | 0.71  | 0.90  | 0.54 | 0.17  | 1.26  |
| AttrakDiff: Attractiveness                | 1.58  | 0.61  | 0.37 | 1.21  | 1.95  |
| SUS                                       | 79.62 | 12.54 | 7.57 | 72.04 | 87.19 |

The mean scores point out that the frontend was considered somewhat above average in all areas. According to AttrakDiff, "attractiveness" was considered the most high-ranking aspect of user experience in the ASP.NET frontend, scoring a mean of 1.58. The lowest ranking went for hedonic quality - stimulation, scoring 0.71.

These findings are compared against quantitative findings collected from the second usability study in chapter 9 starting on page 75.

### 5.3 Qualitative Findings

Qualitative usability findings were collected both during usability sessions and later on when going through the notes and recorded video tapes. Findings were collected into an Excel document where they were given an identification number, a description, a recommended fix, textual tags to link them to an area of the tested product for easier finding and filtering (such as

"Workflow", "Home page" or "Test Browser"), severity rankings, and notes in how many and in which usability sessions these findings were encountered, which helped see how wide-spread the issue presented in the finding was.

A total of 134 qualitative findings were collected from the sessions. The issues were given rankings according to their *frequency* (how many participants encountered the issue, from 0 to 13), *impact* (whether the issue was difficult or easy to overcome, from 1 - easy to 3 - difficult) and *persistence* (how often users would continue be bothered by the issue even once they knew about it, from 1 - one-time issue to 3 - users would encounter it every time).

In addition, the issues were also ranked an overall severity level according to Nielsen's levels of severity [19]:

- 0 Not a usability issue
- 1 Cosmetic issue
- 2 Minor usability issue
- 3 Major usability issue
- 4 Catastrophic usability issue

The 134 findings were ranked into the severity levels like thus:

- 8 level-0 non-usability issues.
- 62 level-1 cosmetic issues.
- 40 level-2 minor usability issues.
- 18 level-3 major usability issues.
- 6 level-4 catastrophic usability issues.

Furthermore, a good portion of the findings could be classified as being specific to a page (or several pages) of the site. These findings have been collected and sorted by high-level severity into table 5.2. Findings that do not belong to any specific page have also been collected into table. Some findings were deemed to belong to several pages. This is reflected in the total finding counts in the table, which sum up to more than the total 134 findings.

While describing all of the 134 findings is out of scope of this thesis, an overview of the findings found during the first evaluation round, grouped

Table 5.2: Usability finding counts per site page for the first usability study. Table is sorted by amount of high-level findings.

| Page                     | Level-4 | Level-3 | Level-2 | Level-1 | Level-0 | Total |
|--------------------------|---------|---------|---------|---------|---------|-------|
| Test Browser             | 3       | 4       | 7       | 9       | 4       | 27    |
| Test Executions          | 1       | 3       | 7       | 7       | 1       | 19    |
| Workflow Manager         | 1       | 3       | 3       | 8       | 0       | 15    |
| Workflow Instance        | 1       | 2       | 12      | 4       | 0       | 19    |
| Workflow Template Editor | 0       | 4       | 5       | 16      | 3       | 28    |
| (no page)                | 0       | 3       | 6       | 10      | 0       | 19    |
| Systems under Test       | 0       | 1       | 0       | 4       | 0       | 5     |
| Home Page                | 0       | 0       | 2       | 8       | 1       | 11    |

by site page, is collected below. As some of these findings were eventually addressed for the React version of the frontend, the findings in question and details on how they were fixed are presented in more detail in chapter 6.2 starting on page 53.

### 5.3.1 Home Page

Many of the findings collected from the home page revolved around unclear terminology. That these issues were found on this particular page is unsurprising considering that the first task in the scenarios presented to participants involved looking at and reading through the home page.

Seven out of thirteen of the participants had some degree of initial uncertainty with what terms such as *system under test* or *workflow* really meant, and six participants of the thirteen were unsure of what sort of functionality was supposed to be found under pages titled *Keyword Designer* and *Test Browser*. This can be viewed as external inconsistency in the conceptual component of the interface of Execution Framework – that is, the metaphors used to describe tasks and actions in the system were inconsistent with what participants were expecting considering their previous experience with the old AUIT system, and from having used other systems with domain similarities.

Another common issue was that the home page offered three different links for each of the main components of the site (Keyword Designer, Test Browser, Workflow Manager and Systems under Test). All these components were presented on the home page as dashboard-style icons with the title of the page, a short description of the page, and a button underneath. The title of the page was styled as a caption for the description text and worked

as a link. The button underneath, likewise, was a link, titled as a verb action ("Open Test Browser"). Finally, a top-bar is drawn on top of the page with navigational links for these same components. See figures 3.1 and 3.2 on page 26. All these three different links worked identically, but some participants were confused whether or not they were being taken to the same page by them. Two of the participants went as far as testing each of the links one by one, making sure they were being redirected to the same page from all of them.

This issue can be categorised under external physical inconsistency – the visual representation of the link elements was not consistent with expectations of participants.

### 5.3.2 Test Browser

The most severe findings in Test Browser were related to participants struggling with side effects of the hierarchical organisation of the tests in a tree structure. Test Browser did not support searching through or sorting the test hierarchy, and since tests in the browser were presented as a hierarchical tree with nodes that could be expanded and collapsed, participants could not use browser's built-in search features either to find the tests they were looking for unless they had already expanded the relevant nodes in the tree. These issues were compounded by a bug that caused the Test Browser to forget which nodes the participant had already expanded when using browser's back functionality to return to the Test Browser from a subpage, forcing the participant to start browsing from scratch. These issues with interaction were categorised under internal (the bug) and external (lack of search) communicational inconsistency.

Participants also wished that overviews of passed/failed states of tests would be presented on parent nodes of the tests, so that a non-test parent node would be shown as "passed" if all the actual tests under it would have passed, and as "failed" if any of the tests under the node would have failed. This sort of information was not available for non-test nodes, making finding failed tests from the hierarchy by just browsing it very cumbersome, requiring the participant to expand every node in the tree.

Some participants had trouble with the way UI required them to choose an active system under test in the Test Browser before they were shown any builds and tests. As with the terminology issues with home page (see page 43), participants did not always have a clear idea how systems under test connected with builds, although experimentation generally allowed them to continue. The dropdown component used to select from the systems under test was also quite small and sometimes took a while for the participants

to notice. One participant also expressed a valid concern on how tall the dropdown might grow once the system has been used for several years and the amount of systems under test has grown large. See figure 5.1.

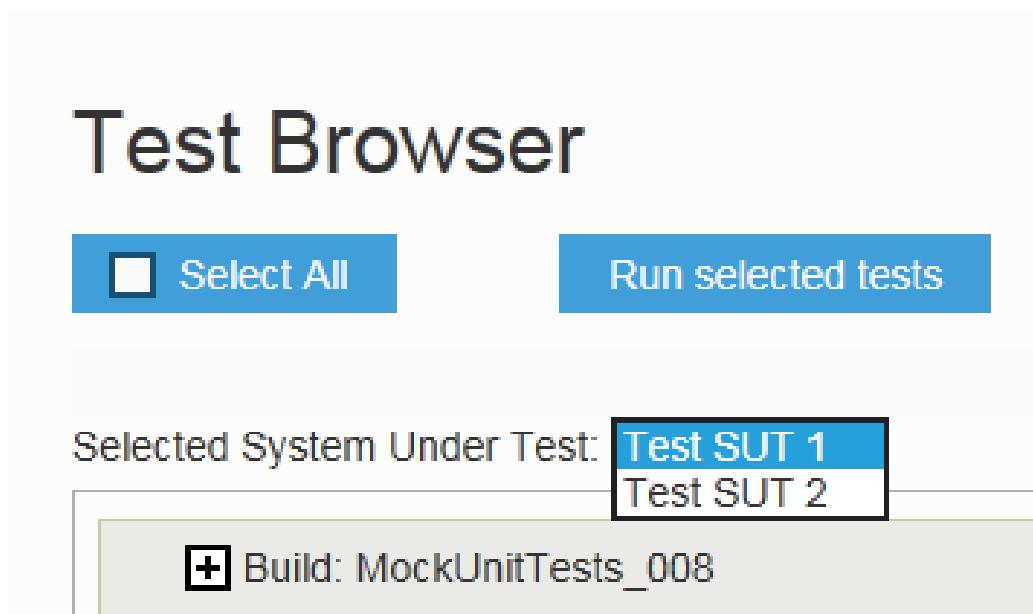


Figure 5.1: Some participants had trouble with either locating the small system under test selection dropdown on Test Browser or figuring out how it was connected to the displayed test hierarchies.

### 5.3.3 Workflow Manager

Six out of the thirteen participants noted that while the Workflow Manager page displayed current test progress of running workflow instances, it only presented this information as how many of the tests in an instance had been run out of the total test count. Participants would have preferred to see numbers of passed and failed tests in workflow instances to better gauge how well these workflow instances had been going. The progress of finished tests within a workflow instance that *was* displayed was also found somewhat confusing by some participants, as it was presented not only as a graphical progress bar, but in addition both as a number ("13/40") and as a text string ("Finished", "Running").

Workflow Manager did also not offer participants any good abilities to filter, sort, search or group workflow templates and workflow instances in the

system, beyond using their internet browser's built-in search functionality and by workflow templates being automatically sorted into sections of "active", "inactive" and "manually queued" workflow templates. Three participants had trouble finding a workflow they were looking for from the Workflow Manager and explicitly noted the lack of search functionality. This issue became increasingly relevant during later usability sessions, as the interface did not directly support removing workflow templates from the system and one of the scenario tasks was to create a new one into the system. This caused a slow build-up of new workflow templates in the system as the first usability study progressed further, causing increasing visual noise in the Workflow Manager. These new workflow templates were set to "inactive" after each usability session, causing them to fall to the bottom of the Workflow Manager page. While they were out of the way of scenario tasks, later participants did comment on the clutter, raising this issue.

### 5.3.4 Workflow Instance

The most common finding during the usability study was every participant but one expecting the workflow instance page to automatically refresh its dynamic contents, whereas this implementation was mostly a static page and required the participant to manually refresh the page to update the presented state of the workflow instance.

As with the Workflow Manager page, the page for workflow instances also did not offer a good summary of passed and failed tests within it. While the actual tests and test runs with their final results were available on the page, their total counts were not, and participants wanting summaries of workflow instances would have had to count the amounts of passed and failed tests by hand.

Seven minor findings were also collected from the table on the page that was used to present tests and test runs in the opened workflow instance. For one, when participants used the sorting functionality in the table, the sorting tended to respond slowly and often caused participants to erroneously click on the sort buttons multiple times, ending up with sorting calls for the table being queued up sequentially, making the table sort itself multiple times in quick succession after a small delay. However most participants did not even try sorting the table, even when doing so would have made their task considerably easier. It is unclear if they did not know that the feature was available or whether it was just part of their usual way of working with tabular data.

### 5.3.5 Test Executions

Eight of the thirteen participants fundamentally misinterpreted the test executions page. While the page was meant to collect all test executions of a single test in one build into one page, these participants misinterpreted it either as showing test runs of a test across several builds and even across systems under test, or as displaying all test runs of a single workflow instance (which would have made it just an alternative view of the workflow instance page).

In a case where a participant entered the test executions page from an older workflow instance it was also unclear which of the displayed test executions was the one that brought them to the page.

Three findings originated from the system limitation of how test result logs are handled. These logs are stored as complicated set of files on another server on the company's internal network, with a single XML file for each test result that the user can open. Due to default security settings in most internet browsers, these files cannot usually be opened directly by following a link from FAST's frontend. Rather, the user is required to copy a link to the test log file into clipboard and paste it into browser's address bar. This used solution was extremely cumbersome which perplexed three participants. While the page had a separate button for copying the URL to the file into user's clipboard in a single click (see figure 5.2), the page did not contain sufficient instructions on why this was necessary and how to actually view the test logs.



Figure 5.2: As a workaround for accessing test results logs on Test Executions page, a separate button for copying link to the test log into user's clipboard was provided on the right side of each test execution item.

### 5.3.6 Workflow Template Editor

As with findings for Test Browser (chapter 5.3.2, page 44) and Workflow Manager (chapter 5.3.3, page 45) the widget used to select test paths for a workflow template did not support filtering, searching or sorting, causing some participants to have a degree of trouble with finding and selecting tests and test paths for their workflow template. This also caused external communicational inconsistency with what participants would have expected

from a this kind of interface, although technically it was internally consistent with the Test Browser component.

The concept of "test paths", essentially just folder wildcards for tests, was also somewhat foreign to participants, although most of them understood the concept after using the system for a bit.

Another major issue that almost every participant encountered was the lack of feedback once a workflow template was saved. Participants expected either a confirmation message that their save was successful, or at least that they would have been redirected to either Workflow Manager or to some kind of other view of the workflow template, but in fact nothing happened. Some participants doubled back to Workflow Manager to see if their new template had appeared there, to make sure that it had been properly created or its changes saved. This issue was both an external and internal inconsistency in communicational aspects of the interface – not only were participants expecting proper feedback from saving their workflow, but, as an internal inconsistency issue, proper feedback was nonetheless given when taking other similarly important actions such as queueing tests for a run.

Five out of the thirteen participants also found that presenting a workflow template's priority as an integer number was confusing as they had no context to judge what was a good number to pick. While a help text to the right of the form field described that lower priority numbers meant higher priority, this did not give enough context for the participants to adjust this number for their own use. The default number for the field, 1, was also deemed oddly high priority for a default value by one participant. Interestingly, this representation of priority as an integer was externally consistent with the physical component of a similar priority feature in the old AUIT interface. However, adjusting priorities of test runs was generally performed only by power users in the old AUIT system, and most of the old AUIT users taking part in this evaluation were merely regular, non-power users. The difference in metaphors in workflows in Execution Framework and test runs in the old AUIT system may also have caused confusion in how priority was handled.

Many participants were also confused by some form component issues that were caused by aggressive future-proofing and edge case handling. A radio button selection component was used for selecting the "test selector" used in the workflow template, but the only available radio option was "test paths" as nothing else had been implemented. Also, when user selects a *build system* used for the workflow template, they are also prompted to select a *build source*. These were both presented as dropdown components, but as the available build sources are directly linked to the used build system, the build source dropdown generally only had one possible option for the participant to choose from. The system technically supports multiple build sources in

a build system so user had to be given an option to choose one of multiple sources from the system, but this is a rare case and it did not occur during this usability study. These issues caused some confusion, especially as the concepts of test selectors and build sources were not familiar to participants.

### 5.3.7 Systems under Test

Not many findings were uncovered from the Systems under Test page as the scenario tasks did not require participants to enter it. The most pressing issue found was that Test Executions pages contained links to the system under test page that the test was contained in. A degree of risk was involved in this as user could modify the system under test directly from this linked page. Two participants followed this link half-accidentally, and while they did not modify any of the systems under test, they could have damaged the system from this page without realising it.

### 5.3.8 Other Findings

Other findings that weren't classifiable under any specific page of the site contain among others the inability for users to manually instantiate a workflow instance out of a workflow template. This is something three of the participants wished was possible to do.

Five of the participants also had issues with navigation across pages. Several of them were afraid to use their internet browser's back button to navigate back from a page. At least one participant mentioned that they had bad experiences with using browser's back functionality with modern dynamic web pages (a sentiment that feels anecdotally common), which made them wary of using it in other sites that seemed similarly modern or dynamic. Two of them also found having navigated themselves into dead-ends, not being sure where to navigate next, and hoping for explicit "back" links or breadcrumbs to allow them to step back.

## Chapter 6

# Building the React Frontend

This chapter presents the chosen technology and architecture for the new, React-based frontend. Following this, several findings from the first usability study are described in detail, including how they are categorised within consistency of the frontend's user experience, and how the finding was improved in the React implementation.

### 6.1 Architecture

One goal of this entire project was to move Execution Framework's frontend from combination of static HTML pages and cobbled-together dynamic JavaScript and JQuery code to take advantage of an actual JavaScript frontend framework or library. One frustration with the original ASP.NET frontend code was its ad hoc, inconsistent nature, which also made constructing consistent interface elements difficult. The hope was that changing to a complete frontend library solution would also make constructing a consistent interface easier, by better supporting writing reusable interface components. This would help in cutting down inconsistencies in communicational and physical areas of the interface.

The change to a new frontend library, however, would require rewriting a lot of user interface code. A lot of the code written for the server-side ASP.NET backend would also need to be changed into client-side JavaScript code. Setting up a new frontend framework, its toolchain, and work process, and getting them all to interact with the existing Visual Studio-based development workflow would also require time and work. All this meant that the actual time reserved for usability and user experience improvements for the findings covered in chapter 5 would be limited and would have to be prioritised.

### 6.1.1 React As the New Frontend Library

Several JavaScript-based frontend frameworks were considered to replace the previous frontend architecture. *AngularJS* was decided to be too different from writing normal JavaScript. Coding in Angular would teach developers how to code in Angular, but not so much in JavaScript. It was felt that this would have made maintenance tasks more difficult in a multi-person development team where most of the developers did not have much experience with JavaScript programming to begin with. *Meteor* was also considered, but it would have been an overly holistic approach to frontend development, offering its own backend implementation. This was decided to be a no-go as the existing ASP.NET infrastructure could easily be changed into a pure API backend for the new frontend implementation. The level of online support for Meteor was also found inadequate.

Eventually *React* was chosen as the frontend library of choice. Unlike other considered frameworks, React is only a renderer library. It was backed up with other JavaScript libraries such as *react-router* and *redux* to handle URL routing and application state.

Expectations for React were that it would have a learning curve, especially for non-JavaScript developers, but making modifications for existing React code would be relatively easy. However, as stated in chapter 1.2 on page 14, maintainability of the code base was not a focus in this study, nor was the project on-going for long enough to see how non-JavaScript developers of the team would find the chosen library.

The idea was to also turn the previous ASP.NET backend into a JSON API. This was done by stripping out all the code generating HTML views and replacing these with API calls returning data from the Execution Framework in JSON format. This data was then parsed and displayed by the React frontend.

### 6.1.2 Focus on Workflow Pages

As previously stated, due to time limitations for the project and the decision to change to a new frontend architecture, it was necessary to prioritise and focus work effort when re-creating and improving pages and site functionality for the new frontend. This work started with combining findings from the first usability study into four distinct areas of the site: the home page, systems under test pages, the test browser, and workflow-related pages. These collections are displayed in table 6.1. Sorting this table by the number of high-severity findings it's clear that workflow-related pages had the most findings requiring attention, having 3 "usability catastrophe"-level findings

and 12 "major usability problem"-level findings.

Table 6.1: Findings of first usability study combined into four distinct areas.

| Page               | Level-4 | Level-3 | Level-2 | Level-1 | Level-0 | Total |
|--------------------|---------|---------|---------|---------|---------|-------|
| Workflow Pages     | 3       | 12      | 27      | 35      | 4       | 81    |
| Test Browser       | 3       | 4       | 7       | 9       | 4       | 27    |
| Systems under Test | 0       | 1       | 0       | 4       | 0       | 5     |
| Home Page          | 0       | 0       | 2       | 8       | 1       | 11    |

The pages for handling systems under test offered relatively simple create, remove, update and delete actions, which were not particularly interesting targets to focus on immediately. These pages also did and likely would not see much frequent use, making them low priority overall.

The Test Browser, on the other hand, was a relatively complicated component, and prioritising the efforts to re-create it in React would not necessarily be justified as the component seemed to see only moderately frequent use among users actively using the Execution Framework.

The workflow pages, however, saw basically daily use by the active users. This alone was a strong justification for focusing on these pages.

The pages that were finally chosen for re-implementation and improvements were the Home page and several workflow-related pages, namely the Workflow Manager page and the Workflow Instance page.

The home page was chosen for its simplicity and static nature. It was easy to implement again in React, and also offered a good starting point for the scenarios that would be written for the second round of usability tests. The final home page ended up looking identical to the ASP.NET frontend home page which was presented in chapter 3.1.1 on page 25.

The Workflow Manager was chosen for being the obvious "root" page of all workflow-related actions currently available on the site. It was also deemed the page most likely to see daily use of all the workflow-related pages. The final design is presented in appendix figure B.1 on page 98. For comparison the old ASP.NET design is presented in figure B.2 on page 98.

The Workflow Instance page was chosen for its ideal nature as the target for implementing auto-refreshing of data from backend to frontend consistently across the site, relating to the issue described in chapter 5.3.4 (page 46) and the following chapter 6.2.3 (page 57). The final design is presented in appendix figure B.3 on page 99. For comparison the old ASP.NET design is presented in figure B.4 on page 99.

The Test Executions page, despite having the most findings from the workflow-related pages as seen in table 5.2 on page 43, was dropped out of

scope of the improvements for the React frontend. The original plan was to re-imagine the Test Executions page as a page displaying runs of a specific test across all builds, like many participants mis-interpreted the page (as described in chapter 5.3.5 on page 47). These major modifications pushed the page out of the time constraints allotted for the project, however. The page was also prioritised as less important than the Workflow Instance page, as the latter was still navigationally an obvious position for entering the re-imagined Test Executions page and would thus be best to construct first.

Adding the actual improvements, as described in chapter 6.2, was relatively straightforward. The previous ASP.NET frontend was re-created fairly identically in React, after which incremental improvements for usability findings were added on top. Developing these improvements was, as predicted, a much simpler affair than it would have been with the old ASP.NET/JQuery-based frontend.

## 6.2 Improvements

In total, due to the scope of the new React frontend being much smaller than the tested previous ASP.NET frontend, only 10 of the 134 found usability issues were addressed. From these 10 two were level-4 "usability catastrophes", two were level-3 "major usability issues", three were level-2 "minor usability issues", and the last three were level-1 "cosmetic issues".

The issues had been ranked according to their severity which was taken into account when choosing which issues to handle when constructing the React frontend. These rankings are presented below. See chapter 5.3 on page 42 for explanations on the different rankings used.

Findings had also been linked to dimensions of consistency, if appropriate. These are also presented below.

Finally, the implemented fixes and improvements are presented.

### 6.2.1 No Overall Result Summary for Workflow Instances

Description of the finding from first round of usability evaluations:

Finding #48: No overall summary of the amount of passed and failed tests within a workflow instance is available on the workflow instance page. User can only see every individual test in the workflow instance and the state of these tests, but would currently need to count the total number of passed and failed tests by hand.

This finding was ranked as follows:

**Frequency: 3/13** 3 out of 13 participants encountered this issue.

**Impact: 3/3** This issue was difficult to overcome. The workflow instance page allows filtering and sorting of tests by their state, but users would still need to manually count the amount of tests in each category.

**Persistence: 3/3** Users would encounter this issue every time on the workflow instance page.

**Severity: 4/4** Total severity was ranked as a "usability catastrophe" for its severe impact and persistence. The issue would have a major effect on basic use of the application, despite it not having occurred very frequently during testing sessions.

This finding had no connection to internal consistency of the system, but was externally inconsistent with the old AUIT system which prominently displayed counts of passed and failed tests within each test group. It can be argued that this issue relates to both physical and communicational components of interface consistency – the expected visual representation of results was completely lacking in the current system (physical component), and the system did also not adequately display the state of an item even with user accessing the item directly (communicational component).

The finding was fixed by adding new columns for numbers of passed, failed, in-progress and cancelled tests into the top summary section of Workflow Instance page. See figure 6.1. Due to backend limitations these numbers were relatively slow to calculate. To increase page load times this section of the page is empty when user first enters the page, but the summary content is filled in dynamically within a couple of seconds as user stays on the page.

| Workflow instance: Test SUT 1 workflow, build MockUnitTests_004 |                   |                            |                   |
|---|-------------------|----------------------------|-------------------|
| <b>Build:</b>   | MockUnitTests_004 | <b>System under test:</b>  | Test SUT 1        |
| <b>Run state:</b>   | Finished          | <b>In-progress tests:</b>  | 0/18              |
| <b>Passed tests:</b>  | 12/18             | <b>Failed tests tests:</b> | 6/18              |
| <b>Start time:</b>  | 10 Jun 2016 15:17 | <b>Finish time:</b>        | 10 Jun 2016 15:34 |
| <b>Description:</b>   |                   | <b>Requested by:</b>       |                   |
|   |                   | <b>Finished tests:</b>     | 18/18             |
|   |                   | <b>Canceled tests:</b>     | 0/18              |
|   |                   | <b>Total run time:</b>     | 16 min, 59 sec    |

Figure 6.1: Numbers of in-progress, finished, passed, failed, and cancelled tests were added to the summary section on the top of the workflow instance page.

As a side effect of implementing a fix for finding #1 (see page 57) the summary section on the page is also periodically updated with actual latest state data from backend as long as user keeps the page open.

### 6.2.2 No Result Summary Available in Workflow Manager

Description of the finding from first round of usability evaluations:

Finding #57: Overall passed/failed summary of workflow instances is not given on the Workflow Manager page. Number of passed or failed tests within the workflow instance is not shown either.

This finding was ranked as follows:

**Frequency: 6/13** 6 out of 13 participants encountered this issue.

**Impact: 3/3** This issue was difficult to overcome. Participants were uncertain of states of workflow instances in the system as the Workflow Manager page was the only page that gave an overview of all workflow instances in the system. The Workflow Manager page showed number of tests run in each instance, but not how well the actual tests had gone. Participants had no choice but to open workflow instances manually and see the state of the test runs within from inside them.

**Persistence: 3/3** Users would encounter this issue each time.

**Severity: 4/4** Total severity was ranked as a "usability catastrophe" for its severe impact and persistence, and quite frequent appearance in testing sessions.

A vague connection could be made from this finding to external consistency in that good system design would involve showing summaries of critical sub-data in a top-level view, and usability testing participants were either expecting or requesting this kind of functionality. As with the previous finding #48 (see page 53), this was an issue in both physical and communicational aspects of the interface.

This finding was addressed by replacing the ambiguous progress bars of workflow instances (see figure 6.2 and the separate finding #89 on page 59) with new columns for amounts of passed, failed and in-progress tests, similar to the previous finding #48. See figure 6.3. To further aid in user being able to quickly ascertain the current state of workflow instances, some visual

emphasis was added to each field whenever the count in them would be over zero: the text colour is changed to green for passed tests, red for failed tests, and left as the default black but bolded for in-progress tests.

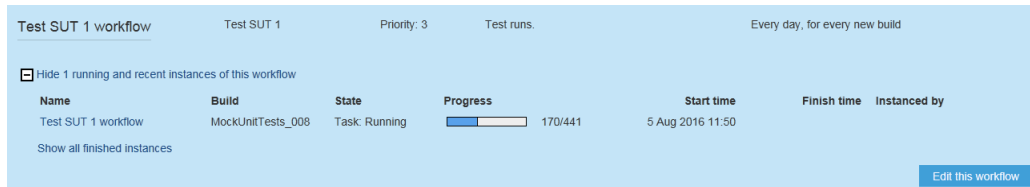


Figure 6.2: Participants had trouble connecting the progress bar and the numbers to the right of it. Numbers of passed or failed tests was also not displayed.

| Name                | Build                       | State    | In-progress | Passed  | Failed  |
|---------------------|-----------------------------|----------|-------------|---------|---------|
| Test SUT 1 workflow | MockUnitTests_008           | Error    | 0/441       | 80/441  | 95/441  |
| Test SUT 1 workflow | MockUnitTests_005           | Finished | 0/490       | 249/490 | 237/490 |
| Test SUT 1 workflow | MockUnitTests_004           | Finished | 0/18        | 12/18   | 6/18    |
| Test SUT 1 workflow | MockUnitTests_003           | Finished | 0/9         | 9/9     | 0/9     |
| Test SUT 1 workflow | mstestsharp_9_6_2016 - Copy | Finished | 0/4         | 0/4     | 4/4     |
| Test SUT 1 workflow | mstestsharp_9_6_2016        | Finished | 0/4         | 1/4     | 3/4     |
| Test SUT 1 workflow | mstestsharp_8_6_2016        | Finished | 0/4         | 1/4     | 3/4     |
| Test SUT 1 workflow | mstestsharp_2_6_2016        | Finished | 0/4         | 1/4     | 3/4     |

Figure 6.3: The progress bar was replaced by listing the counts of passed, failed, and in-progress tests separately.

To save visual space on the page, the fourth regular state of tests, "cancelled", was left out of the table, even though this created some slight internal physical inconsistency with the columns on the workflow instance page introduced in the fix for the similar finding #48. The number of cancelled tests can still be inferred, however, as the amount of tests left over when passed, failed and in-progress tests are subtracted from total number of tests in a workflow instance.

As with the fix for finding #48, the calculation of the data required for displaying this information is relatively expensive to compute. Thus, the data cells for the passed, failed and in-progress items are initially loaded on the web page as empty fields, and the actual data is loaded in dynamically within a couple of moments. Depending on the amount of items on the page this can take several seconds.

Also as with finding #48, the displayed information is periodically updated with latest result data from backend (but unlike with the Workflow

Instance page in finding #48, automatic background refreshes had already been implemented in the previous version of the Workflow Manager page, the subject of this finding).

### 6.2.3 Workflow Instance Page Does Not Auto-Refresh With Latest data

Description of the finding from first round of usability evaluations:

Finding #1: Participants expected the Workflow Instance page to automatically refresh itself with latest progress data and results of test runs in background. However, this had not been implemented and participants were required to manually refresh the page to see current data and results. One participant explained that they expected the page to update itself because a similar view of test runs and their results in the desktop application Visual Studio updated itself automatically without manual intervention, as desktop applications generally do.

This finding was ranked as follows:

**Frequency: 12/13** 12 out of 13 participants encountered this issue.

**Impact: 2/3** This issue was rated as moderately difficult to overcome. Without the ability to easily see whether anything was actually happening or not on the page, participants generally waited for some time before either asking whether the page auto-updated itself or refreshing the page manually.

**Persistence: 1/3** Participants learned to bypass the issue by refreshing the page manually after encountering the issue for the first time.

**Severity: 3/4** Total severity was ranked as a "major usability problem", mainly due to its frequent appearance in almost every testing session.

The finding can be seen linked to external communicational consistency in that modern web applications generally do implement background auto-refreshing for updating user's screen with current data relevant to them. Also, as mentioned by one of the participants, other software used by the participants designed to monitor test runs, web-based or not, implement auto-refreshing.

A link to internal consistency also exists: some pages of the previous version of the application already implemented auto-refreshing, in particularly on the Workflow Manager page.

The finding was fixed by implementing automatic refreshing on the Workflow Instance page. Notably, the new ReactJS architecture made the implementation more straightforward than it would have been with the previous version. After the fix the page polls updated data from the backend every 10 to 15 seconds. A horizontal spinner is painted by top of the data table whenever the page is performing an auto-refresh operation. See figure 6.4.

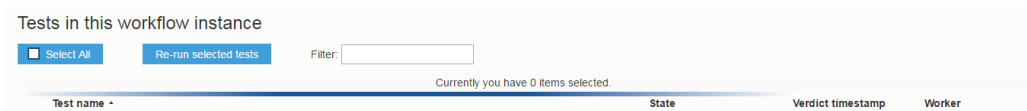


Figure 6.4: A horizontal spinner, in the shape of a moving bar with a blue gradient colour on it, is now briefly visible when the workflow instance page auto-refreshes itself.

## 6.2.4 Workflow Manager Page Has No Column Titles

Description of the finding from first round of usability evaluations:

Finding #66: Workflow templates in Workflow Manager have no labels for their columns. This makes deciphering some of the fields guesswork if the field hasn't been particularly well filled.

This finding was ranked as follows:

**Frequency: 2/13** 2 out of 13 participants encountered this issue.

**Impact: 3/3** This issue was difficult to overcome if fields of the workflow template were filled so that they weren't very self-evident.

**Persistence: 2/3** Users might be able to learn which unlabelled column would match with which field of the workflow template, but this would put a lot of burden on them.

**Severity: 3/4** Total severity was ranked as a "major usability problem" due to high impact and moderate persistence.

This finding conflicts with internal physical consistency in that all other columns in all other tables in the frontend have been labelled, although the Workflow Manager is not such an obvious table element as these other tables.

See figure 6.5 for an example of the issue from the ASP.NET frontend. This finding was fixed by adding labels "template name", "system under test", "priority", "description", and "schedule" for the template items in Workflow Manager. See figure 6.6.

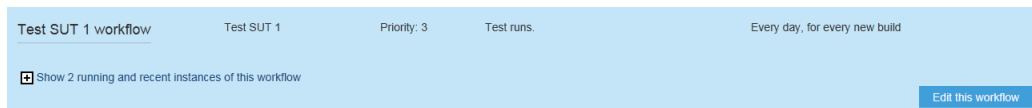


Figure 6.5: Previous version of the frontend did not have column labels for workflow template items in Workflow Manager.

| Template name       | System under test | Priority | Description | Schedule                       |
|---------------------|-------------------|----------|-------------|--------------------------------|
| Test SUT 1 workflow | Test SUT 1        | 3        | Test runs.  | Every day, for every new build |

Figure 6.6: Column labels were added to the workflow template items.

### 6.2.5 Workflow Manager Has Three Different Ways of Conveying Workflow Instance Progress

Description of the finding from first round of usability evaluations:

Finding #89: A participants expressed some confusion by the three different ways that Workflow Manager used to convey test run progress for workflow templates: a progress bar, a numerical representation of the progress bar ("13/40"), and a textual state ("Finished").

This finding was ranked as follows:

**Frequency: 1/13** 1 out of 13 participants encountered this issue.

**Impact: 2/3** This issue was moderately difficult to overcome when the participant had no mental model on how the three different ways of expressing the same state mapped together.

**Persistence: 2/3** Participant was able to correctly parse the information in the three different fields together after they had asked and were told about it.

**Severity: 2/4** Total severity was ranked as a "minor usability problem".

This finding has a vague connection to external physical usability in that most third-party systems attempt to display state or progress of a task with only just a progress bar, or a progress bar and a textual summary of the state, and sometimes with a numerical expression of the values used in the progress bar. This would conform with how this system has been implemented, but the connection between these three items was not obvious enough. The numerical representation of the progress bar ("13/40") was not positioned quite close enough to the progress bar itself to be obviously linked to it. The textual representation of the progress likewise wasn't close enough to the progress bar, and was easy to misinterpret as not being related to it.

In the React frontend this finding was fixed by completely removing the progress bar and its numerical representation, replacing it with the columns for the number of passed, failed and in-progress tests in a workflow instance implemented for finding #57 (see page 55). This still left the textual representation of the state easy to misinterpret as not being related to the test progress numbers, but this was deemed a sufficient improvement for the next iteration.

See figures 6.2 and 6.3 on page 56 for a before and after comparison.

A further fix would be to implement a multi-colour progress bar that would show overall progress of test runs, and number of passed, failed, cancelled and in-progress tests as differently coloured segments on the bar. Numeric representations of these should also be introduced again, but put into a tooltip that gets displayed when user hovers over the progress bar element.

### 6.2.6 Considerable Lag When Sorting Tables

Description of the finding from first round of usability evaluations:

Finding #19: Sorting tables in Workflow Instance page by clicking on table headers has considerable lag before the table actually gets visually sorted. This can cause a case where user thinks their first click on the table header was not registered while in fact the sort is being processed in the background. This may cause user to click on the header multiple times, which ends up having all of the individual sort clicks getting registered and processed and the table getting sorted back and forth multiple times against user's wishes after a small delay, disorienting and annoying them.

This finding was ranked as follows:

**Frequency: 2/13** 2 out of 13 participants encountered this issue.

**Impact: 2/3** This issue was moderately difficult to overcome, as it was difficult for the participants to predict.

**Persistence: 2/3** Participants tended to encounter this issue multiple times, not being able to infer what caused the occasional lag in performance.

**Severity: 2/4** Total severity was ranked as a "minor usability problem".

The finding has a link to external consistency in that most programs the participants would use daily that also had sortable tables would be desktop applications with efficient client-side sorting. This finding is also linked to communicational component of interface in that the inconsistency stems from the delay and uncertainty in user interacting with the system.

The root cause for the lag was table sorting happening on the server: whenever user wanted to sort a table, a new table for the current data was constructed on the server, sorted, and converted into HTML which was sent back to the client (user's browser). This caused delay even when sorting small tables. User, not knowing that the sorting operation they wanted was already being performed, could click on the table headers to make sure the table was being sorted. This would queue up the sort requests, all of which would eventually get performed, resulting in the table being re-drawn rapidly and chaotically, alternating between ascending and descending sort, possibly also ending up in a sort order that would be against user's wishes.

This was fixed by moving the table sort functionality into client-side code. The new sort is much faster, although it can still cause adverse performance when sorting tables with large amounts of items (one hundred or more). This is not actually due to the sorting, but rather because of having to re-draw the entire table from scratch in browser's DOM. On the upside this now causes the browser's UI to freeze momentarily when re-drawing the DOM, effectively stopping user from performing unwanted multiple clicks on table headers (the DOM re-draw freeze occurred also in the previous implementation, but was eclipsed by the non-freezing wait time caused by network lag).

### 6.2.7 Tables Do Not Look Sortable

Description of the finding from first round of usability evaluations:

Finding #50: Most participants did not attempt to sort tables in Workflow Instance or Test Executions pages, even when it would have quite obviously made their task easier. At least one

participant wondered out loud if the table was going to be sortable before they attempted to sort it by clicking on one of the table's column headers. The assumption is that for at least some people the table is not obviously sortable, as tables on the web often aren't.

This finding was ranked as follows:

**Frequency: 1/13** 1 out of 13 participants encountered this issue.

**Impact: 2/3** This issue had moderate impact on using the site.

**Persistence: 1/3** The table was easy to sort once a participant knew it was possible.

**Severity: 2/4** Total severity was ranked as a "minor usability problem".

The finding was connected to external physical consistency in that sortable tables generally have a default column they are sorted by, and an arrow to indicate what the default sort is. This implementation, while supporting sorting and displaying arrows once sorting had been performed, offered no default sort or default arrows.

The issue was fixed by adding a default sort column and the related arrow to tables. See figure 6.7.

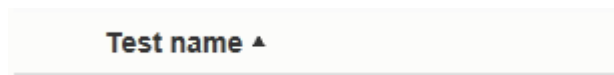


Figure 6.7: Table caption with a default sort column and an indicator arrow.

### 6.2.8 Misinterpreting Auto-Refresh Spinner

Description of the finding from first round of usability evaluations:

Finding #87: Workflow Manager page displays an animated spinner by the right side of a workflow template item when it's refreshing the item's contents from backend. One participant did not see any link between the page possibly auto-refreshing itself and the animation. Another told that they found the animation very distracting whenever it was on screen.

This finding was ranked as follows:

**Frequency: 2/13** 2 out of 13 participants encountered this issue.

**Impact: 1/3** This issue had low impact on using the site.

**Persistence: 2/3** Once participants knew what the spinner animation was for, they weren't as dumbfounded by it. One participant still found it distracting, however.

**Severity: 1/4** Total severity was ranked as a "cosmetic problem only".

The finding broke external physical consistency slightly in that animated spinners on the web tend to be circular, whereas this was an animation of moving vertical bars. See figure 6.8. External communicational consistency was also breached in that the mental model of at least one of the users did not connect the spinner with the page auto-refreshing itself.

The entire spinner was removed when building the new version of the frontend. This was justified by the circumstances surrounding finding #1 (see page 57) which showed that participants were expecting the workflow instance page to automatically refresh itself even when there were no spinners on it.

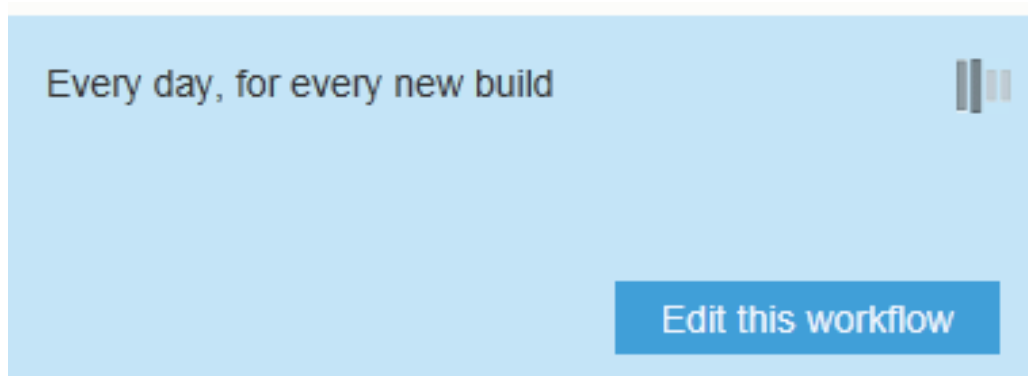


Figure 6.8: The occasionally appearing spinner on the top-right corner of a workflow template item caused some confusion.

### 6.2.9 Run Time for a Workflow Instance Is Displayed With Too Much Accuracy

Description of the finding from first round of usability evaluations:

Finding #91: Total run time of a workflow instance is displayed as a decimal number with many decimal points. This kind of accuracy is not only needless for human use, but also makes the numbers more difficult to parse.

This finding was ranked as follows:

**Frequency: 1/13** 1 out of 13 participants encountered this issue.

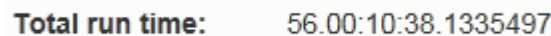
**Impact: 1/3** The issue had very little impact on use.

**Persistence: 3/3** Technically, users would encounter this issue every time they would have the need to get a total run time of a workflow instance.

**Severity: 1/4** Total severity was ranked as a "cosmetic problem only".

This finding was internally physically inconsistent in that some other time-related decimal numbers on the site were already being printed in a more human-readable format.

The finding was fixed by not only rounding run time to the nearest full second, but also by formatting it to separately display hours, minutes and seconds. See figures 6.9 and 6.10 for a before and after.



**Total run time:** 56.00:10:38.1335497

Figure 6.9: Previous version of total run time had needless decimal precision.



**Total run time:** 16 min, 59 sec

Figure 6.10: Total run time was rounded in the new version.

### 6.2.10 Undo Function In Table Is Confusing

Description of the finding from first round of usability evaluations:

Finding #37: The sortable tables which contain selectable items offer an "undo" function which is used to undo actions that mass-select (or mass-de-select) multiple items on the table. This is really only useful if user has already selected several items on the table and then accidentally clicks either the "clear all" button or

the top root checkbox that can be used to quickly select or de-select all items on the table. This undo feature was not used by any of the participants, and one participant expressed confusion on what it would actually "un-do".

This finding was ranked as follows:

**Frequency: 1/13** 1 out of 13 participants encountered this issue.

**Impact: 1/3** This issue had low impact on using the site.

**Persistence: 1/3** Users would have no issue with using the feature once learning it.

**Severity: 1/4** Total severity was ranked as a "cosmetic problem only".

While the finding does not strictly break any consistency standards, it's not a feature that is naturally seen with tables, so a vague link to external physical and communicational inconsistency can be argued.

See figure 6.11 for an example of the finding.

The finding was fixed by simply removing the undo option in the React frontend as it went completely unused in the previous version.

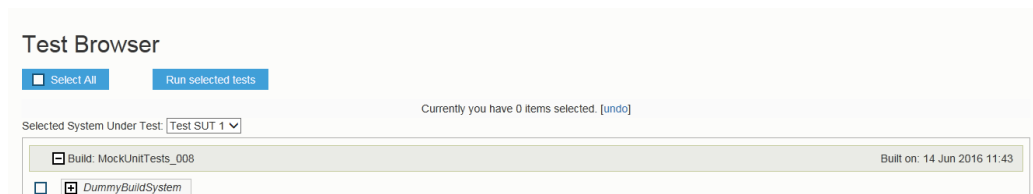


Figure 6.11: The "undo" link appearing next to the selection count indicator after mass-de-selecting all items on the page caused some confusion.

## Chapter 7

# Evaluating the React Frontend

This section explains how and with whom the new React frontend for the Execution Framework was usability tested, and which methods were used.

The gathered evaluation data is presented in chapter 8, starting on page 69.

### 7.1 Evaluation Goals

The main goal of the second round of usability tests was to see whether usability and user experience had improved after the change to the new React frontend. The quantitative measurements captured in this second usability study would be compared against the measurements of the first study, as reported in chapters 4 (page 34) and 5 (page 40).

Secondary goal of the second usability study round was to collect further qualitative findings which could be used as basis of further improvements later on, and to see whether the specific fixes described in chapter 6.2 (starting from page 53) addressed the issues they were supposed to.

### 7.2 Participants

A total of 13 participants were recruited for the second round of usability evaluations, including the pilot session. The idea was to use as many people as possible from the previous round to ensure that all participants would have at least a certain level of familiarity with the previous Execution Framework frontend. 11 of the 13 participants of the previous round were able to attend the second usability study. In addition, two members of the automated testing team that had not taken part in the previous round were recruited for this round. These two extra participants were deemed too familiar with the Ex-

ecution Framework system for the first round, but their previous experience with the system made them ideal candidates for the second round.

## 7.3 Procedure

The procedure used was for the most part identical to the one used in the first round of usability tests, with the exception of addition of post-test questions, described in section 7.6 on page 68. The earlier procedure is described in chapter 4.3 (page 35).

The time between the two rounds was about two months.

A pilot session was held a day before the rest of the sessions, but no adjustments to the scenarios or procedure were made after the session. Remaining sessions were held over 9 working days in June 2016, with up to two sessions per day. An hour was reserved for each session, plus fifteen minutes of setup time before each session. The sessions generally lasted between 15 and 30 minutes, with several sessions going up to 40 minutes.

## 7.4 Scenarios

The initial plan was to use the same scenarios from the first round as described in chapter 4.4 on page 36. However, as only a subset of the original feature set was implemented for the React frontend, likewise also a subset of the original scenarios were used.

A total of two scenarios were devised for the second round. Eventually only the fourth scenario of the first round, "Investigating a Failing Workflow", was re-used, with some of the minor details such as test names changed. A new initial scenario was also constructed. Both of these scenarios are presented and explained below.

### 7.4.1 Scenario 1: Current State of the System

To start, can you explain to me what do you think is the current state of the workflows running in the system?

This short initial scenario was used to guide the participant to navigate to the Workflow Manager page and to see whether the Workflow Manager-related fixes for displaying progress and state as described in chapters 6.2.2 (page 55) and 6.2.5 (page 59) were enough to enable the participant to quickly gather that information from the page now.

### 7.4.2 Scenario 2: Investigating a Failing Workflow

A previously configured workflow template for Component X has started to fail lately. You've been asked to investigate the issue, and then to attempt if running the failing tests again would solve the failures.

This scenario is essentially the same that was in the previous round, as described in chapter 4.4.4 on page 37.

This second and final scenario was used to gauge how improvements made to the Workflow Instance page affected usability and user experience, particularly whether participants would notice that the page had now been changed to automatically refresh itself, as described in chapter 6.2.3 on page 57.

## 7.5 Pre-Test and Post-Test Questionnaires

Before each testing session participants were asked to fill a short pre-study questionnaire. After finishing the scenario tasks participants were asked to fill AttrakDiff and SUS questionnaires. All three questionnaires were the same ones used in the previous round, as described in chapter 4.6 on page 39. The participants were this time specifically asked to only grade the workflow-related pages they had just used.

## 7.6 Post-Test Questions

At the end of each usability session three post-test questions were presented to participants right after they had filled the post-test questionnaires:

1. Do you feel that the design of the system has improved or worsened since the previous version?
2. Do you feel there's anything in the current design that could cause problems for use?
3. Do you feel there's anything else in the system that should be changed? What sort of changes?

The motivation behind these questions was to gauge whether participants felt that the design of the frontend was going into the right or wrong direction.

## Chapter 8

# Results of Second Evaluation

This chapter discusses findings collected from the second round of usability evaluation studies conducted on the new React frontend for the Execution Framework.

### 8.1 Overview

A total of 13 evaluation sessions were held, including the pilot session. One extra session had been booked, but was eventually not held due to scheduling conflicts.

### 8.2 Quantitative Findings

Numerical quantitative findings were collected using SUS and AttrakDiff in questionnaire format as with the first round of usability studies and as clarified in section 4.6 on page 39. The findings were scored similarly to earlier questionnaires in chapter 5.2 on page 40. Calculated mean averages for AttrakDiff and SUS, along with standard deviations and confidence intervals calculated at the 95% confidence level, can be found from table 8.1.

The mean scores point out that the frontend was considered slightly above "neutral" in all areas, with the aspect of attractiveness being the only AttrakDiff dimension scoring above 1.0. Further analysis is presented in the next chapter, as these findings are compared against quantitative findings collected from the first round of usability studies. Chapter 9 starts on page 75.

Table 8.1: Mean AttrakDiff and SUS results from the second usability study. (AttrakDiff: Min. = -3/Max. = 3; SUS: Min. = 0/Max. = 100; N = 13; CI = 95%. SD: standard deviation, CI: confidence interval, Lower: lower confidence bound, Upper: upper confidence bound.)

| Scale                                     | Mean  | SD    | CI   | Lower | Upper |
|---|-------|-------|------|-------|-------|
| AttrakDiff: Pragmatic quality             | 0.95  | 0.97  | 0.58 | 0.36  | 1.53  |
| AttrakDiff: Hedonic quality - identity    | 0.78  | 0.61  | 0.37 | 0.41  | 1.15  |
| AttrakDiff: Hedonic quality - stimulation | 0.76  | 0.71  | 0.43 | 0.33  | 1.19  |
| AttrakDiff: Attractiveness                | 1.53  | 0.80  | 0.48 | 1.04  | 2.01  |
| SUS                                       | 77.50 | 14.25 | 8.61 | 68.89 | 86.11 |

### 8.3 Qualitative Findings

Qualitative findings were collected and ranked identically to the first round, as explained in chapter 5.3 on page 41.

A total of 50 qualitative findings were collected from the second round. They ranked into severity levels as follows:

- 5 level-0 non-usability issues.
- 9 level-1 cosmetic issues.
- 27 level-2 minor usability issues.
- 9 level-3 major usability issues.
- 0 level-4 catastrophic usability issues.

As with the first round, the number of findings for each page was also collected, as seen in table 8.2. Findings that do not really belong to any specific page are also collected into the table.

Table 8.2: Usability finding counts per site page for the second usability study. Table is sorted by amount of high-level findings.

| Page              | Level-4 | Level-3 | Level-2 | Level-1 | Level-0 | Total |
|-------------------|---------|---------|---------|---------|---------|-------|
| Workflow Instance | 0       | 6       | 12      | 5       | 2       | 25    |
| Workflow Manager  | 0       | 2       | 9       | 2       | 1       | 14    |
| (no page)         | 0       | 1       | 6       | 1       | 2       | 10    |
| Home Page         | 0       | 0       | 0       | 1       | 0       | 1     |

As with the previous round, an overview of the findings per page is collected below.

### 8.3.1 Workflow Manager

Four out of the thirteen participants still voiced confusion regarding used terminology. The concepts of workflows, workflow templates and workflow instances weren't entirely clear for them, although as with last time, the participants were able to deduce what workflows were used for in the system by browsing the site for a bit. This issue of unclear terminology was compounded for a few of the participants by usage of the term "template" as a title for a field used to display the name of a workflow template. This change was a part of the fix for the first round's finding #66 (see chapter 6.2.4 on page 58). Wording the title as "template" turned out to just confuse participants as it was the only mention of that term on the Workflow Manager page. The previous version also had a button on the page for creating new workflow templates titled "Create New Workflow Template" but this entire button had been removed in the second version as the page had not yet been implemented after the switch to React frontend library.

In other words, the issues with external conceptual inconsistencies remained, although some of the participants had learned the terminology by the second study due to their daily use of the software and thus did not have these problems. The result is not surprising since these issues of external conceptual inconsistency were not addressed during the interface improvements.

As with findings for the Workflow Manager page in the previous study (chapter 5.3.3 on page 45), two participants voiced concerns on how they would find their workflows from the page as options for filtering, sorting or grouping workflows had not been implemented.

The fix for findings #57 (see chapter 6.2.2 on page 55) and #89 (chapter 6.2.5 on page 59) involved swapping the slightly confusing progress bar, used for displaying progress of test runs in a workflow instance, to table columns displaying numbers of passed, failed and in-progress tests. This change was well-liked, with 6 of the 13 participants saying out loud that being able to now see the current state of passed and failed tests in a running workflow instance was a good improvement. Three of the participants noted that the displayed numbers of passed, failed and in-progress tests did not add up to the total amount of tests in an instance and were confused by this. This was because the number of cancelled tests in an instance was not displayed anywhere on the Workflow Manager page. Although this was an intentional design decision to save space, it created internal physical inconsistency which

should be reconsidered in light of this feedback.

The implementation of the fix for finding #57 caused another finding in internal physical/communicational consistency in the interface among several participants: the improvement required the system to collect test result data in backend in order to serve accurate numbers of passed and failed tests in workflow instances to users. Gathering of this data turned out to be a relatively slow operation, so these fields were left empty during initial page load and filled in progressively by using AJAX calls to retrieve accurate data from backend. This caused two of the participants to not initially notice that there was any progress state available, or that the page would update itself with this state data after a few moments.

### 8.3.2 Workflow Instance

The Workflow Instance page turned out to have the most findings in the React frontend. A common issue among participants was the inability to cleanly cancel only one test run in a workflow instance. The only workaround was to cancel the entire instance, then re-run all test runs in it except the ones user would like to cancel. This was understandably deemed a clumsy solution.

Another issue was the lack of transparency in certain aspects of the system. The actual state of the worker cloud was not available, which frustrated three of the participants. One participant, who had already been using the previous ASP.NET frontend almost daily since its release, described the lack of a page showing states of each individual machine in the worker cloud as making maintenance of the Execution Framework cumbersome. Another participant was unsure what it meant when a test's state was set to "waiting" after they had attempted to re-run tests in a workflow instance, as they had no idea what the tests were exactly waiting for and the system did not provide them with answers.

Four of the participants would have preferred to see tests in a workflow instance presented as a tree-like structure, sorted by their test paths, instead of being displayed as the flat list they were in now. Implementing this would certainly make the visualization of these tests more internally physically consistent with the Test Browser page of the previous frontend version which displayed tests in a hierarchical tree.

An interesting note is that only several of the participants noticed that finding #1 (see chapter 6.2.3 on page 57) had been fixed and the page now auto-refreshed itself with latest test state progress. Most participants stayed on the page and waited it to refresh itself without refreshing the page manually. When asked about this, several of these participants admitted not

remembering that this had been an issue the last time.

Ironically, while fixing finding #87 (chapter 6.2.8 on page 62) involved removing a distracting spinner from Workflow Manager, the fix for finding #1 added a new spinner on the Workflow Instance page whenever it was fetching new data from backend. This distracted some participants as the connection between the page refreshing itself and the spinner was not very obvious, especially if there was nothing to update and thus nothing changed on the page as the spinner disappeared.

Participants did generally not notice the added summaries to workflow instance pages introduced in the fix to finding #48 (see chapter 6.2.1 on page 53) or the default sort arrows and columns for the fix to finding #50 (see chapter 6.2.7 on page 61). At least one participant noted the new summaries, and several participants now used the table sorting options although the utilization of these features was not as wide as hoped.

### 8.3.3 Other Findings

The only finding on the home page was a recurring finding from the previous study (see chapter 5.3.1 on page 43): that the page offered three different types of links to each of the main components of the site.

As also mentioned in chapter 5.3.8 (see page 49) participants still had trouble with navigation and were sometimes left in dead-end pages where they did not know how to navigate away from and were reluctant to use their browser's back button - namely when wanting to return to Workflow Manager from a Workflow Instance page.

In addition there were two noteworthy findings related to duration and finish times in workflows that appeared in both Workflow Manager and the Workflow Instance page:

Re-running tests in a workflow instance caused confusion in that an instance that had already finished once was given a finish time, but this was not removed when tests in it were being run again - it was merely updated once the tests being re-run had finished. Some participants found this unclear, as a workflow instance having a finish time implied to them that the instance had stopped all activity. Yet, tests were being ran in them.

Durations for workflow instances were also calculated in a way that caused concern in participants. Instead of tracking actual total run times of tests in a workflow instance, the total duration of a workflow instance was calculated as a subtraction of the finish time from the start time. The result was a quickly calculated rough estimate of how long an instance had been running. This became very difficult to interpret for participants, however, if any tests had been re-run in a workflow instance as that updated the instance's finish

time, making the duration sometimes vastly longer than it should have been, also rendering the number quite meaningless.

## 8.4 Other Feedback

Overall the feedback received during think-aloud protocol and post-test questions was that the participants preferred the new version and they thought it was going into a better direction, despite the lower scores from the questionnaires.

Many stated that the design was more clear, although it should be mentioned that the test data on the website was less cluttered during the second usability study, giving the appearance of a cleaner user interface.

## Chapter 9

# Analysis of Results

Quantitative results of both usability studies are compared side-by-side in table 9.1. Results for SUS and all four AttrakDiff qualities are presented for both ASP.NET and React versions of the frontend, matching the first and the second usability study. The measured change between the different frontend versions is also calculated. Note that all the numbers in the tables in this section have been rounded to nearest two decimal places.

As can be read from the table, the change in the mean averages of almost all of the used scales to measure user experience and usability has been negative, apart from the 0.05 increase in AttrakDiff's hedonic quality - stimulation. This means that, according to quantitative results of this entire study, the user experience and usability were measured to be slightly worse in the second usability study where the React frontend for Execution Framework was used. This outcome conflicts with the result from qualitative post-test question findings of the second usability study which claims that many of the participants found the React frontend at least slightly better than the previous version and that the development of the frontend was progressing towards a direction of their liking (see section 8.4 on page 74).

This conflict is not significantly changed if mean averages are compared only from the 11 participants that were present in both usability studies as can be seen in table 9.2. Again, usability and user experience have suffered minor net loss during the improvements made for the React frontend apart from AttrakDiff's scale of hedonic quality - stimulation.

Full per-adjective pair results for AttrakDiff for both of the usability studies are available in figure A.5 in the appendix, page 96. Interesting points in the graph are the two only major differences in the adjective pairs between the two frontend versions: the pairs for stylish-tacky and confusing-clearly structured. The difference of about 0.69 points in favour of "stylish" in the ASP.NET frontend is interesting, since the actual visual layout of the fron-

Table 9.1: Mean averages of AttrakDiff and SUS results from usability studies conducted for the ASP.NET frontend and the React frontend. (AttrakDiff: Min. = -3/Max. = 3; SUS: Min. = 0/Max. = 100; N = 13 (both studies/frontends); CI = 95%. SD: standard deviation, CI: confidence interval, Lower: lower confidence bound, Upper: upper confidence bound.)

| Quality                | System  | Mean  | SD    | CI    | Lower | Upper |
|------------------------|---------|-------|-------|-------|-------|-------|
| AD: Pragmatic          | ASP.NET | 1.04  | 0.91  | 0.55  | 0.49  | 1.60  |
|                        | React   | 0.95  | 0.97  | 0.58  | 0.36  | 1.53  |
|                        | Change  | -0.10 | 0.05  | 0.03  | -0.13 | -0.07 |
| AD: Hedonic - identity | ASP.NET | 1.01  | 0.33  | 0.20  | 0.81  | 1.21  |
|                        | React   | 0.78  | 0.61  | 0.37  | 0.41  | 1.15  |
|                        | Change  | -0.23 | 0.27  | 0.16  | -0.40 | -0.07 |
| AD: Hedonic - stimul.  | ASP.NET | 0.71  | 0.90  | 0.54  | 0.17  | 1.26  |
|                        | React   | 0.76  | 0.71  | 0.43  | 0.33  | 1.19  |
|                        | Change  | 0.04  | -0.19 | -0.12 | 0.16  | -0.07 |
| AD: Attractiveness     | ASP.NET | 1.58  | 0.61  | 0.37  | 1.21  | 1.95  |
|                        | React   | 1.52  | 0.80  | 0.48  | 1.04  | 2.01  |
|                        | Change  | -0.05 | 0.19  | 0.11  | -0.17 | 0.06  |
| SUS                    | ASP.NET | 79.62 | 12.54 | 7.57  | 72.04 | 87.19 |
|                        | React   | 77.50 | 14.25 | 8.61  | 68.89 | 86.11 |
|                        | Change  | -2.12 | 1.72  | 1.04  | -3.15 | -1.08 |

tend was not adjusted significantly during the improvements. Likewise the about one-point difference for "clearly structured" in the ASP.NET frontend is interesting as participants seemed to have much more trouble with navigation and searching for user interface elements among cluttered GUI items in the ASP.NET frontend.

Ultimately, however, the net losses in usability and user experience as measured with SUS and AttrakDiff – even the two AttrakDiff adjective pairs mentioned above – fall within the confidence interval of both studies, as can be seen in figure 9.1 for AttrakDiff and figure 9.2 for SUS (see page 78). This renders the result as statistically insignificant at the 95% confidence level. The best that can be said from these quantitative results is that if there was any change in usability and user experience, it was too minute to be noticed by these measurements, assuming neither of the measurements were skewed.

Table 9.2: Mean averages of AttrakDiff and SUS results from usability studies conducted for the ASP.NET frontend and the React frontend using only the 11 participants that were present in both studies. (AttrakDiff: Min. = -3/Max. = 3; SUS: Min. = 0/Max. = 100; N = 11; CI = 95%.)

| Quality                | System  | Mean  | SD    | CI    | Lower | Upper |
|------------------------|---------|-------|-------|-------|-------|-------|
| AD: Pragmatic          | ASP.NET | 1.01  | 0.99  | 0.67  | 0.35  | 1.68  |
|                        | React   | 0.95  | 1.02  | 0.68  | 0.26  | 1.63  |
|                        | Change  | -0.06 | 0.03  | 0.01  | -0.09 | -0.05 |
| AD: Hedonic - identity | ASP.NET | 1.03  | 0.36  | 0.24  | 0.78  | 1.27  |
|                        | React   | 0.75  | 0.66  | 0.44  | 0.31  | 1.20  |
|                        | Change  | -0.28 | 0.30  | 0.20  | -0.47 | -0.07 |
| AD: Hedonic - stimul.  | ASP.NET | 0.65  | 0.97  | 0.65  | 0.0   | 1.28  |
|                        | React   | 0.73  | 0.75  | 0.51  | 0.22  | 1.23  |
|                        | Change  | 0.08  | -0.22 | -0.14 | 0.22  | -0.05 |
| AD: Attractiveness     | ASP.NET | 1.68  | 0.61  | 0.41  | 1.26  | 2.09  |
|                        | React   | 1.52  | 0.87  | 0.58  | 0.94  | 2.10  |
|                        | Change  | -0.16 | 0.26  | 0.17  | -0.32 | 0.01  |
| SUS                    | ASP.NET | 80.68 | 12.85 | 8.63  | 72.05 | 89.31 |
|                        | React   | 76.14 | 15.02 | 10.09 | 66.05 | 86.22 |
|                        | Change  | -4.55 | 2.17  | 1.44  | -6.00 | -3.09 |

Overall, this leaves the results of the evaluation studies inconclusive, if cautiously optimistic. The qualitative results and overall participant feedback show promise, but they are not backed by the conflicting yet statistically insignificant quantitative results.

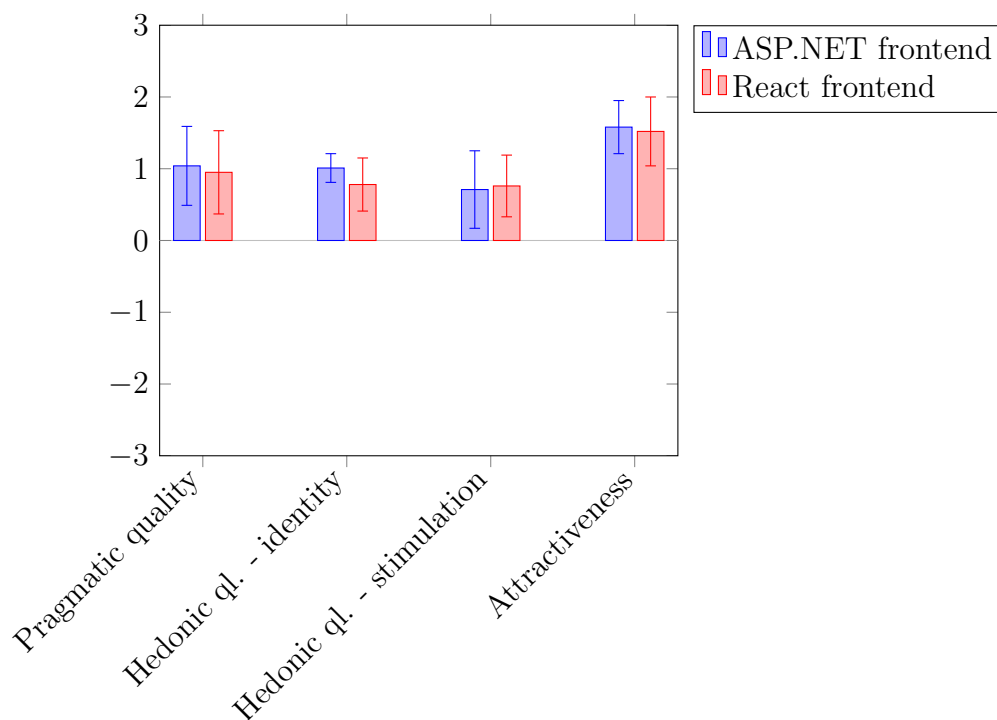


Figure 9.1: Mean AttrakDiff results from ASP.NET and React frontends. Error bars represent 95% confidence interval.

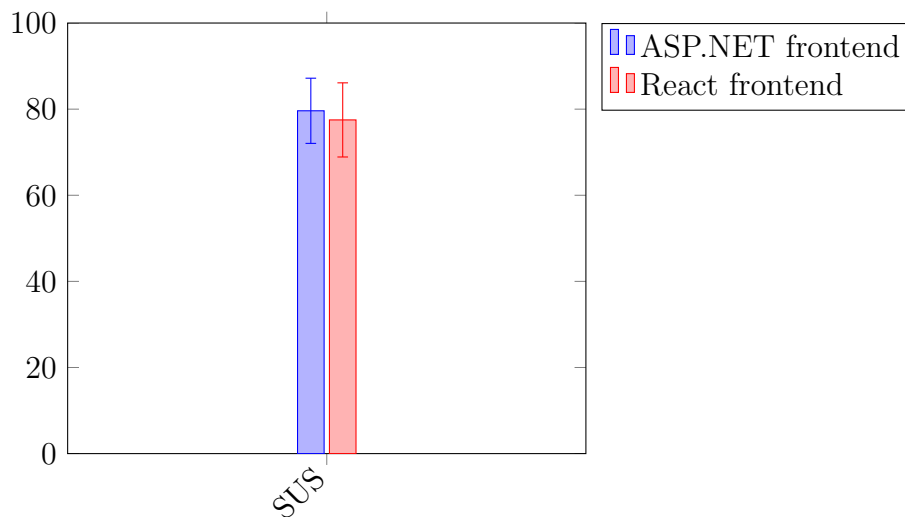


Figure 9.2: Mean SUS results from ASP.NET and React frontends. Error bars represent 95% confidence interval.

## Chapter 10

# Discussion

The first section of this chapter focuses on discussing the results gained from the study. The second section reflects on the reliability and validity of the conducted study.

### 10.1 Reflection on Results

As summed in chapter 9 the results of the study were mainly inconclusive, with the positive findings from qualitative data conflicting with the decrease in questionnaire scores in the quantitative data. If we assume that the measuring tools used had been perfectly calibrated, the statistical insignificance of the quantitative data would still mean that at best the improvement in user experience and usability would have been minor to non-existent, if not negative. This would still be an underwhelming result for the research question of whether focus on consistency would or would not affect user experience and usability.

What could have caused this discrepancy between quantitative and qualitative results? Or what could have caused the quantitative results not to improve during the second usability study? These are several potential ex-

planations:

1. Contrast with the old AUIT system.
2. Execution Framework taken into everyday use in-between usability study evaluation rounds.
3. Only subtle improvements.
4. Unimplemented features.
5. Use of within-subjects design in the usability study.
6. Qualitative results too positive.

The most likely reason seems that the participants had unconsciously contrasted the old Execution Framework frontend with that of the even older AUIT system's frontend, as described in chapter 3.2 on page 31. Almost all of the participants had specified their experience with the old AUIT system as either "some" or "quite a lot" in their pre-test questionnaires (see table 10.1 on page 81). It is possible that participants thus compared the Execution Framework's first frontend version they were testing to the old AUIT system they were familiar with. The contrast between these two systems is enormous, and may have coloured the quantitative results as "too good" in the first place. It is also possible that likewise when testing the second frontend for Execution Framework the participants were in fact comparing it to the first version of the frontend, not the old AUIT system, which could have resulted in their experiences with the system being much more muted.

Related to this is that the Execution Framework was rolled out and taken into everyday use during the two months between the two usability studies. In fact, 8 of the 11 participants that were in both usability studies reported in the second usability test's pre-test questionnaire that their experience with the Execution Framework had improved by one or two levels, generally from "not at all/not sure" to either "very little" or "some". All in all, during the first usability study only four participants self-reported their Execution Framework experience level as "very little" (everyone else reported it as "not at all/not sure"); during the second study five participants reported their experiences at the level of "some" and two at "quite a lot" (the numbers were two for "not at all/not sure" and four for "very little"). See table 10.1. During the usability tests around half of the participants also stated that they had been using the Execution Framework and the first version of the frontend in their daily work in the two months between the studies, whereas some participants said they had not used it since the previous study. As the

Table 10.1: Summary of pre-test questionnaire results regarding previous testing and execution environment experience at Varian, from both usability evaluation rounds. N=13 on both rounds. See the actual questionnaire as appendix figure A.1 on page 92.

|                            | Previous experience with... (round 1 / round 2) |         |         |
|----------------------------|---|---------|---------|
|                            | ...automated testing                            | ...AUIT | ...FAST |
| 1 - not at all or not sure | 2 / 0   | 1 / 1   | 9 / 2   |
| 2 - very little            | 2 / 1   | 1 / 1   | 4 / 4   |
| 3 - some                   | 2 / 5   | 7 / 4   | 0 / 5   |
| 4 - quite a lot            | 7 / 7   | 4 / 7   | 0 / 2   |

fresh novelty of the Execution Framework has undoubtedly worn off during this time, it's also possible that the scores the participants would give the system in AttrakDiff and SUS questionnaires would get normalised down to less overly enthusiastic levels.

The largest changes the frontend saw were effectively major refactoring changes to the source code, while the user-facing changes were relatively subtle. As there were so few changes that were easily perceivable by participants, and as the improvements only covered about one third of the first version of the Execution Framework frontend with plenty of unimplemented sections, it's possible that participants graded the React fronted more poorly than they would have otherwise, despite that they were asked to only grade the workflow-related sections that had been implemented.

All of these issues were further compounded by the use of within-subjects design (each study participant tested each version of the system, apart from only a few exceptions) in the usability study, as participants gaining practice during the first round of a study can have a carry-over effect on the second one, affecting its result [7]. However, using between-subjects design (testing each version of the system with different participants) would not have been reasonable during this study due to specifics of this particular study: the site-wide adoption of the Execution Framework system between the study evaluation rounds and the limited size of the pool of potential study participants.

And, of course, it's possible that the qualitative results gathered during either of the usability rounds were too positive, as participants may be more eager to please the session moderator and talk up the software they've been testing [3, pp. 208-209]. Eliminating this sort of biased behaviour could push the results into either direction of statistical significance – the second version of the frontend could be either better or worse than the first one – but one would hope that the symmetrical nature of using colleagues as usability evaluation participants in both usability studies would eliminate this possibility.

Overall, with these inconclusive results it is difficult to say if the focus on consistency in frontend improvements brought anything for user experience and usability.

## 10.2 Reliability and validity

As the quantitative measurements done in this research were done by using AttrakDiff and SUS, so is the reliability of this research tied heavily to those two tools. As both AttrakDiff and SUS are considered to be reliable [2, 34], so is the measuring of the effect of consistency on user experience and usability done in this research.

Internal validity of the research, however, is not particularly strong. Qualitative results were collected, but quantitative results remain statistically insignificant. It's also difficult to say how much of an effect the modified frontend had on participants AttrakDiff and SUS rankings compared to influences caused by external factors such as participants' moods at the time.

The construct validity of the performed research is quite low, also: it is currently not possible to say with high confidence whether the changes in measured quantitative and qualitative user experience and usability were due to the changes done to consistency.

One root issue for this lack of construct validity was the use of within-subjects design, the use of a lot of the same participants in both of the evaluation studies. To better measure actual merits of focusing on consistency-related improvements three different groups of non-overlapping participants would be required, and each of the groups would have to evaluate a different variation of the frontend: one control frontend with no modifications; one frontend with consistency-related improvements only; and one frontend with only improvements that have no relation to consistency. This degree of research was not feasible in the scope of this study due to limited amount of participants and limited development time for the frontend improvements (as this proposed study would have required two different variations of the

new, improved frontend iteration).

Had the quantitative results been neutral instead of slightly negative it could at least have been said that consistency improvements had no effect on user experience and usability. However, not only would this result be inherently statistically insignificant, it would also still conflict with the positive results from qualitative data.

Another way to further the construct validity in any similar research would be to employ more triangulation, such as measuring the change in task performance times between the studies. As is, this particular example would not have added much benefit to this study. First, measuring task performance would have focused on pragmatic aspects of user experience, so the result would have been interesting only from the point of view of usability, sidelining hedonic aspects of user experience. Furthermore, as per the original problems with the quantitative data, one potential reason for the unexpected results was many of the participants starting to use the system in the two months between the studies. Participants getting accustomed to the system would have most likely reflected similarly in task performance times.

## Chapter 11

# Conclusion

This thesis studied whether gains in measured user experience (and usability) could be achieved by focusing on consistency-related issues when improving a system. This was done by conducting two studies of task-based usability tests where quantitative user experience and usability differences were measured with AttrakDiff and SUS post-evaluation questionnaires. The tested interface in question, a web frontend for an in-house automated test execution system, was refactored into ReactJS platform and received consistency-related improvements between the studies.

The research question presented in the introductory chapter on page 14 asked whether a consistent user interface is better from user experience point of view than a heterogeneous one. A clear answer was not reached for this primary research question, however. The qualitative findings from the user experience evaluation study pointed towards a confirming position, but the quantitative findings from the same study conflicted with this, showing a slight, albeit ultimately statistically insignificant, loss in measured user experience and usability after the consistency-related changes to the frontend. Chapter 10 included some speculation on why this result was reached, including the possibility of the evaluation participants grading the frontend "too well" on the first evaluation round, resulting in a more evened-out and ultimately worse questionnaire results in the second round.

First of the two sub-questions asked what a consistent user experience means. This was answered in section 2.1.4 on page 20 by defining consistency of user experience via dimensions of internal and external consistency, across conceptual, communicational and physical components.

The second sub-question asked how consistency of user experience could be measured. This was answered in section 2.3 on page 23 by suggesting measuring it as a change between two quantitative user experience measurements done on a system or a product that has had only consistency-related

improvements done to it between the two measurements.

While this thesis was not able to provide a satisfying answer to the main research question – whether a consistent interface improves user experience or not – it does hopefully give groundwork into formalizing the use of consistency as a concept in context of user experience. The suggested methodology for measuring consistency in user experience is also potentially useful.

The most significant issue with the research done on in this thesis was its low validity. This could be improved in any future research by using between-subjects design instead of within-subjects – that is, using different participants for each iteration of the system. Effects of practice on results should also be eliminated or lessened by only using participants that have never used the system before, or participants that all have similar previous experience with the system. Another variation would be to use three different groups of participants as described on page 82 - one to test consistency-related improvements, one to test improvements not related to consistency, and one control group testing the original, unmodified version. Further triangulation could also be used, as described on page 83.

Future research into developing a method for measuring consistency in user experience that is not based on measuring differences between iterations would also certainly be a significant improvement and would potentially enable a tool that is much more straightforward to use as measurements could be taken "in place" instead of relying on relative comparisons of different versions of the system or product.

A tangential angle that came up during this research was the benefit of improving the hedonic aspects of user experience in an in-house software. There does not really seem to be research in this subject matter, with or without the consistency angle. Existing research seems to be focused on studying user experience in consumer products. It's uncertain how applicable this consumer-centered approach is to in-house systems and products that generally do not involve a "choice" for the user to decide whether to use it or not. User experience research on consumer products is often based on how these products can be made attractive and usable enough for users to either begin or keep using them, whereas in-house products are often characterised by their non-voluntary use, and inferior user satisfaction does not necessarily correlate as directly to "lost sales" as with consumer products.

So is there any advantage at all in enhancing user experience in in-house software? Norman [24, pp. 19-20] states that attractive things, as subjective as they are, make people feel good, which in turn makes them think more creatively, which, in turn, makes them more effective at finding alternative solutions and can make them more tolerant of minor difficulties. This could offer an interesting avenue into studying how hedonic aspects of user research

might increase effectiveness and error tolerance in in-house systems. In their 2015 paper Hassenzahl, Wiklund-Engblom, Bengs, Hägglund & Diefenbach [12], while not discussing about in-house products and systems, do recommend taking hedonic aspects into account during product evaluation, as they found out that hedonic aspects had an impact on experienced need fulfilment, whereas pragmatic aspects had none. A study by Diefenbach & Hassenzahl [5] discovered a desire for hedonic aspects even in tools that were considered as purely utilitarian.

Another potential avenue of approach is the whole question of consistency as such a core component of usability. Considering the emphasis consistency has in such fundamental human-computer interaction literature as Nielsen's usability heuristics it was reasonable to expect that the effect of consistency on user experience would have had a more obviously prominent effect to this study. But certainly the research conducted in this thesis is not nearly enough to infer consistency as an irrelevant aspect of usability.

On the whole, consistency is an intuitive but a vaguely defined concept in human-computer interaction which has not seen much attention as the focus within the field of HCI has moved towards user experience. Consistency can be measured through the effects it has on UX, but the levels of its practical importance for user experience remain elusive.

# Bibliography

- [1] ALTABOLI, A., AND ABOU-ZEID, M. R. Effect of Physical Consistency of Web Interface Design on Users' Performance and Satisfaction. *Proceedings of the 12th International Conference on Human-computer Interaction: Applications and Services* (2007), 849–858.
- [2] BANGOR, A., KORTUM, P. T., AND MILLER, J. T. An Empirical Evaluation of the System Usability Scale. *International Journal of Human-Computer Interaction* 24, March 2015 (2008), 574–594.
- [3] BARNUM, C. M. *Usability Testing Essentials*. Elsevier, Burlington, 2011.
- [4] BROOKE, J. SUS - A quick and dirty usability scale. *Usability evaluation in industry* 189, 194 (1996), 4–7.
- [5] DIEFENBACH, S., AND HASSENZAHL, M. The dilemma of the hedonic - Appreciated, but hard to justify. *Interacting with Computers* 23, 5 (2011), 461–472.
- [6] Ergonomics of human-system interaction - Part 210: Human-centred design for interactive systems (ISO 9241-210:2010). Standard, European Committee for Standardization, Brussels, BE, Oct. 2010.
- [7] GREENWALD, A. G. Within-Subjects Designs: To Use or Not To Use? *Psychological Bulletin* 83, 2 (1976), 314–320.
- [8] GRUDIN, J. The case against user interface consistency. *Communications of the ACM* 32, 10 (1989), 1164–1173.
- [9] GRUDIN, J. Consistency, standards, and formal approaches to interface development and evaluation: a note on Wiecha, Bennett, Boies, Gould, and Greene. *ACM Transactions on Information Systems* 10, 1 (1992), 103–111.

- [10] HASSENZAHL, M. *The Thing and I: Understanding the Relationship Between User and Product*. Springer Netherlands, Dordrecht, 2003, ch. 3, pp. 31–42.
- [11] HASSENZAHL, M. The hedonic/pragmatic model of user experience. In *Towards a UX Manifesto*, E. Law, A. Vermeeren, M. Hassenzahl, and M. Blythe, Eds. COST294-MAUSE affiliated workshop, 2007, pp. 10–14.
- [12] HASSENZAHL, M., WIKLUND-ENGBLOM, A., BENGS, A., HÄGGLUND, S., AND DIEFENBACH, S. Experience-oriented and product-oriented evaluation: psychological need fulfillment, positive affect, and product perception. *International Journal of Human-Computer Interaction* 31, 8 (2015), 530–544.
- [13] KELLOGG, W. A. Conceptual consistency in the user interface: Effects on user performance. *Proceedings of INTERACT'87 Conference on Human-Computer Interaction* (1987), 389–394.
- [14] KELLOGG, W. A. The Dimensions of Consistency. In *Coordinating User Interfaces for Consistency*, J. Nielsen, Ed. Morgan Kaufmann Publishers, San Francisco, 2002, pp. 9–20. Original work published 1989.
- [15] KORITZINSKY, I. H. New Ways to Consistent Interfaces. In *Coordinating User Interfaces for Consistency*, J. Nielsen, Ed. Morgan Kaufmann Publishers, San Francisco, 2002, pp. 93–106. Original work published 1989.
- [16] KÜHNEL, C. *Quantifying Quality Aspects of Multimodal Interactive Systems*. T-Labs Series in Telecommunication Services. Springer, Berlin, 2012.
- [17] KÜHNEL, C., WESTERMANN, T., WEISS, B., AND MÖLLER, S. Evaluating multimodal systems: a comparison of established questionnaires and interaction parameters. In *NordiCHI '10 Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries* (New York, 2010), ACM, pp. 286–294.
- [18] MENDEL, J., PAK, R., AND DRUM, J. E. Designing for Consistency: Can Interface Consistency Reduce Workload in Dual-task Situations? *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 55, 1 (2011), 2000–2004.
- [19] NIELSEN, J. Severity Ratings for Usability Problems. <https://www.nngroup.com/articles/>

- [how-to-rate-the-severity-of-usability-problems/](#) (visited on 8.8.2016).
- [20] NIELSEN, J. Coordinating user interfaces for consistency. *ACM SIGCHI Bulletin* 20, 3 (1989), 63–65.
- [21] NIELSEN, J. Heuristic Evaluation. In *Usability Inspection Methods*, J. Nielsen and R. L. Mack, Eds. John Wiley & Sons, New York, 1994, ch. 2, pp. 25–62.
- [22] NIELSEN, J. Executive Summary: Coordinating User Interfaces for Consistency. In *Coordinating User Interfaces for Consistency*, J. Nielsen, Ed. Morgan Kaufmann Publishers, San Francisco, 2002, pp. 1–7. Original work published 1989.
- [23] NIELSEN, J. Preface. In *Coordinating User Interfaces for Consistency*, J. Nielsen, Ed. Morgan Kaufmann Publishers, San Francisco, 2002, pp. vii–viii.
- [24] NORMAN, D. A. *Emotional Design: Why We Love (or Hate) Everyday Things*. Basic Books, New York, 2004.
- [25] OLMSTED-HAWALA, E. L., MURPHY, E. D., HAWALA, S., AND ASHENFELTER, K. T. Think-Aloud Protocols: A Comparison of Three Think-Aloud Protocols for use in Testing Data-Dissemination Web Sites for Usability. *Chi2010: Proceedings of the 28Th Annual Chi Conference on Human Factors in Computing Systems, Vols 1-4* (2010), 2381–2390.
- [26] OZOK, A. A., AND SALVENDY, G. Measuring consistency of web page design and its effects on performance and satisfaction. *Ergonomics* 43, 4 (2000), 443–460.
- [27] PEEDU, G., AND LAMAS, D. Minu Viljandi: A Case Study on the Effects of Introducing Web 2.0 Features in e-Government Services on the Overall User Experience Perception. 305–308.
- [28] SATZINGER, J. W., AND OLFMAN, L. User Interface Consistency across End-User Applications: The Effects on Mental Models. *Journal of Management Information Systems* 14, 4 (1998), 167 – 193.
- [29] SHARP, H., ROGERS, Y., AND PREECE, J. *Interaction Design: Beyond Human-Computer Interaction*, 2nd ed. John Wiley & Sons Ltd, West Sussex, 2007.

- [30] TOGNAZZINI, B. Achieving Consistency for the Macintosh. In *Coordinating User Interfaces for Consistency*, J. Nielsen, Ed. Morgan Kaufmann Publishers, San Francisco, 2002, pp. 57–73. Original work published 1989.
- [31] TULLIS, T., AND ALBERT, B. Measuring the User Experience. <http://www.measuringux.com/> (visited on 8.8.2016).
- [32] VARSALUOMA, J., AND SAHAR, F. Measuring Retrospective User Experience of Non-Powered Hand Tools: An Exploratory Remote Study with UX Curve. *Proceedings of the 18th International Academic MindTrek Conference: Media Business, Management, Content & Services* (2014), 40–47.
- [33] WALSH, T., VARSALUOMA, J., KUJALA, S., NURKKA, P., PETRIE, H., AND POWER, C. Axe UX : Exploring Long-Term User Experience with iScale and AttrakDiff. In *Proceedings of the 18th International Academic MindTrek Conference: Media Business, Management, Content & Services* (New York, 2014), ACM, pp. 32–39.
- [34] WETZLINGER, W., AUINGER, A., AND DÖRFLINGER, M. Comparing Effectiveness, Efficiency, Ease of Use, Usability and User Experience When Using Tablets and Laptops. *Third International Conference, DUXU 2014, Held as Part of the HCI International 2014*, March (2014), 704.
- [35] WOLF, R. Consistency as Process. In *Coordinating User Interfaces for Consistency*, J. Nielsen, Ed. Morgan Kaufmann Publishers, San Francisco, 1989, pp. 89–92. Original work published 1989.
- [36] ZIMMERMANN, P. G. *Beyond Usability - Measuring Aspects of User Experience*. PhD thesis, Swiss Federal Institute of Technology Zurich, 2008.

## Appendix A

# Questionnaires

This appendix contains all the questionnaires used during usability evaluation studies conducted in this thesis work. A semantic differential graph of the AttrakDiff questionnaire results is also included.

Name:

Team:

Date:

How much have you implemented, maintained or investigated automated unit or integration tests at Varian? Please circle your answer:

Not at all/not sure    Very little    Some    Quite a lot

How much have you previously used the current "AUIT" framework, or accessed test logs through *uitlogs.xml*, or used the AUIT frontend at *he-  
vuit-db*? Please circle your answer:

Not at all/not sure    Very little    Some    Quite a lot

How much have you used the new "FAST" Execution Framework, or accessed its frontend, to view, configure or schedule test runs? Please circle your answer:

Not at all/not sure    Very little    Some    Quite a lot

Figure A.1: Pre-test questionnaire for first round of usability evaluations.

Participant ID: \_\_\_\_\_ Site: \_\_\_\_\_ Date: \_\_\_/\_\_\_/\_\_\_

## System Usability Scale

**Instructions:** For each of the following statements, mark one box that best describes your reactions to the website *today*.

|     |  | Strongly<br>Disagree     |                          |                          |                          | Strongly<br>Agree        |
|-----|--|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|
| 1.  | I think that I would like to use this website frequently.                      | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| 2.  | I found this website unnecessarily complex.                                    | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| 3.  | I thought this website was easy to use.  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| 4.  | I think that I would need assistance to be able to use this website.           | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| 5.  | I found the various functions in this website were well integrated.            | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| 6.  | I thought there was too much inconsistency in this website.                    | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| 7.  | I would imagine that most people would learn to use this website very quickly. | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| 8.  | I found this website very cumbersome/awkward to use.                           | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| 9.  | I felt very confident using this website.                                      | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| 10. | I needed to learn a lot of things before I could get going with this website.  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

This questionnaire is based on the System Usability Scale (SUS), which was developed by John Brooke while working at Digital Equipment Corporation. © Digital Equipment Corporation, 1986.

Figure A.2: Systems Usability Test (SUS) questionnaire by Tullis and Albert [31], based on the work by Brooke [4].

AttrakDiff Evaluation Questionnaire

Participant name or ID: \_\_\_\_\_

Instructions: Please choose the most appropriate assessment between word pairs from each line. You may feel that some word pairs do not fit the product very well. However, we would ask you to give an answer anyway. Remember that there are no "right" or "wrong" answers – your personal opinion is what counts.

|              |                       |                       |                       |                       |                       |                       |                       |                 |
|--------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------|
| human        | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | technical       |
| isolating    | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | connective      |
| pleasant     | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | unpleasant      |
| inventive    | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | conventional    |
| simple       | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | complicated     |
| professional | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | unprofessional  |
| ugly         | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | attractive      |
| practical    | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | impractical     |
| likeable     | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | disagreeable    |
| cumbersome   | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | straightforward |
| stylish      | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | tacky           |
| predictable  | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | unpredictable   |
| cheap        | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | premium         |

*Please turn this page over and continue the questionnaire*

Figure A.3: AttrakDiff questionnaire, page 1.

AttrakDiff Evaluation Questionnaire

|                            |                       |                       |                       |                       |                       |                       |                       |                          |
|----------------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|--------------------------|
| alienating                 | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | integrating              |
| brings me closer to people | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | separates me from people |
| unpresentable              | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | presentable              |
| rejecting                  | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | inviting                 |
| unimaginative              | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | creative                 |
| good                       | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | bad                      |
| confusing                  | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | clearly structured       |
| repelling                  | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | appealing                |
| bold                       | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | cautious                 |
| innovative                 | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | conservative             |
| dull                       | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | captivating              |
| undemanding                | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | challenging              |
| motivating                 | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | discouraging             |
| novel                      | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | ordinary                 |
| unruly                     | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | manageable               |

Figure A.4: AttrakDiff questionnaire, page 2.

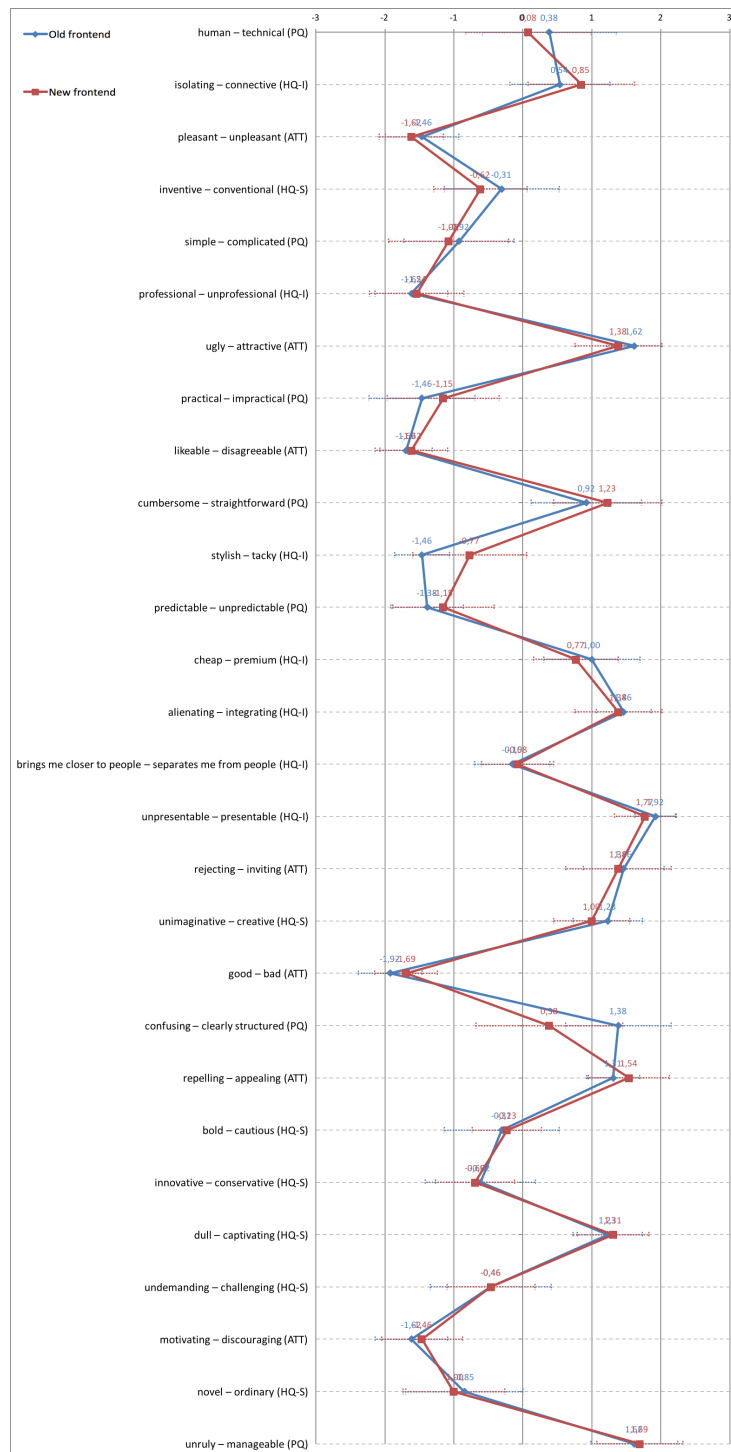


Figure A.5: Mean results from AttrakDiff questionnaires for each adjective pair, from both usability studies. (Both studies: N=13; CI=95%)

## Appendix B

# Final React Frontend Designs

This appendix contains comparison screenshots of the final React frontend page designs versus their earlier ASP.NET frontend counterparts.

### Workflow Manager

| Template name          | System under test | Priority | Description                              | Progress |        |                   | Schedule                       |
|------------------------|-------------------|----------|--|----------|--------|-------------------|--------------------------------|
| <b>Test SUT 2, all</b> | Test SUT 2        | 2        | Nightly runs for every Test SUT 2 build. |          |        |                   | Every day, for every new build |
| Name                   | Build             | State    | In-progress                              | Passed   | Failed | Start time        | Finish time                    |
| Test SUT 2, all        | 15.0.24.02        | Finished | 0/222                                    | 155/222  | 67/222 | 5 Aug 2016 11:50  | 5 Aug 2016 12:35               |
| Test SUT 2, all        | 15.0.23.03        | Finished | 0/222                                    | 155/222  | 67/222 | 10 Jun 2016 12:33 | 10 Jun 2016 17:18              |
| Test SUT 2, all        | 15.0.23.02        | Finished | 0/222                                    | 155/222  | 67/222 | 10 Jun 2016 12:32 | 10 Jun 2016 12:50              |
| Test SUT 2, all        | 15.0.23.01        | Finished | 0/222                                    | 155/222  | 67/222 | 10 Jun 2016 11:08 | 10 Jun 2016 11:25              |
| Test SUT 2, all        | 15.0.22.02        | Finished | 0/222                                    | 155/222  | 67/222 | 9 Jun 2016 23:04  | 9 Jun 2016 23:19               |
| Test SUT 2, all        | 15.0.21.08        | Finished | 0/222                                    | 155/222  | 67/222 | 9 Jun 2016 14:37  | 9 Jun 2016 15:35               |
| Test SUT 2, all        | 15.0.21.05        | Finished | 0/222                                    | 155/222  | 67/222 | 9 Jun 2016 13:43  | 9 Jun 2016 14:36               |
| Test SUT 2, all        | 15.0.21.01        | Finished | 0/222                                    | 155/222  | 67/222 | 9 Jun 2016 13:04  | 9 Jun 2016 13:31               |
| Test SUT 2, all        | 15.0.20.02        | Finished | 0/222                                    | 155/222  | 67/222 | 2 Jun 2016 12:38  | 8 Jun 2016 23:30               |

| Template name              | System under test           | Priority | Description | Progress |         |                   | Schedule                             |
|----------------------------|-----------------------------|----------|-------------|----------|---------|-------------------|--------------------------------------|
| <b>Test SUT 1 workflow</b> | Test SUT 1                  | 3        | Test runs.  |          |         |                   | Every day, for every new build       |
| Name                       | Build                       | State    | In-progress | Passed   | Failed  | Start time        | Finish time                          |
| Test SUT 1 workflow        | MockUnitTests_008           | Error    | 0/441       | 80/441   | 95/441  | 5 Aug 2016 11:50  | <span style="color:red">error</span> |
| Test SUT 1 workflow        | MockUnitTests_005           | Finished | 0/490       | 249/490  | 237/490 | 10 Jun 2016 15:33 | 5 Aug 2016 15:44                     |
| Test SUT 1 workflow        | MockUnitTests_004           | Finished | 0/18        | 12/18    | 6/18    | 10 Jun 2016 15:17 | 10 Jun 2016 15:34                    |
| Test SUT 1 workflow        | MockUnitTests_003           | Finished | 0/9         | 9/9      | 0/9     | 10 Jun 2016 15:08 | 10 Jun 2016 15:13                    |
| Test SUT 1 workflow        | mstestsharp_9_6_2016 - Copy | Finished | 0/4         | 0/4      | 4/4     | 10 Jun 2016 15:05 | 10 Jun 2016 15:05                    |
| Test SUT 1 workflow        | mstestsharp_9_6_2016        | Finished | 0/4         | 1/4      | 3/4     | 10 Jun 2016 12:49 | 10 Jun 2016 12:51                    |
| Test SUT 1 workflow        | mstestsharp_8_6_2016        | Finished | 0/4         | 1/4      | 3/4     | 9 Jun 2016 13:05  | 9 Jun 2016 13:06                     |
| Test SUT 1 workflow        | mstestsharp_2_6_2016        | Finished | 0/4         | 1/4      | 3/4     | 2 Jun 2016 12:36  | 8 Jun 2016 23:09                     |

Figure B.1: The new React frontend Workflow Manager page.

### Workflows

[+ Create a new Workflow](#)

| Test SUT 2 Primary Tests  | Test SUT 2 | Priority: 1 | Primary Tests for Test SUT 2 branch.                                   | Every day, for every new build |                   |                                      |
|---|------------|-------------|--|--------------------------------|-------------------|--------------------------------------|
| <input type="checkbox"/> Hide 1 running and recent instances of this workflow |            |             |  |                                |                   |                                      |
| Name  | Build      | State       | Progress   | Start time                     | Finish time       | Instanced by                         |
| Test SUT 2 Primary Tests  | 15.0.24.02 | Error       | <div style="width: 100%;"><div style="width: 100%;"></div></div> 72/72 | 5 Aug 2016 11:51               |                   | <span style="color:red">error</span> |
| Test SUT 2 Primary Tests  | 15.0.23.03 | Finished    | <div style="width: 100%;"><div style="width: 100%;"></div></div> 72/72 | 10 Jun 2016 12:33              | 10 Jun 2016 15:46 |                                      |
| Test SUT 2 Primary Tests  | 15.0.23.02 | Finished    | <div style="width: 100%;"><div style="width: 100%;"></div></div> 72/72 | 10 Jun 2016 12:32              | 10 Jun 2016 12:36 |                                      |
| Test SUT 2 Primary Tests  | 15.0.23.01 | Finished    | <div style="width: 100%;"><div style="width: 100%;"></div></div> 72/72 | 10 Jun 2016 11:08              | 10 Jun 2016 11:12 |                                      |
| Test SUT 2 Primary Tests  | 15.0.22.02 | Finished    | <div style="width: 100%;"><div style="width: 100%;"></div></div> 72/72 | 9 Jun 2016 23:04               | 10 Jun 2016 10:34 |                                      |
| Test SUT 2 Primary Tests  | 15.0.21.08 | Finished    | <div style="width: 100%;"><div style="width: 100%;"></div></div> 72/72 | 9 Jun 2016 14:37               | 9 Jun 2016 14:43  |                                      |
| Test SUT 2 Primary Tests  | 15.0.21.05 | Finished    | <div style="width: 100%;"><div style="width: 100%;"></div></div> 72/72 | 9 Jun 2016 13:43               | 9 Jun 2016 13:50  |                                      |
| Test SUT 2 Primary Tests  | 15.0.21.01 | Finished    | <div style="width: 100%;"><div style="width: 100%;"></div></div> 72/72 | 9 Jun 2016 13:04               | 9 Jun 2016 13:11  |                                      |
| Test SUT 2 Primary Tests  | 15.0.20.02 | Finished    | <div style="width: 100%;"><div style="width: 100%;"></div></div> 72/72 | 2 Jun 2016 12:38               | 8 Jun 2016 18:28  |                                      |
| <input type="checkbox"/> Hide finished instances                              |            |             |  |                                |                   |                                      |
| <a href="#">Edit this workflow</a>  |            |             |  |                                |                   |                                      |

| Test SUT 2, all   | Test SUT 2 | Priority: 2 | Nightly runs for every Test SUT 2 build. | Every day, for every new build |
|---|------------|-------------|--|--------------------------------|
| <input type="checkbox"/> Show 1 running and recent instances of this workflow |            |             |  |                                |
| <a href="#">Edit this workflow</a>  |            |             |  |                                |

Figure B.2: The old ASP.NET frontend Workflow Manager page.

Workflow instance: Test SUT 1 workflow, build MockUnitTests\_004

**Build:** MockUnitTests\_004    **System under test:** Test SUT 1    **Requested by:**

**Run state:** Finished    **In-progress tests:** 0/18    **Finished tests:** 18/18

**Passed tests:** 12/18    **Failed tests tests:** 6/18    **Canceled tests:** 0/18

**Start time:** 10 Jun 2016 15:17    **Finish time:** 10 Jun 2016 15:34    **Total run time:** 16 min, 59 sec

**Description:**

[Cancel this workflow instance](#)

Tests in this workflow instance

Select All    [Re-run selected tests](#)    Filter:

Currently you have 0 items selected.

| Test name   | State                 | Verdict timestamp | Worker   |
|---|-----------------------|-------------------|----------|
| <input type="checkbox"/> MockUnitTestForFastTests.dll/MockClassAAA/RandomlyFailingMockTest_30SecSleep_AAA_1 | ✔ Test result: Passed | 10 Jun 2016 15:33 | HE-UITB6 |
| <input type="checkbox"/> MockUnitTestForFastTests.dll/MockClassAAA/RandomlyFailingMockTest_30SecSleep_AAA_2 | ✔ Test result: Passed | 10 Jun 2016 15:22 | HE-UITB6 |
| <input type="checkbox"/> MockUnitTestForFastTests.dll/MockClassAAA/RandomlyFailingMockTest_30SecSleep_AAA_3 | ✔ Test result: Passed | 10 Jun 2016 15:24 | HE-UITB6 |
| <input type="checkbox"/> MockUnitTestForFastTests.dll/MockClassAAA/RandomlyFailingMockTest_30SecSleep_AAA_4 | ✔ Test result: Passed | 10 Jun 2016 15:34 | HE-UITB6 |
| <input type="checkbox"/> MockUnitTestForFastTests.dll/MockClassAAA/RandomlyFailingMockTest_30SecSleep_AAA_5 | ✔ Test result: Passed | 10 Jun 2016 15:24 | HE-UITB6 |
| <input type="checkbox"/> MockUnitTestForFastTests.dll/MockClassAAA/RandomlyFailingMockTest_30SecSleep_AAA_6 | ✔ Test result: Passed | 10 Jun 2016 15:26 | HE-UITB6 |
| <input type="checkbox"/> MockUnitTestForFastTests.dll/MockClassAAA/RandomlyFailingMockTest_30SecSleep_AAA_7 | ✔ Test result: Passed | 10 Jun 2016 15:22 | HE-UITB6 |
| <input type="checkbox"/> MockUnitTestForFastTests.dll/MockClassAAA/RandomlyFailingMockTest_30SecSleep_AAA_8 | ✔ Test result: Passed | 10 Jun 2016 15:26 | HE-UITB6 |
| <input type="checkbox"/> MockUnitTestForFastTests.dll/MockClassAAA/RandomlyFailingMockTest_30SecSleep_AAA_9 | ✔ Test result: Passed | 10 Jun 2016 15:25 | HE-UITB6 |
| <input type="checkbox"/> MockUnitTestForFastTests.dll/MockClassAAB/RandomlyFailingMockTest_30SecSleep_AAB_1 | ✘ Test result: Failed | 10 Jun 2016 15:20 | HE-UITB6 |
| <input type="checkbox"/> MockUnitTestForFastTests.dll/MockClassAAB/RandomlyFailingMockTest_30SecSleep_AAB_2 | ✔ Test result: Passed | 10 Jun 2016 15:17 | HE-UITB6 |
| <input type="checkbox"/> MockUnitTestForFastTests.dll/MockClassAAB/RandomlyFailingMockTest_30SecSleep_AAB_3 | ✔ Test result: Passed | 10 Jun 2016 15:21 | HE-UITB6 |
| <input type="checkbox"/> MockUnitTestForFastTests.dll/MockClassAAB/RandomlyFailingMockTest_30SecSleep_AAB_4 | ✘ Test result: Failed | 10 Jun 2016 15:18 | HE-UITB6 |
| <input type="checkbox"/> MockUnitTestForFastTests.dll/MockClassAAB/RandomlyFailingMockTest_30SecSleep_AAB_5 | ✘ Test result: Failed | 10 Jun 2016 15:23 | HE-UITB6 |
| <input type="checkbox"/> MockUnitTestForFastTests.dll/MockClassAAB/RandomlyFailingMockTest_30SecSleep_AAB_6 | ✔ Test result: Passed | 10 Jun 2016 15:25 | HE-UITB6 |
| <input type="checkbox"/> MockUnitTestForFastTests.dll/MockClassAAB/RandomlyFailingMockTest_30SecSleep_AAB_7 | ✘ Test result: Failed | 10 Jun 2016 15:21 | HE-UITB6 |

Figure B.3: The new React frontend Workflow Instance page.

Workflow instance: Test SUT 1 workflow

**Build:** MockUnitTests\_008    **System under test:** Test SUT 1    **Requested by:**

**State:** Active    **Finished tests:** 170/441

**Start time:** 5.8.2016 11:50:58

**Description:**

[Cancel this workflow instance](#)

Tests in this workflow instance

Select All    [Re-run selected tests](#)    Filter:

Currently you have 0 items selected.

| Test name   | State                 | Verdict timestamp | Worker      |
|---|-----------------------|-------------------|-------------|
| <input type="checkbox"/> MockUnitTestForFastTests.dll/MockClassAAA/RandomlyFailingMockTest_5SecSleep_ABA_11 | ✔ Test result: Passed | 5 Aug 2016 13:31  | HE4D1RKK212 |
| <input type="checkbox"/> MockUnitTestForFastTests.dll/MockClassAAC/RandomlyFailingMockTest_5SecSleep_ABC_19 | ✔ Test result: Passed | 5 Aug 2016 13:31  | HE4D1RKK212 |
| <input type="checkbox"/> MockUnitTestForFastTests.dll/MockClassAAE/RandomlyFailingMockTest_30SecSleep_AAA_2 | ✔ Test result: Passed | 5 Aug 2016 13:31  | HE4D1RKK212 |
| <input type="checkbox"/> MockUnitTestForFastTests.dll/MockClassAAG/RandomlyFailingMockTest_30SecSleep_AAA_2 | ✔ Test result: Passed | 5 Aug 2016 13:29  | HE4D1RKK212 |
| <input type="checkbox"/> MockUnitTestForFastTests.dll/MockClass_AllFail/FailingMockTest_5SecSleep_ABA_12    | ✘ Test result: Failed | 5 Aug 2016 13:28  | HE4D1RKK212 |
| <input type="checkbox"/> MockUnitTestForFastTests.dll/MockClassAAA/RandomlyFailingMockTest_5SecSleep_AAA_09 | ✔ Test result: Passed | 5 Aug 2016 13:28  | HE4D1RKK212 |
| <input type="checkbox"/> MockUnitTestForFastTests.dll/MockClassAAD/RandomlyFailingMockTest_5SecSleep_ABA_17 | ✘ Test result: Failed | 5 Aug 2016 13:28  | HE4D1RKK212 |
| <input type="checkbox"/> MockUnitTestForFastTests.dll/MockClassAAF/RandomlyFailingMockTest_5SecSleep_ABA_13 | ✘ Test result: Failed | 5 Aug 2016 13:27  | HE4D1RKK212 |
| <input type="checkbox"/> MockUnitTestForFastTests.dll/MockClass_AllPass/PassingMockTest_5SecSleep_AAA_10    | ✔ Test result: Passed | 5 Aug 2016 13:27  | HE4D1RKK212 |
| <input type="checkbox"/> MockUnitTestForFastTests.dll/MockClass_AllFail/FailingMockTest_5SecSleep_ABA_11    | ✘ Test result: Failed | 5 Aug 2016 13:27  | HE4D1RKK212 |

Figure B.4: The old ASP.NET frontend Workflow Instance page.