FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Engenharia reversa de padrões de interação

Clara Raquel da Costa e Silva Sacramento

DISSERTATION PLANNING

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Ana Paiva (PhD)

February 6, 2014

Engenharia reversa de padrões de interação

Clara Raquel da Costa e Silva Sacramento

Mestrado Integrado em Engenharia Informática e Computação

February 6, 2014

This work is financed by the ERDF - European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT - Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project FCOMP-01-0124-FEDER-020554

Abstract

Resumo

Acknowledgements

Contents

CONTENTS

List of Figures

LIST OF FIGURES

List of Tables

Abbreviations

FEUP	Faculty of Engineering of the University of Porto *(Faculdade de Engenharia da Universidade do Porto)*
RIA	Rich Internet Applications
API	Application Programming Interface
AST	Abstract Syntax Tree
AUA	Application Under Analysis
CIO	Concrete Interaction Objects
EFG	Event Flow Graph
FSM	Finite State Machine
GUI	Graphical User Interface
MBT	Model-Based Testing
SU	System Under Testing
TDD	Test-Driven Development
UI	User Interface
HTML	HyperText Markup Language
UML	Unified Modeling Language
AUIDL	Abstract UI Description Language
HCI	Human Computer Interaction
REGUI	Reverse Engineering of Graphical User Interface
V&V	Verification and Validation
XML	eXtensible Markup Language

Chapter 1

Introduction

This chapter aims at giving a general overview about the themes addressed by this thesis. We will address the context in which the thesis is inserted, as well as the motivation that led to its proposal. Furthermore there will be a brief description of the main objectives of this thesis, and the methods that will be used to achieve those objectives.

1.1 Context

Web applications are getting more and more important. Due to their stability and security against losing data, there is a growing trend to move applications towards the Web, with the most notorious examples being Google's mail and office software applications. Web applications can now handle tasks that before could only be performed by desktop applications [G+05], like editing images or creating spreadsheet documents.

Despite the relevance that Web applications have in the community, they still suffer from a lack of standards and conventions [CL02], unlike desktop and mobile applications. This means that the same task can be implemented in many different ways, which makes automated testing difficult to accomplish and inhibits reuse of testing code.

GUIs *(Graphical User Interfaces)* of all kinds are populated with recurring behaviours that vary slightly. For example, authentication *(login/password)* is a common behaviour in many software applications. However, the implementation of those behaviours may vary significantly. For a login, in some cases an error message may appear when the authentication fails; in others, the software application simply erases the inserted data and doesn't send a message to the user. These behaviours (patterns) are called User Interface (UI) patterns [VWVDVE01] and are recurring solutions that solve common design problems. Due to their widespread use, UI patterns allow users a sense of familiarity and comfort when using applications.

1.2 Motivation and Objectives

The proposal aims to continue the work done on a reverse engineering tool, PARADIGM-RE, designed to extract common interface patterns from Web applications [NPCF13]. This tool identifies interface patterns

Considering the problem described and the proposed solution, the main goal for this research work is

1.3 Expected Contributions

1.4 Structure of the Report

This document is structured into four main chapters. In this first section, Chapter 1, we start by introducing the theme to be developed during the course of the dissertation, starting by defining the context and issue at hand and describing the goals of this thesis.

Chapter 2 introduces essential concepts to understand the problems with which this document deals, presents the state of the art of approaches that reverse-engineer Web applications, and lastly, gives some insight about data mining algorithms and how they will be applied to this work.

Chapter 3 explains the solution proposed to the problem at hand. This chapter itself is also divided into several sections. First, to give a better understanding on the context in which the tool to develop will be implemented, a brief overview of the PARADIGM-RE tool is provided. Following this exposition, one then explains the approaches the author is considering to take, although these might be subject to change as the work progresses.

Chapter 4 outlines the main steps in the development of this thesis (and the respective software prototype) and attempts to provide a feasible schedule for the execution of the work to be done.

Chapter 5 sums up the what has been defined in the report, emphasizing the problem that the thesis addresses and the work that will be executed towards solving that problem. It will also give a brief idea of what are the expected results at the end of the project.

Chapter 2

State-of-the-Art

2.1 Introduction

In order to find the best approach to this problem, some research on already existing methodologies and concepts was needed. First an overview for the general categories researched is given, followed by the actual state of the art found, divided by relevant subcategories.

2.2 Reverse Engineering

Reverse engineering is "the process of analysing the subject system to identify the system components and interrelationships and to create representations of the system in another form or at a higher level of abstraction" [CC$^+$90].

There are two methods of applying reverse engineering to a system: the dynamic method, in which the data are retrieved from the system at run time without access to the source code, and the static method, which obtains the data from the system source code [Sys99]. There is also the hybrid method, which combines the two previous methods, and the historical method, which includes historic information to see the evolution of the software system [CDPC11]. These approaches follow the same main steps: collect the data, and analyse it and represent it in a legible way, and in the process allow the discovery of information about the system's control and data flow [PRW03].

2.2.1 Extraction of Information from Execution Traces

Plenty of approaches that extract information from execution traces have been found. TraceServer [AA11] is an extension of the Java PathFinder model checking tool [jpf] which collects and analyzes execution traces. jRapture [SCFP00] is a technique and a tool for capture and replay of Java program executions. ReGUI [CMPPF11, CMPPF12] is a dynamic reverse engineering tool made to reduce the effort of modelling the structure and behaviour of a software application GUI.

Duarte, Kramer and Uchitel defined an approach for behaviour model extraction which combines static and dynamic information [DKU06]. Fischer *et al.* developed a methodology that analyzes and compares execution traces of different versions of a software system to provide insights into its evolution, named EvoTrace [FOGG05]. Amalfitano's approach [AFT10] generates test cases from execution traces to help testing from Rich Internet Applications (RIAs), with the execution traces being obtained from user sessions and crawling the application.

2.2.2 Extraction of Information from Web Applications

The following approaches extract information from Web applications for analysis and processing.

Ricca and Tonella's ReWeb [RT01] dynamically extracts information from a Web application's server logs to analyze its structure and evolution, and so aims to find inconsistencies and connectivity problems. Benedikt *et al.* introduced a framework called VeriWeb [BFG02] that discovers and explores automatically Web-site execution paths that can be followed by a user in a Web application. Di Lucca *et al.*'s approach [DLDP05] integrates WARE [DLFT04], a static analysis tool that generates UML diagrams from a Web application's source code, and WANDA [ADPZ04], a Web application dynamic analysis tool, to identify groups of equivalent built client pages and to enable a better understanding of the aplication under study. Crawljax [Roe10] is a tool that obtains graphical sitemaps by automatically crawling through a Web application. WebDiff [CVO10] is a tool that searches for cross-browser inconsistencies by analyzing a Website's DOM and comparing screenshots obtained in different browsers. Mesbah *et al.* proposed an automated technique for generating test cases with invariants from models inferred through dynamic crawling [MvDR12]. Artzi *et al.* developed a tool called Artemis [ADJ$^+$11] which performs feedback-directed random test case generation for Javascript Web applications. Artemis triggers events at random, but the events are prioritized by less covered branch coverage in previous sequences. Amalfitano *et al.* developed a semi-automatic approach [AFT11] that uses dynamic analysis of a Web application to generate end user documentation, compliant with known standards and guidelines for software user documentation. Another approach by Mesbah *et al.*, named FeedEx [FM13] is a feedback-directed Web application exploration technique to derive test models. It uses a greedy algorithm to partially crawl a RIA's GUI, and the goal is that the derived test model capture different aspects of the given Web application's client-side functionality. Dallmeier *et al.*'s Webmate [DBOZ12, DBOZ13] is a tool that analyzes the Web application under test, identifies all functionally different states, and is then able to navigate to each of these states at the user's request.

2.2.3 Capture-Replay Tools

Selenium [sel] is an open-source capture/replay tool that captures an user's interaction with a Web application in HTML files. It has multi browser, OS and language support, can be installed server-side and as a Mozilla Firefox add-on, has its own IDE *(Integrated Development Environment)*, and allows recording and playback of tests.

Watir Webdriver [wat] is is an open-source (BSD) family of Ruby libraries for automating Web browsers and Web application testing. It has multi browser and OS support, a rich API, and has a functionality for non-tech users: the ‘Simple’ class. There also exist ports for other programming languages, such as Watij (for Java) and Watin (.NET).

IBM Rational Functional Tester (RFT) [rft] is an automated functional testing and regression testing tool. This software provides automated testing capabilities for functional, regression, GUI, and data-driven testing. Rational Function Tester supports a range of applications, such as .Net, Java, Siebel, SAP, terminal emulator-based applications, PowerBuilder, Ajax, Adobe Flex, and others. It permits storyboard testing, automated testing, data-driven testing, and test scripting.

Sahi [sah] is an open-source automation and testing tool for web applications. It allows recording and replaying across browsers, provides different language drivers for writing test scripts, and supports Ajax and highly dynamic web applications.

2.3 Data Mining

A data mining algorithm is a set of heuristics and calculations that creates a data mining model from data [HKP06]. To create a model, the algorithm first analyzes the data provided and looks for specific types of patterns or trends. The algorithm uses the results of this analysis to define the optimal parameters for creating the mining model. These parameters are then applied across the entire data set to extract actionable patterns and detailed statistics. The mining model that an algorithm creates from the provided data can take various forms, including:

- A set of clusters that describe how the cases in a dataset are related;
- A decision tree that predicts an outcome, and describes how different criteria affect that outcome;
- A mathematical model that forecasts sales;
- A set of rules that describe how products are grouped together in a transaction, and the probabilities that products are purchased together.

2.3.1 Algorithm Categories

The broad categories that data mining algorithms belong to are:

Classification algorithms predict one or more discrete variables, based on the other attributes in the dataset.

Regression algorithms predict one or more continuous variables, such as profit or loss, based on other attributes in the dataset.

Segmentation algorithms divide data into groups, or clusters, of items that have similar properties.

Association algorithms find correlations between different attributes in a dataset. The most common application of this kind of algorithm is for creating association rules, which can be used in a market basket analysis.

Sequence analysis algorithms summarize frequent sequences or episodes in data, such as a Web path flow.

2.3.2 Data Analysis Tools

are

2.4 Patterns

User Interaction (UI) patterns are well-documented in a various number of sources [Tid10, VWVDVE01, Nei, SGR+05]. The patterns already supported (like the Search and Master/Detail patterns) enter the list of most popular patterns, according to the sources found, and if the selection of supported patterns were to be broadened, the pick of the next one(s) would be heavily influenced by the literature.

2.5 Chapter Conclusions

Chapter 3

Suggested Approach

This chapter aims to give a brief overview of the PARADIGM-RE tool Following this exposition and explain the approaches the author is considering to take.

3.1 PARADIGM-RE

In this section, we'll provide a brief overview of the tool which this thesis means to improve, PARADIGM-RE [NPCF13].

3.2 Solution Approach

Suggested Approach

Chapter 4

Work Plan

The work plan for the proposed project can be divided into the following major tasks:

- State of the Art Research;
- Study of the existing PARADIGM-RE tool;
- Choice of approach to follow;
- Implementation/adaptation of the learning algorithm, additional patterns, and model export module;
- Period dedicated to running the algorithm on learning GUIs;
- Period dedicated to testing and validating the results obtained;
- Writing of a scientific report;
- Writing of the dissertation document

Whilst the dates for each work section are defined by this point, they could be subject to change along the course of the project. Since the periods in which each task is set to be worked upon are not independent, we believe the overall work structure in relation to the time available can be better understood by use of a Gantt diagram (Figure 4.1).

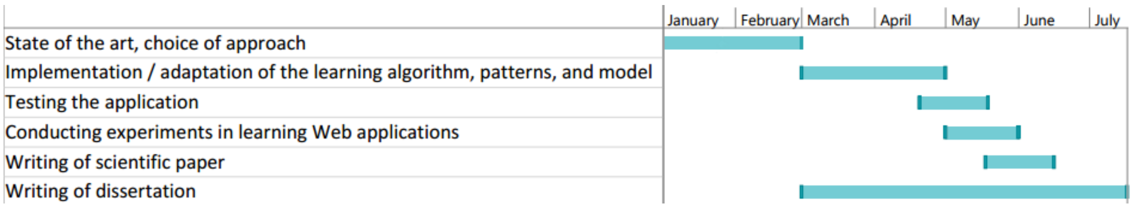

Figure 4.1: Gantt chart representing the proposed work plan.

As of the moment of the writing of this article, the research on both the state of the art regarding reverse engineering approaches, learning algorithms and the familiarization process with the

PARADIGM-RE tool will carry on being done until the ending of February. Following these steps, the effective work is then ready to be started, comprising a phase which should last until roughly the end of April. By then, the work developed thus far will undergo a learning phase (realization of experiences on learning GUIs, for the learning algorithm to draw conclusions on heuristics) and a testing and validation phase (of the newly developed patterns identified, the XMI model production and of the heuristics gotten via the learning algorithm, respectively) making adjustments as needed. This phase is expected to be concluded until the month of May at most.
This early deadline aims to make time to write a scientific paper, as well as to wrap up the dissertation document, which will be progressing in parallel with the previous phases. All the work here detailed is expected to be done by June of the current year.

Chapter 5

Conclusions

References

[AA11] Igor Andjelkovic and Cyrille Artho. Trace server: A tool for storing, querying and analyzing execution traces. In *JPF Workshop, Lawrence, USA*, 2011.

[ADJ$^+$11] Shay Artzi, Julian Dolby, Simon Holm Jensen, Anders Moller, and Frank Tip. A framework for automated testing of javascript web applications. In *Software Engineering (ICSE), 2011 33rd International Conference on*, pages 571–580. IEEE, 2011.

[ADPZ04] Giuliano Antoniol, Massimiliano Di Penta, and Michele Zazzara. Understanding web applications through dynamic analysis. In *Program Comprehension, 2004. Proceedings. 12th IEEE International Workshop on*, pages 120–129. IEEE, 2004.

[AFT10] Domenico Amalfitano, Anna Rita Fasolino, and Porfirio Tramontana. Rich internet application testing using execution trace data. In *Software Testing, Verification, and Validation Workshops (ICSTW), 2010 Third International Conference on*, pages 274–283. IEEE, 2010.

[AFT11] Domenico Amalfitano, Anna Rita Fasolino, and Porfirio Tramontana. Using dynamic analysis for generating end user documentation for web 2.0 applications. In *Web Systems Evolution (WSE), 2011 13th IEEE International Symposium on*, pages 11–20. IEEE, 2011.

[BFG02] Michael Benedikt, Juliana Freire, and Patrice Godefroid. Veriweb: Automatically testing dynamic web sites. In *In Proceedings of 11th International World Wide Web Conference (WW W'2002*. Citeseer, 2002.

[CC$^+$90] Elliot J Chikofsky, James H Cross, et al. Reverse engineering and design recovery: A taxonomy. *Software, IEEE*, 7(1):13–17, 1990.

[CDPC11] Gerardo Canfora, Massimiliano Di Penta, and Luigi Cerulo. Achievements and challenges in software reverse engineering. *Communications of the ACM*, 54(4):142–151, 2011.

[CL02] Larry L Constantine and Lucy AD Lockwood. Usage-centered engineering for web applications. *Software, IEEE*, 19(2):42–50, 2002.

[CMPPF11] Inês Coimbra Morgado, Ana Paiva, and João Pascoal Faria. Reverse engineering of graphical user interfaces. In *ICSEA 2011, The Sixth International Conference on Software Engineering Advances*, pages 293–298, 2011.

[CMPPF12] Inês Coimbra Morgado, Ana CR Paiva, and João Pascoal Faria. Dynamic reverse engineering of graphical user interfaces. *International Journal On Advances in Software*, 5(3 and 4):224–236, 2012.

[CVO10] Shauvik Roy Choudhary, Husayn Versee, and Alessandro Orso. Webdiff: Automated identification of cross-browser issues in web applications. In *Software Maintenance (ICSM), 2010 IEEE International Conference on*, pages 1–10. IEEE, 2010.

[DBOZ12] Valentin Dallmeier, Martin Burger, Tobias Orth, and Andreas Zeller. Webmate: a tool for testing web 2.0 applications. In *Proceedings of the Workshop on JavaScript Tools*, pages 11–15. ACM, 2012.

[DBOZ13] Valentin Dallmeier, Martin Burger, Tobias Orth, and Andreas Zeller. Webmate: Generating test cases for web 2.0. In *Software Quality. Increasing Value in Software and Systems Development*, pages 55–69. Springer, 2013.

[DKU06] Lucio Mauro Duarte, Jeff Kramer, and Sebastian Uchitel. Model extraction using context information. In *Model Driven Engineering Languages and Systems*, pages 380–394. Springer, 2006.

[DLDP05] Giuseppe A Di Lucca and Massimiliano Di Penta. Integrating static and dynamic analysis to improve the comprehension of existing web applications. In *Web Site Evolution, 2005.(WSE 2005). Seventh IEEE International Symposium on*, pages 87–94. IEEE, 2005.

[DLFT04] Giuseppe Antonio Di Lucca, Anna Rita Fasolino, and Porfirio Tramontana. Reverse engineering web applications: the ware approach. *Journal of Software Maintenance and Evolution: Research and Practice*, 16(1-2):71–101, 2004.

[FM13] A Milani Fard and Ali Mesbah. Feedback-directed exploration of web applications to derive test models. In *Proceedings of the 24th IEEE International Symposium on Software Reliability Engineering (ISSRE). IEEE Computer Society*, page 10, 2013.

[FOGG05] Michael Fischer, Johann Oberleitner, Harald Gall, and Thomas Gschwind. System evolution tracking through execution trace analysis. In *Program Comprehension, 2005. IWPC 2005. Proceedings. 13th International Workshop on*, pages 237–246. IEEE, 2005.

[G$^+$05] Jesse James Garrett et al. Ajax: A new approach to web applications, 2005.

[HKP06] Jiawei Han, Micheline Kamber, and Jian Pei. *Data mining: concepts and techniques*. Morgan kaufmann, 2006.

[jpf] Java path finder. Accessed: 2014-01-21.

[MvDR12] Ali Mesbah, Arie van Deursen, and Danny Roest. Invariant-based automatic testing of modern web applications. *Software Engineering, IEEE Transactions on*, 38(1):35–53, 2012.

[Nei] T. Neil. 12 standard screen patterns. Accessed: 2014-01-22.

[NPCF13] Miguel Nabuco, Ana CR Paiva, Rui Camacho, and Joao Pascoal Faria. Inferring ui patterns with inductive logic programming. In *Information Systems and Technologies (CISTI), 2013 8th Iberian Conference on*, pages 1–5. IEEE, 2013.

[PRW03] Michael J Pacione, Marc Roper, and Murray Wood. A comparative evaluation of dynamic visualisation tools. In *10th Working Conference on Reverse Engineering*, pages 80–89, 2003.

[rft] Ibm - rational functional tester - united states. Accessed: 2014-02-06.

[Roe10] Danny Roest. Automated regression testing of ajax web applications. Master's thesis, Delft University of Technology, February 2010.

[RT01] Filippo Ricca and Paolo Tonella. Understanding and restructuring web sites with reweb. *Multimedia, IEEE*, 8(2):40–51, 2001.

[sah] Sahi web test automation tool. Accessed: 2014-02-06.

[SCFP00] John Steven, Pravir Chandra, Bob Fleck, and Andy Podgurski. *jRapture: A capture/replay tool for observation-based testing*, volume 25. ACM, 2000.

[sel] Selenium documentation. Accessed: 2014-01-22.

[SGR$^+$05] Daniel Sinnig, Ashraf Gaffar, Daniel Reichart, Peter Forbrig, and Ahmed Seffah. Patterns in model-based engineering. In *Computer-Aided Design of User Interfaces IV*, pages 197–210. Springer, 2005.

[Sys99] Tarja Systä. Dynamic reverse engineering of java software. In *ECOOP Workshops*, pages 174–175, 1999.

[Tid10] Jenifer Tidwell. *Designing interfaces*. O'Reilly, 2010.

[VWVDVE01] Martijn Van Welie, Gerrit C Van Der Veer, and Anton Eliëns. Patterns as tools for user interface design. In *Tools for Working with Guidelines*, pages 313–324. Springer, 2001.

[wat] Watir webdriver | the most elegant way to use webdriver with ruby. Accessed: 2014-01-22.