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 11, 2014

Engenharia reversa de padrões de interação

Clara Raquel da Costa e Silva Sacramento

Mestrado Integrado em Engenharia Informática e Computação

February 11, 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

Graphical user interfaces (*GUIs*) are populated with recurring behaviors that vary only slightly. For example, authentication (login / password) is a behavior common to many software applications. However, there are different behaviors between different implementations of this behavior. Sometimes a message appears when the user does not enter the correct data, sometimes, the application software only erases entered data and shows no indication to the user. These recurring behaviors (*UI patterns*) are well identified in the literature.

The goal of this dissertation is to continue the work already done on an existing tool called PARADIGM-RE, a dynamic reverse engineering approach to extract User Interface (UI) Patterns from existent Web applications. As such, we will develop a data analysis module with the goal of improving and substantiate the existing identifying heuristics set, and we will extend the current set of identifiable patterns.

First the theme to be developed during the course of the dissertation is introduced, starting by defining the context and issue at hand and describing the goals of this dissertation. Afterwards we present a literary review on reverse engineering approaches, approaches that infer patterns from Web applications, and data mining algorithms and tools relevant to the problem. Lastly, we will provide an estimated work plan for the project development.

Resumo

As interfaces gráficas estão populadas de comportamentos recorrentes que variam apenas ligeiramente. Por exemplo, a autenticação (*login/password*) é um comportamento comum a muitas aplicações de software. No entanto, há comportamentos diferentes entre diferentes implementações desse padrão. Por vezes, aparece uma mensagem quando o utilizador não introduz os dados correctos, outras vezes, a aplicação de software apenas apaga os dados introduzidos e não apresenta indicação nenhuma ao utilizador. Estes comportamentos recorrentes (*padrões de interface*) estão bem identificados na literatura.

O objetivo deste trabalho é dar continuidade ao trabalho já realizado numa ferramenta existente chamada PARADIGM-RE, uma abordagem de engenharia reversa dinâmica para extrair padrões de interface de aplicações Web existentes. Como tal, vamos desenvolver um módulo de análise de dados, cujo objetivo é melhorar e fundamentar as heurísticas de identificação existentes definidas, e vamos ampliar o atual conjunto de padrões identificáveis.

Primeiro é introduzido o tema a ser desenvolvido durante da dissertação, a começar por definir o contexto e o assunto em questão e descrever os objetivos desta dissertação. Em seguida apresentamos uma revisão literária sobre abordagens de engenharia reversa, abordagens que inferem padrões de aplicações Web, e os algoritmos e ferramentas relevantes para o problema de análise de dados. Por fim, iremos fornecer um plano de trabalho estimado para o desenvolvimento do projeto.

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 dissertation. We will address the context in which the dissertation 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 dissertation, 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 behaviors that vary slightly. For example, authentication *(login/password)* is a common behavior in many software applications. However, the implementation of those behaviors 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 behaviors (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

This dissertation is part of an investigation project named PBGT *(Pattern-based GUI Testing)* [MPM13]. The goal of this investigation project is to develop a model-based GUI testing tool and approach, usable as an industrial tool. This project has five parts: a DSL *(Domain Specific Language)* named **PARADIGM** to define GUI testing models based on UI patterns; a modeling and testing environment, named **PARADIGM-ME**, made to support the creation of test models; an automatic test case generation tool, named **PARADIGM-TG**, that generates test cases from test models defined in PARADIGM; a test case execution tool, named **PARADIGM-TE**, which executes test cases, analyzes their coverage, and returns detailed execution reports; and finally **PARADIGM-RE**, a Web application reverse engineering tool whose purpose is to extract UI patterns from Web pages without access to their source code, and use the extracted patterns to generate a test model defined in PARADIGM.

The relationship between the different components can be better understood in Figure 1.1. The activities (rounded corner rectangles) with the human figure mean that they are not fully automatic requiring manual intervention. The activities with the cog mean that part (or all) of that activity is automatic. The numbers within the activities define their sequencing.

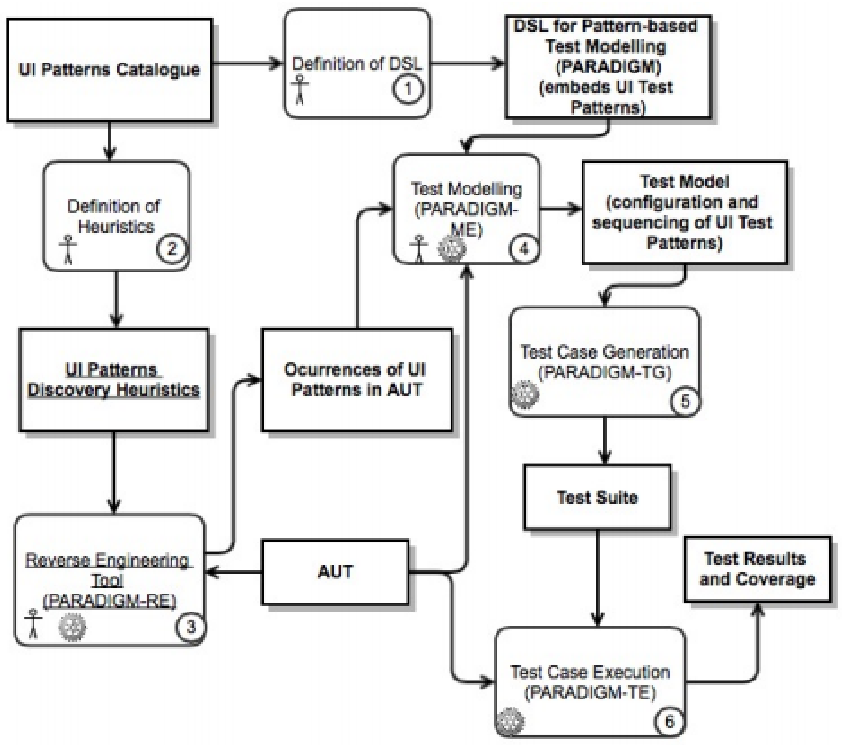

UI Patterns Catalogue
Definition of DSL
1
DSL for Pattern-based Test Modelling (PARADIGM) (embeds UI Test Patterns)
Definition of Heuristics
2
Test Modelling (PARADIGM-ME)
4
Test Model (configuration and sequencing of UI Test Patterns)
UI Patterns Discovery Heuristics
Ocurrences of UI Patterns in AUT
Test Case Generation (PARADIGM-TG)
5
Test Suite
Test Results and Coverage
Reverse Engineering Tool (PARADIGM-RE)
3
AUT
Test Case Execution (PARADIGM-TE)
6

Figure 1.1: An overview of the PBGT project [NPCF13]

The proposal aims to continue the work done on PARADIGM-RE [NPCF13]. This tool identifies interface patterns using Machine Learning inference with the Aleph ILP system [1] running on user interaction execution traces produced using Selenium [2]. It was deemed necessary then to transform the whole process into an iterative one, with the model being updated at every iteration.

This was accomplished in [NPF14], where the tool was extended with a pattern identifying module using heuristics. The current structure of the tool can be seen in Figure 1.2.

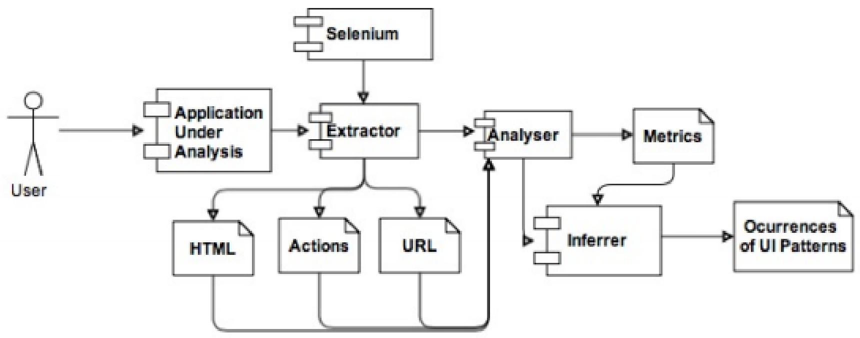


Figure 1.2: Structure of the PARADIGM-RE tool [NPF14]

The user interacts with the Web application, using Selenium to save the actions taken. An example of execution traces saved by Selenium can be seen on table 1.1.

amazon		
actionType	**element**	**text**
open	/	
type	id=twotabsearchtextbox	tablet
clickAndWait	css=input.nav-submit-input	
select	id=sort	label=Most Popular
click	id=pagnNextString	
click	id=pagnNextString	
clickAndWait	link=Image	
clickAndWait	link=Detail	
click	link=android tablet	
click	css=li.refinementImage >a.. >span.refinementLink	
clickAndWait	css=#result_3 >h3.newaps >a >span.lrg.bold	
clickAndWait	link=Explore similar items	

Table 1.1: An example of execution traces produced on the Amazon.com website.

The **extractor** saves the HTML of all pages visited, their URLs, and the actions taken in each page. All that information is passed along to the **analyzer**, whose purpose is to produce metrics like page ratios, differences between consecutive pages, and others, from the data given.

[1]Aleph: http://www.cs.ox.ac.uk/activities/machlearn/Aleph/aleph_toc.html
[2]Selenium: http://docs.seleniumhq.org/

Those metrics are passed to the **inferrer**, who runs the heuristics suite, identifies existing patterns, and produces a XMI file with the occurrences found.An example of the contents of such a file, produced consuming the execution traces in Table 1.1, can be found on Listing 1.1.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Paradigm:Model xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
  xmlns:Paradigm="http://www.example.org/Paradigm" title="patterns"/>
  <nodes xsi:type="Paradigm:Init" name="XInit" number="1"/>
  <nodes xsi:type="Paradigm:Sort" name="Sort1" number="2"/>
  <nodes xsi:type="Paradigm:End" name="End" number="3"/>
</Paradigm:Model>
```

Listing 1.1: An example of a .paradigm file with identified patterns

This approach was evaluated on several worldwide used Web applications and the results were deemed satisfactory, since the tool identified most of the occurring patterns and their location on the page. However, there are some patterns the tool struggles with, such as the Menu pattern, and the heuristics are considered to be still in an incipient state.

1.3 Expected Contributions

As stated before, the main goal for this research work is to improve and continue the work on the PARADIGM-RE. The primary task is to adapt a learning algorithm for the tool, in order to improve the existing pattern identifying heuristics, and the other goals are extending the existent identification of patterns and implementing the prodution of a PARADIGM model for the PARADIGM-ME tool to process. As such, either an existing data mining tool will be used in conjunction with the PARADIGM-RE tool to create a data model, or a data mining framework will be integrated into the tool. Examples of tools and frameworks considered can be found in Section 2.3.2.

For the identification of patterns, the information available is the execution traces produced by a user and the HTML code and URLs of all the visited pages. Since execution traces can be considered paths in which we wish to extract sub-paths, **association rule learning** have been considered, along with its subtype, **sequential pattern mining**. Sequential pattern mining is preferred because general association rule mining typically does not consider the order of items, as opposed to sequential pattern mining. Another possible venue to pursue are **classification** algorithms, in which the classes would be the current identifiable patterns, and **clustering** algorithms. This alternative probably won't be followed, since the data is mostly text and most classification and clustering algorithms deal with numeric values. Consequently, to follow this alternative some transformation of the data would be required to pursue this approach. All study related to data mining can be found on the Section 2.3.

There is also the fact that the previously considered paths would only mine execution traces and ignore the other available data (HTML pages, URLs, metrics) so further research is needed to find a way to include that data into the pattern mining process. A possible way could be to use

Inductive Logic Programming (ILP) in which one of the steps run would be the sequential pattern mining algorithm, but it would possibly require more time than is allocated for this dissertation. Further study is needed to choose the optimal approach to follow.

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 dissertation.

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

Chapter 4 sums up the what has been defined in the report, emphasizing the problem that the dissertation 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 analyzing 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" [$\text{CC}^{+}90$].

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, 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. Elbaum [EKR03] presents a testing approach that utilizes data captured in user sessions to create test cases. Duarte, Kramer and Uchitel defined an approach for behavior model extraction which combines static and dynamic information [DKU06]. Sampath *et al.* developed an approach for achieving scalable user-session-based testing of web applications, that applies concept analysis for clustering user sessions and a set of heuristics for test case selection [SSG+07]. 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 modeling the structure and behavior of a software application GUI. 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 application under study. Bernardi *et al.* [BDLD08] presents an approach for the semi-automatic recovery of user-centered conceptual models from existing web aplications, where the models represents the application's contents, their organization and associations, from a user-centered perspective. Marchetto *et al.* proposed a state-based Web testing approach [MTR08] that abstracts the Document Object Model (DOM) into a state model, and from the model derives test cases. Crawljax [Roe10] is a tool that obtains graphical site maps 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. Memon presented an end-to-end model-based Web application automated testing approach [Mem07] by consolidating previous model development work into one general event-flow model, and employs three ESESs (event space exploration strategies) for model checking, test-case generation, and test-oracle creation. 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. Dincturk *et al.* [DCvB$^+$12] proposed a RIA crawling strategy using a statistical model based on the model-based crawling approach introduced in [BVBD$^+$11] to crawl RIAs efficiently. 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 Inferring Patterns from Web applications

Despite the fact that there are plenty of approaches to mine patterns from Web applications, no approaches have been found that infer UI patterns from Web applications beside the work this dissertation means to extend [NPCF13, MPFC12]. The approaches found deal mostly with Web mining, with the goal of finding relationships between different data or finding the same data in different formats. Brin [Bri99] presented an approach to extract relations and patterns for the same data spread through many different formats. Chang [CHL03] proposes a similar method to discover patterns, by extracting structured data from semi-structured Web documents. Freitag [Fre98] proposed a general-purpose relational learner for information extracting from Web applications.

2.2.4 Capture-Replay Tools

The execution traces of a Web application, on the client side, are usually captured via a capture-replay tool. Here we present the most popular capture-replay tools used nowadays.

Selenium [1] 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 [2] 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) [3] 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.

[1]Selenium: http://docs.seleniumhq.org/
[2]Watir: http://watirwebdriver.com/
[3]IBM RFT: http://www-03.ibm.com/software/products/en/functional

Sahi [4] 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

Data mining is *"(...) the non-trivial extraction of previously unknown and potentially useful information from data"* [FPSS96]. It is the analysis step of a Knowledge Discovery in Databases (KDD) process, and an interdisciplinary sub-field of computer science. It combines artificial intelligence, machine learning, statistics and database systems [CEF$^+$04].

The goal of data mining is to extract information from a dataset, or past data, and transform it into an understandable structure. Discovering information from data takes two major forms: **description** (finding human-interpretable patterns describing the data) and **prediction** (using some variables to predict unknown or future values of other variables) [MR05]. Common types of data mining analysis include:

Anomaly detection (can also be called outlier/change/deviation detection) involves getting a sense of the typical cases that the dataset tends to contain, to better detect cases that are different from the regular pattern.

Association learning aim to find correlations between different attributes in a dataset.

Cluster detection is the task of discovering groups and structures in the data that are in some way or another "similar", without using known structures in the data.

Classification classifies new cases based on pre-determined categories. Learning from a large set of pre-classified examples, algorithms can detect systematic differences between items in each group and apply these corresponding models to new classification problems.

Regression aims to fit an equation into a dataset, in order to predict one or more continuous variables, such as profit or loss, based on other attributes in the dataset.

The performance of an algorithm depends greatly on the characteristics of the data. There is no single algorithm that works best on all given problems [WM95] so in the interest of determining the best approach, the best choice is to try a wide variety and then compare their results.

2.3.1 Data Mining Algorithm Categories

In this project we will be concerned with the association and sequence mining side of data mining. Below, we will review some algorithms that can be used in these kinds of problems.

[4]Sahi: http://sahi.co.in/

2.3.1.1 Association Rules

Association rule learning is a method for discovering interesting relations between variables in large databases. It is intended to identify strong rules discovered in databases using different measures of interestingness [HKP06]. Association rules are employed today in many application areas including Web usage mining, intrusion detection, continuous production, and bioinformatics. Association rules have the form

$$Body \rightarrow Head\ [support, confidence] \tag{2.1}$$

(for a definition of support and confidence, please check Section 2.3.1.3). Association rule mining can be broadly classified into categories: **boolean** or **quantitative** associations, **single dimension** or **multidimensional** associations, **single level** or **multilevel** associations. As opposed to sequence mining, association rule learning typically does not consider the order of items either within a transaction or across transactions.

To better explain these categories, we present two equations:

$$buys(x, SQLServer) \cap buys(x, DMBook) \rightarrow buys(x, DBMiner\ [5\%, 65\%] \tag{2.2}$$

$$age(x, [30..39]) \cap income(x, [42..50]) \rightarrow buys(x, PC)\ [1\%, 75\%] \tag{2.3}$$

Boolean and **quantitative** association rules difer mainly on the type of values handled (for example, equation 2.2 is a boolean rule, while equation 2.3 is a quantitative rule).

Single dimension and **multidimensional** association rules difer in the number of dimensions/predicates employed in the rule (the equation 2.2 is a single-dimension rule, since it only uses one dimension (*buys*) while the equation 2.3 is multidimensional because it includes three dimensions, *age*, *income*, and *buys*). **Multidimensional** association rules themselves have two subtypes: **inter-dimension** association rules (no repeated predicates on the left and right sides of the rule) and **hybrid** association rules (repeated predicates on the left and right sides of the rule). Equation 2.3 is an inter-dimension rule.

Single level association rules only have one level of concept abstraction, while **multi-level** association rules have more than one (for example, we don't have only bread, we have different brands of bread like wheat bread and rye bread, who are still bread themselves). An example of a multilevel rule could be 80% of customers who buy milk also buy bread, and inside that 80%, 75% of customers who buy skim milk also buy wonder bread.

A notable and popular association rule algorithm is the **Apriori** algorithm [WKQ+08]. **Apriori** [AS+94] is a seminal algorithm for finding frequent itemsets using candidate generation [AS+94]. It is characterized as a level-wise complete search algorithm using anti-monotonicity of itemsets, "if an itemset is not frequent, any of its superset is never frequent". By convention, Apriori assumes that items within a transaction or itemset are sorted in lexicographic order. Apriori first scans the database and searches for frequent itemsets of size 1 by accumulating the count for each item and collecting those that satisfy the minimum support requirement. It then iterates on the following three steps and extracts all the frequent itemsets. Ergo, Apriori uses a "bottom up" approach, where frequent subsets are extended one item at a time (a step known as *candidate generation*), and groups of candidates are tested against the data. The algorithm terminates when no further successful extensions are found.

Frequent patterns without candidate generation

As mentioned before, Apriori includes a step called “candidate generation”, in which itemsets are generated and extended iteratively. However, candidate set generation is costly, especially when there exist prolific patterns and/or long patterns [HPYM04]. The **FP-growth** algorithm [HPYM04] uses the frequent pattern tree (FP-tree) structure to mine the complete set of frequent patterns by pattern fragment growth.

Multilevel association rules

There are applications which need to find associations at multiple concept levels. For example, besides finding 80% of customers that purchase milk may also purchase bread, it could be informative to also show that 75% of people buy wheat bread if they buy skim milk. The association relationship in the latter statement is expressed at a lower concept level but often carries more specific and concrete information than that in the former. This requires progressively deepening the knowledge mining process for finding refined knowledge from data [HF99]. To find interesting relations, there need to be multiple support and confidence thresholds for different levels, and only consider the descendants of interesting rules.

Correlation rules

Association rules simply say that when X occurs, there is a chance of Y also occurring. They do not capture many interesting dependencies among items (for example, *negative relations* like “who buys coffee does not usually buy tea”). In correlation rules, the key is that buying coffee and buying tea are not associated but *correlated*, or in other words, buying coffee influences buying tea in some way [BMS97]. The key is determining if two variables are dependent on each other or not. A classical method for determining independence between variables is the *chi square test* [LS69]. Correlation rules consider items as random variables and transactions as observations of n binary random variables, and for every itemmset the chi square test is used to infer if the items involved are dependent on each other. The measure of interestingness for a correlation rule ceases to be *support* and *confidence*, and instead is used *interest*. Interest is defined in Equation 2.4.

$$interest(A,B) = \frac{prob(A \cap B)}{prob(A) \times prob(B)} \tag{2.4}$$

Pseudo-constraints

Pseudo-constraints are predicates which don't need to be true on every instance, but have such strong dependencies among data that they can almost be called constraints. Violations to pseudo-constraints are therefore interesting because of their rarity; they are anomalous and therefore interesting [CGL07]. This type of associaton rules are good for finding errors in data, cheating people, or selected market targets.

Perfectly sporadic rules

If the frequencies of items vary a great deal, we will encounter two problems. If *minsup* is set too high, those rules that involve rare items will not be found; but if *minsup* is set too low, it may cause combinatorial explosion because those frequent items will be associated with one another in all possible ways. To solve this problem, the minimum support of a rule is expressed in terms of minimum item supports of the items that appear in the rule, with each item having its minimum item support. A rule satisfies its minimum support if its actual support is bigger or equal to the minimum support value of all items involved. Notable algorithms that solve this problem are the MSApriori algorithm [LHM99] and the AprioriInverse algorithm [KR05].

Indirect rules

Consider a pair of items (A, B) with a low support value. If there is an itemset Y such that the presence of A and B are highly dependent on items in Y, then (A, B) are said to be indirectly associated via Y. It can be used to identify synonyms in text mining: for example,the words *coal* and *data* can be indirectly associated via *mining*. If a user queries the word mining, the collection of documents returned often contain a mixture of both mining context. Indirect mining allows segmentation of the document collection into different contexts [TKS00].

Sequential rules

A sequential rule (also called episode rule, temporal rule or prediction rule) indicates that if some event(s) occurred, some other event(s) are also likely to occur with a given confidence or probability. Sequential rules are different from sequence patterns in that sequence patterns are sequences that occur often in a database, while sequence rules can be used to predict events [FVNT11].

2.3.1.2 Sequential Pattern Mining

Sequence mining is a topic of Data Mining concerned with finding statistically relevant patterns between data examples where the values are delivered in a sequence. A **sequential pattern** consists of finding sequences of events that appear frequently in a sequence database. An example of a sequence could be transactions made by a customer, with frequent transactions being the patterns (a certain customer might buy milk and diapers together often). Sequence mining algorithms have two broad categories: apriori-based approaches and pattern-growth based approaches.

Apriori-based approaches

There are two important algorithms in this category: **GSP** (Generalized Sequential Pattern) algorithm and the **SPADE** algorithm.

GSP Algorithm (Generalized Sequential Pattern algorithm) [SA96] is an algorithm used for sequence mining. The algorithms for solving sequence mining problems are mostly based on the a priori (level-wise) algorithm. One way to use the level-wise paradigm is to first discover all the frequent items in a level-wise fashion, counting the occurrences of all singleton elements in the database. Afterwards, the non-frequent items are removed. At the end of this step, each transaction consists of only the frequent elements it originally contained. This modified database becomes an input to the GSP algorithm. This process requires one pass over the whole database.

SPADE [Zak01] is a fundamentally different sequential pattern algorithm. In place of repeated database scans, this method uses lattice-search techniques and simple join operations to discover all sequence patterns. First a vertical id-list is created to associate with each item, a list of the sequences in which it occurs, along with the appropriate time-stamps.

Pattern-growth based approaches

There are two important algorithms in this category: **FreeSpan** and **PrefixSpan**.

FreeSpan [HPMA$^+$00] was developed to substantially reduce the expensive candidate generation and testing of Apriori, while maintaining its basic heuristic. In general, FreeSpan uses frequent items to recursively project the sequence database into projected databases while growing subsequence fragments in each projected database. Each projection partitions the database and confines further testing to progressively smaller and more manageable units. The trade-off is a considerable amount of sequence duplication, as the same sequence could appear in more than one projected database. However, the size of each projected database usually (but not necessarily) decreases rapidly with recursion.

Prefix Span [PHMA$^+$04] was developed to address the costs of FreeSpan. Its general idea is that, instead of projecting sequence databases by considering all the possible occurrences of frequent subsequences, the projection is based only on frequent prefixes because any frequent subsequence can always be found by growing a frequent prefix.

Closed sequential pattern mining

Sequential pattern mining mine the full set of frequent subsequences satisfying the *minsup* threshold. However, since frequent long sequences contains a combination of frequent subsequences, the process will generate an explosion of frequent subsequences [YHA03]. Closed sequential pattern mining mines *frequent closed sequences* (sequences that contain no super-sequence with the same support) instead. For a definition of *support* see Equation 2.5. This kind of sequence mining produces significantly less sequences than the alternative while preserving the same expressive power [YHA03].

Multi-dimensional sequential pattern mining

Generally, sequential pattern algorithms mine only one dimension. This is fine if the transaction dataset to mine is also unidimensional (simple transactions with timestamps associated) but usually sequence patterns are associated with different circumstances (for example, customer purchases can be associated with region, customer group, date, and others). When one or more dimensions of information is mined and the order of the dimension values is not important, it is known as multi-dimensional sequential pattern mining [PHP+01]. The goal of multidimensional sequential pattern mining is to cover more useful information than regular sequential patterns.

Closed multi-dimensional sequential pattern mining

Same as single dimension sequential pattern mining, multidimensional sequencial pattern mining generally produce a large set of redundant patterns [SB08]. Since the problem of redundancy is similar to those of itemset pattern mining and sequential pattern mining, the combination of closed itemset pattern mining and closed sequential pattern mining returns closed multidimensional sequential patterns (or CMDS patterns). This method consists of two major steps; (1) combination closed itemset pattern mining with closed sequential pattern mining, and (2) elimination of redundant patterns.

2.3.1.3 Evaluation Metrics

Applying evaluation procedures is not sufficient by itself. In order to appraise the quality of the produced models, it is necessary to use standard quality measures, that give meaning to the obtained results. Below are presented some evaluation metrics used for evaluating association rules.

Itemset. An itemset is a collection of one or more items. It can also be called k-itemset, where k is its size.

Support count. The support count of an itemset is defined as its frequency of occurrence.

Support. The support of an itemset is the fraction of transactions that contain the itemset (see Equation 2.5).

$$supp(X) = \frac{itemset\_occurrences(X)}{all\_transactions} \tag{2.5}$$

Confidence. Confidence is the support for occurrences of transactions where X and Y both appear (see Equation 2.6).

$$conf(X \Rightarrow Y) = \frac{supp(X \cap Y)}{supp(X)} \tag{2.6}$$

Frequent itemset. A frequent itemset is one with support count equal or superior to the *minsup* threshold, the minimum number of occurrences specified by the user.

Lift. Lift is the ratio of the observed support to that expected if X and Y were independent (see Equation 2.7).

$$lift(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X) \times supp(Y)} \tag{2.7}$$

Conviction. Conviction can be interpreted as the ratio of the expected frequency that X occurs without Y (that is to say, the frequency that the rule makes an incorrect prediction) if X and Y were independent divided by the observed frequency of incorrect predictions (see Equation 2.8). In sum, it measures the degree of independence between two variables.

$$conv(X \Rightarrow Y) = \frac{1 - supp(Y)}{1 - conf(X \Rightarrow Y)} \tag{2.8}$$

There are also **subjective** measures of measuring a rule's interestingness [ST95]. A rule (pattern) is interesting if (1) it is *unexpected* (surprising to the user) and/or (2) *actionable* (the user can do something with it).

2.3.2 Data Mining Tools and Frameworks

Except in rare cases of very specific problems, it typically makes no sense for someone to implement any data mining algorithm that they might need. There are many data mining tools (many of which free) that already implement many of those algorithms and have customization capabilities that make it easy to adapt them to most problems; and there are also many data mining frameworks and libraries who implement a wide variety of algorithms. A data mining tool is a powerful software that makes use of data mining algorithms, and supports a complete KDD process [MR11]. In the following sections we will briefly address some of the most commonly used open-source tools and frameworks, namely RapidMiner, WEKA, SPMF, and R.

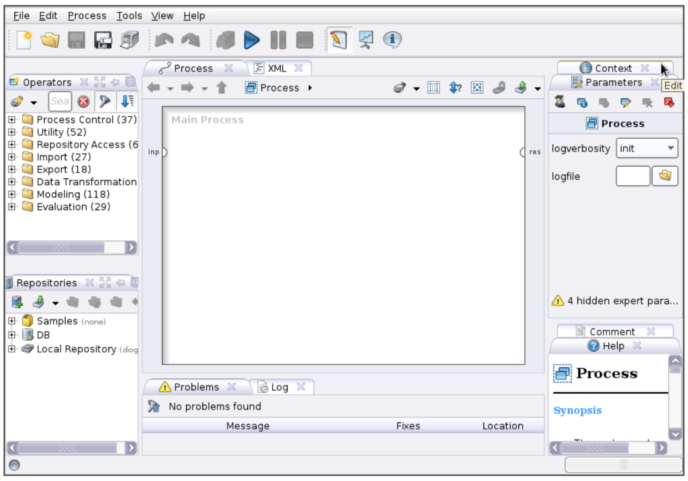
2.3.2.1 RapidMiner

RapidMiner [5] is a complete solution for data mining problems. It's available as a standalone GUI based application, as seen in Figure 2.1. It is a commercial application, although its core and earlier versions are distributed under an open source license, and it offers a free version, beyond its multiple paid versions. Being one of the most popular data mining tools used today, its applications span several domains, including education, training, industrial and personal applications, among others. Its functionality can also be easily extended through the use of plugins. reflecting in an increased value for this tool.

2.3.2.2 WEKA

Weka [6] is an open source tool that collects several machine learning algorithms and allows its user to easily apply those algorithms to data mining tasks [HKP06]. Created at the University

[5]RapidMiner: `http://www.rapidminer.com`
[6]WEKA: `http://www.cs.waikato.ac.nz/ml/weka/`

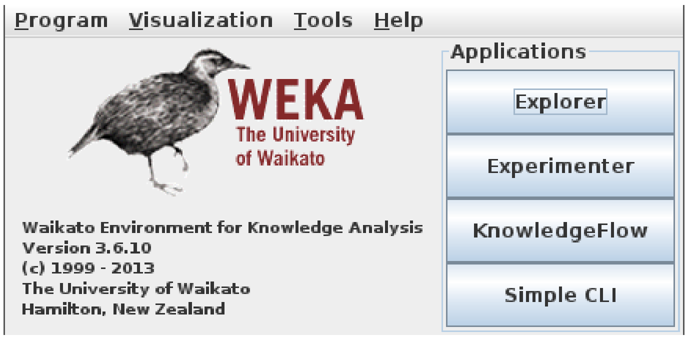


No problems found

4 hidden expert para...

Figure 2.1: RapidMiner's interface

of Waikato, New Zeland in 1997 (the current version was completely rewritten in 1997, despite the first iteration of the tool being developed as early as 1993), it's still in active development to date. Weka supports several common data mining tasks, like data preprocessing, classification, clustering, regression and data visualization. Its core libraries are written in Java and allow for an easy integration of its data mining algorithms in pre existing code and applications. Other than that, Weka can be used directly through a command line/terminal or through one of its multiple GUIs (Figure 2.2). Its simple API and well structure architecture allow it to be easily extended by users, should they need new functionalities.



Waikato Environment for Knowledge Analysis
Version 3.6.10
(c) 1999 - 2013
The University of Waikato
Hamilton, New Zealand

Figure 2.2: WEKA's interface

2.3.2.3 R

R [7] is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. R provides a wide variety of statistical (linear and nonlinear modeling, classical statistical tests, time-series analysis, classification,

[7]R: http://www.r-project.org/

clustering, among others) and graphical techniques, and is highly extensible. R is typically used by statisticians and data miners, either for direct data analysis or for developing new statistical software [FA05].
R is an open-source implementation of the S programming language, borrowing some characteristics from the Scheme programming language. Its core is written in a combination of C, Fortran and R itself. It is possible to directly manipulate R objects in languages like C, C++ and Java. R can be used directly through the command line or through several third party graphical user interfaces like Deducer [8]. There are also R wrappers for several scripting languages.
R provides several different statistical and graphical techniques, including linear and nonlinear modeling, classical statistical tests, time-series analysis, classification, clustering, among others. It can also be used to produce publication-quality static graphics. Tools like Sweave [Lei02] allow users to embed R code in LaTeXdocuments, for complete data analysis.

arules

The arules package [9] is a R package for mining association rules and frequent itemsets. Its sister package, arulesViz [10], allows the visualization of the results found by arules. Since it is common to work with large sets of rules and itemsets, the package uses sparse matrix representations to minimize memory usage. The infrastructure provided by the package was also created to explicitly facilitate extensibility, both for interfacing new algorithms and for adding new types of interest measures and associations.

TraMineR

TraMineR [11] (a contraction of Life Trajectory Miner for R) is a R-package for mining, describing and visualizing sequences of states or events, and more generally discrete sequential data. An example of the visualization features can be found in Figure 2.3. Its primary aim is the analysis of biographical longitudinal data in the social sciences, such as data describing careers or family trajectories. Most of its features also apply, however, to non temporal data. TraMineR is developed at the Institute for Demographic and Life Course Studies (IDEMO), University of Geneva, Switzerland under the responsibility of the TraMineR Scientific Committee.

2.3.2.4 SPMF (Sequential Pattern Mining Framework)

SPMF [12] is an open-source data mining library written in Java and distributed under the GPL v3 license. It includes implementations for sequential pattern mining, association rule mining, frequent itemset mining, sequential rule mining, and clustering algorithms, and has over 80 citations.

[8]Deducer: http://www.deducer.org/pmwiki/index.php
[9]arules: http://cran.r-project.org/web/packages/arules/index.html
[10]http://cran.r-project.org/web/packages/arulesViz/index.html
[11]TraMineR: http://mephisto.unige.ch/traminer/
[12]SPMF: http://www.philippe-fournier-viger.com/spmf/

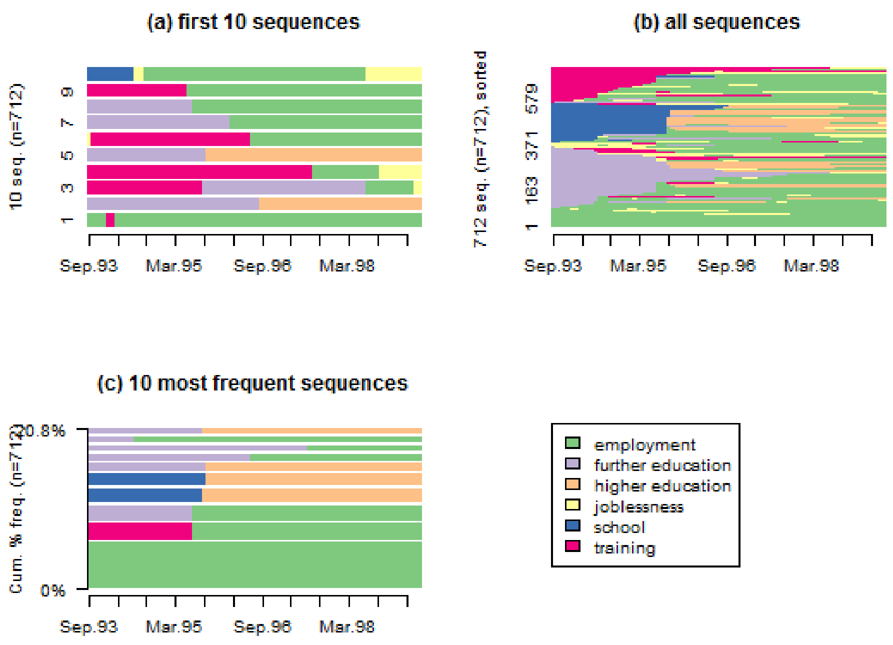


Figure 2.3: An example of data visualization using TraMineR[13]

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. Lin and Landay's approach [LL08] uses UI patterns for Web applications that run on PCs and mobile phones, and prompt-and-response style voice interfaces. Pontico *et al.*'s approach [PWL08] presents UI patterns common in eGovernment applications.

2.5 Chapter Conclusions

In this section we gave a brief introduction of concepts that serve as basis for this dissertation, namely reverse engineering and data mining, in specific association learning and its subtype, sequence pattern mining.

We also reviewed the state of the art on the following topics:

1. **Reverse engineering approaches**, subcategorized by approaches that extract information from execution traces, approaches that extract information from Web applications, and approaches that infer patterns from Web applications;

2. Popular **capture-replay tools**;

[13]Image extracted from: `http://mephisto.unige.ch/traminer/preview-visualizing.shtml`

3. Data mining **tools and frameworks**;
4. UI **patterns**.

At this moment we are still evaluating the best approach to follow regarding how we're going to analyze our data, since we are trying to find a method that efficiently evaluates all data produced by the PARADIGM-RE tool, and not just mine the execution traces. As such, we presented ILP as an alternative approach for this situation. In case ILP techniques become in fact necessary, further work will include a more profound and complete revision.

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;
- 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 3.1).

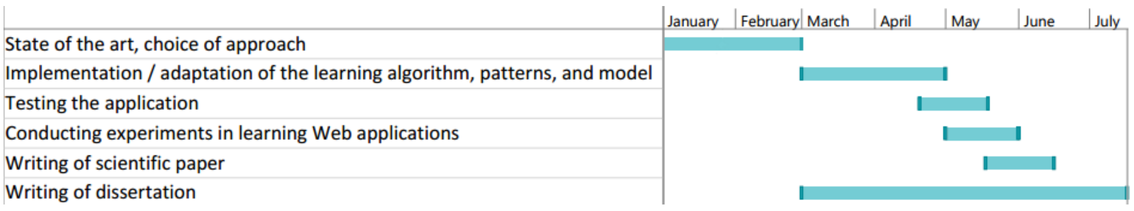


Figure 3.1: Gantt chart representing the proposed work plan.

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 produce a relevant data model) 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

In this chapter we will review the general objectives of this dissertation. We will also recap the idea behind the proposed project, along with its implementation idealization. Lastly, we will give some perspective about future work.

4.1 Objectives

As stated before, our main objective for this dissertation is to implement a data analysis module that will apply data mining techniques to the information available (user actions captured via Selenium, HTML and URLs of each page visited, metrics) and so identify UI patterns in a Web aplication. The other major goals of this dissertation are extending the existent identification of patterns and implementing the prodution of a PARADIGM model for the PARADIGM-ME tool to process

4.2 Project

This project will be the materialization of the aforementioned objectives. As such, we will be adaptating the chosen learning algorithm, implement the identification of additional patterns, and implementing the model export module.

4.3 Future Work

The project will be implemented in three main phases. The first phase is the implementation of the objectives mentioned above. The other two phases are intensive testing of the resulting application and conducting experiments in learning Web applications, respectively. The results of the data analysis module will be compared with the current identifying heuristics to check if there have been improvements compared to the previous work.

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.

[AS+94] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, volume 1215, pages 487–499, 1994.

[BDLD08] Mario Luca Bernardi, Giuseppe A Di Lucca, and Damiano Distante. Reverse engineering of web applications to abstract user-centered conceptual models. In *Web Site Evolution, 2008. WSE 2008. 10th International Symposium on*, pages 101–110. IEEE, 2008.

[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.

[BMS97] Sergey Brin, Rajeev Motwani, and Craig Silverstein. Beyond market baskets: Generalizing association rules to correlations. In *ACM SIGMOD Record*, volume 26, pages 265–276. ACM, 1997.

[Bri99] Sergey Brin. Extracting patterns and relations from the world wide web. In *The World Wide Web and Databases*, pages 172–183. Springer, 1999.

[BVBD$^+$11] Kamara Benjamin, Gregor Von Bochmann, Mustafa Emre Dincturk, Guy-Vincent Jourdan, and Iosif Viorel Onut. *A strategy for efficient crawling of rich internet applications*. Springer, 2011.

[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.

[CEF$^+$04] Soumen Chakrabarti, Martin Ester, Usama Fayyad, Johannes Gehrke, Jiawei Han, Shinichi Morishita, Gregory Piatetsky-Shapiro, and Wei Wang. Data mining curriculum: A proposal (version 0.91). 2004.

[CGL07] Stefano Ceri, Francesco Di Giunta, and Pier Luca Lanzi. Mining constraint violations. *ACM Transactions on Database Systems (TODS)*, 32(1):6, 2007.

[CHL03] Chia-Hui Chang, Chun-Nan Hsu, and Shao-Cheng Lui. Automatic information extraction from semi-structured web pages by pattern discovery. *Decision Support Systems*, 35(1):129–147, 2003.

[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.

[DCvB$^+$12] Mustafa Emre Dincturk, Suryakant Choudhary, Gregor von Bochmann, Guy-Vincent Jourdan, and Iosif Viorel Onut. A statistical approach for efficient crawling of rich internet applications. In *Web Engineering*, pages 362–369. Springer, 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.

[EKR03] Sebastian Elbaum, Srikanth Karre, and Gregg Rothermel. Improving web application testing with user session data. In *Proceedings of the 25th International Conference on Software Engineering*, pages 49–59. IEEE Computer Society, 2003.

[FA05] John Fox and Robert Andersen. Using the r statistical computing environment to teach social statistics courses. *Department of Sociology, McMaster University*, 2005.

[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.

[FPSS96] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37, 1996.

[Fre98] Dayne Freitag. Information extraction from html: Application of a general machine learning approach. In *AAAI/IAAI*, pages 517–523, 1998.

[FVNT11] Philippe Fournier-Viger, Roger Nkambou, and Vincent Shin-Mu Tseng. Rulegrowth: mining sequential rules common to several sequences by pattern-growth. In *Proceedings of the 2011 ACM symposium on applied computing*, pages 956–961. ACM, 2011.

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

[HF99] Jiawei Han and AW Fu. Mining multiple-level association rules in large databases. *Knowledge and Data Engineering, IEEE Transactions on*, 11(5):798–805, 1999.

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

[$HPMA^{+}00$] Jiawei Han, Jian Pei, Behzad Mortazavi-Asl, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu. Freespan: frequent pattern-projected sequential pattern mining. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 355–359. ACM, 2000.

[HPYM04] Jiawei Han, Jian Pei, Yiwen Yin, and Runying Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data mining and knowledge discovery*, 8(1):53–87, 2004.

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

[KR05] Yun Sing Koh and Nathan Rountree. Finding sporadic rules using apriori-inverse. In *Advances in Knowledge Discovery and Data Mining*, pages 97–106. Springer, 2005.

[Lei02] Friedrich Leisch. Sweave: Dynamic generation of statistical reports using literate data analysis. In *Compstat*, pages 575–580. Springer, 2002.

[LHM99] Bing Liu, Wynne Hsu, and Yiming Ma. Mining association rules with multiple minimum supports. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 337–341. ACM, 1999.

[LL08] James Lin and James A Landay. Employing patterns and layers for early-stage design and prototyping of cross-device user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1313–1322. ACM, 2008.

[LS69] Henry Oliver Lancaster and E Seneta. *Chi-Square Distribution*. Wiley Online Library, 1969.

[Mem07] Atif M Memon. An event-flow model of gui-based applications for testing. *Software Testing, Verification and Reliability*, 17(3):137–157, 2007.

[MPFC12] Inês Coimbra Morgado, Ana CR Paiva, Joao Pascoal Faria, and Rui Camacho. Gui reverse engineering with machine learning. In *Realizing Artificial Intelligence Synergies in Software Engineering (RAISE), 2012 First International Workshop on*, pages 27–31. IEEE, 2012.

[MPM13] Rodrigo MLM Moreira, Ana CR Paiva, and Atif Memon. A pattern-based approach for gui modeling and testing. In *Software Reliability Engineering (ISSRE), 2013 IEEE 24th International Symposium on*, pages 288–297. IEEE, 2013.

[MR05] Oded Z Maimon and Lior Rokach. *Data mining and knowledge discovery handbook*. Springer, 2005.

[MR11] Ralf Mikut and Markus Reischl. Data mining tools. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(5):431–443, 2011.

[MTR08] Alessandro Marchetto, Paolo Tonella, and Filippo Ricca. State-based testing of ajax web applications. In *Software Testing, Verification, and Validation, 2008 1st International Conference on*, pages 121–130. IEEE, 2008.

[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.

[NPF14] Miguel Nabuco, Ana CR Paiva, and Joao Pascoala Faria. Inferring user interface patterns from execution traces of web applications. Manuscript submitted for publication, 2014.

[PHMA$^+$04] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Jianyong Wang, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu. Mining sequential patterns by pattern-growth: The prefixspan approach. *Knowledge and Data Engineering, IEEE Transactions on*, 16(11):1424–1440, 2004.

[PHP$^+$01] Helen Pinto, Jiawei Han, Jian Pei, Ke Wang, Qiming Chen, and Umeshwar Dayal. Multi-dimensional sequential pattern mining. In *Proceedings of the tenth international conference on Information and knowledge management*, pages 81–88. ACM, 2001.

[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.

[PWL08] Florence Pontico, Marco Winckler, and Quentin Limbourg. Organizing user interface patterns for e-government applications. In *Engineering Interactive Systems*, pages 601–619. Springer, 2008.

[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.

[SA96] Ramakrishnan Srikant and Rakesh Agrawal. *Mining sequential patterns: Generalizations and performance improvements*. Springer, 1996.

[SB08] Panida Songram and Veera Boonjing. Closed multidimensional sequential pattern mining. *International Journal of Knowledge Management Studies*, 2(4):460–479, 2008.

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

[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.

[SSG$^+$07] Sreedevi Sampath, Sara Sprenkle, Emily Gibson, Lori Pollock, and Amie Souter Greenwald. Applying concept analysis to user-session-based testing of web applications. *Software Engineering, IEEE Transactions on*, 33(10):643–658, 2007.

[ST95] Abraham Silberschatz and Alexander Tuzhilin. On subjective measures of interestingness in knowledge discovery. In *KDD*, volume 95, pages 275–281, 1995.

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

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

[TKS00] Pang-Ning Tan, Vipin Kumar, and Jaideep Srivastava. *Indirect association: Mining higher order dependencies in data*. Springer, 2000.

[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.

[WKQ$^+$08] Xindong Wu, Vipin Kumar, J Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, S Yu Philip, et al. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37, 2008.

[WM95] David H Wolpert and William G Macready. No free lunch theorems for search. Technical report, Technical Report SFI-TR-95-02-010, Santa Fe Institute, 1995.

[YHA03] Xifeng Yan, Jiawei Han, and Ramin Afshar. Clospan: Mining closed sequential patterns in large datasets. In *Proc. 2003 SIAM Int'l Conf. Data Mining (SDM'03)*, pages 166–177, 2003.

[Zak01] Mohammed J Zaki. Spade: An efficient algorithm for mining frequent sequences. *Machine learning*, 42(1-2):31–60, 2001.