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)

January 31, 2014

Engenharia reversa de padrões de interação

Clara Raquel da Costa e Silva Sacramento

Mestrado Integrado em Engenharia Informática e Computação

January 31, 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

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.

1.1 Context and Motivation

1.2 Problem Description

1.3 Goals

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

1.4 Structure of the Report

Besides the introduction chapter, this document is composed three additional chapters. These chapters have the following structure:

Chapter 2 introduces essential concepts to understand the problems with which this document deals. Furthermore, we describe the [cenas]. Lastly, we give some insight about data mining algorithms and how they will be applied to this work.

Chapter 3 outlines the main steps in the development of this thesis (and the respective software prototype) and attempts to provide a feasible schedule for the work's execution.

Chapter 4 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 a 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.

2.2.3 Dynamic Approaches

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. Mesbah *et al.* proposed [**?**] an automated technique for generating test cases with invariants from models inferred through dynamic crawling. Another approach by Mesbah *et al.*, named FeedEx [**?**] uses a greedy algorithm to partially crawl a RIA's GUI, in order to derive test models. Benedikt *et al.* introduced a framework called Veriweb [**?**] that discovers and explores automatically Web-site execution paths that can be followed by a user in a Web application. Webmate [**?**].

2.2.4 Hybrid Approaches

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. Selenium [sel] is an open-source capture/replay tool that captures an user's interaction with a Web application in HTML files.

2.3 Data Mining

Data Analysis Algorithms are

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

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;
- 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 3.1).

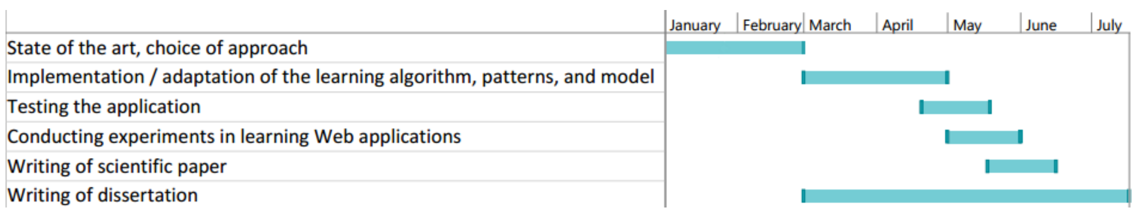

Figure 3.1: Gantt chart representing the proposed work plan.

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 4

Conclusions

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.

[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.

[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.

[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.

[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.

[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.

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

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

[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.

[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.

[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.