Zur Erlangung des akademischen Grades eines Doktors der Wirtschaftswissenschaften (Dr. rer. pol.) von der Fakultät für Wirtschaftswissenschaften des Karlsruher Instituts für Technologie (KIT) genehmigte Dissertation.

Adaptive and Reactive Rich Internet Applications

von

Diplom-Informatiker Kay-Uwe Schmidt

Tag der mündlichen Prüfung: 23.06.2010

Referent: Prof. Dr. Rudi Studer

Korreferent: Prof. Dr. Nicola Henze

Prüfer: Prof. Dr. Andreas Oberweis

2010 Karlsruhe

To Christine

Acknowledgements

I am grateful to all who supported me in writing this thesis. Especially, I wish to acknowledge:

my supervisor
Prof. Dr. Rudi Studer;

my co-referent
Prof. Dr. Nicola Henze;

my reviewers
Dr. Ljiljana Stojanovic and Susan Marie Thomas;

my colleagues
Darko Anicic, Klaus Deissner, Jörg Dörflinger, Dr. Elmar Dorner, Dr. Matthias Flügge, Babis Magoutas, Dr. Daniel Oberle, Tirdad Rahmani, Mehdi Sahbi, Tobias Sarnow, Axel Spriestersbach, Dr. Nenad Stojanovic and Roland Stühmer;

the SAP Research CEC Karlsruhe Scrum Team;

and, last but not least, my friends & my family.

June 2010, Karlsruhe

Kay-Uwe Schmidt

Abstract

Rich Internet Applications significantly raise the user experience compared to conventional web applications by providing highly responsive user interfaces. Although, this is already a tremendous advance in usability, it does not solve the usability issues of one-size-fits-all user interfaces. So far, research on adaptive hypermedia came up with server-side solutions for adapting web applications to the individual user. However, these approaches do not take into account the new opportunities Rich Internet Applications offer for ad-hoc personalization.

In this thesis we present the client-side approach of Adaptive and Reactive Rich Internet Applications as the main result of our research into how to bring in time adaptivity to Rich Internet Applications. Our approach leverages previous work on adaptive hypermedia, event processing and other research disciplines; and we provide a comprehensive overview of related, similar and subsumed approaches.

We present a holistic framework covering the design-time as well as the run-time aspects of Adaptive and Reactive Rich Internet Applications focusing especially on the run-time aspects. The Design-Time Framework supports the modeling of user-centric adaptation rules. As part of the Design-Time Framework we propose the application of semantic web usage mining to semantically enriched web server access log files in order to discover behavioral patterns common to a group of users. These patters serve as a basis for modeling adaptation rules.

In order to declaratively encode adaptation logic we designed the light-weight Adaptation Rule Language. This language is tailored to the needs of being executed on time on the client directly in the user's Internet browser. As user interfaces of web applications are event-driven, we based the Adaptation Rule Language on the event-condition-action paradigm.

At run-time the adaptation rules are processed by the Adaptation Engine of the Run-Time Framework directly on the client. We detail the basic principles of the Adaptation Engine, and show how it facilitates ad-hoc personalization. Instructed by declarative adaptation rules the Adaptation Engine tracks the browsing behavior of an individual user and reacts to predefined behavioral patterns by adapting the user interface of a web application. The Adaptation Engine combines a graph-based algorithm for the detection of complex events with an efficient pattern matching algorithm for executing declarative production rules.

We demonstrate the universal applicability of Adaptive and Reactive Rich Internet Applications on three use case specific scenarios: personalized e-Government, personalized web advertisement and personalized web search. Within the personalized e-Government use case we discuss the results of a comprehensive user-driven usability test in which we analyzed the user experience of an adaptive versus a non-adaptive e-Government portal. Finally, we identify a number of promising areas for future research.

Contents

Acknowledgements v

Abstract vii

Contents ix

List of Figures xiii

List of Tables xv

List of Acronyms xvii

I Problem Statement 1

1 Introduction 3
- 1.1 Motivation . . . 3
- 1.2 Research Methodology . . . 7
- 1.3 Research Challenges . . . 8
- 1.4 Research Contributions . . . 10
- 1.5 Structure of the Thesis . . . 14

2 Foundations 17
- 2.1 Rich Internet Applications . . . 18
  - 2.1.1 Rich Clients . . . 18
  - 2.1.2 Ajax and JSON . . . 19
- 2.2 Ontologies . . . 20
- 2.3 Adaptive Hypermedia . . . 22
  - 2.3.1 Hypermedia System . . . 23
  - 2.3.2 User Models . . . 23
  - 2.3.3 Adaptation Technologies . . . 24
- 2.4 Collective Intelligence . . . 24
  - 2.4.1 Web Usage Mining . . . 25
  - 2.4.2 Semantic Web Usage Mining . . . 26

2.4.3 Association Rules 28
2.5 Production Systems 29
2.5.1 The Rete Algorithm 31
2.6 Complex Event Processing 32
2.6.1 Basic Concepts of CEP 34
2.6.2 Snoop, SnoopIB and Sentinel 36
2.7 Summary 37

3 Use Case Analysis **39**
3.1 Personalized E-Government 39
3.1.1 Use Case Scenarios 42
3.2 Personalized Web Advertisement 46
3.2.1 Use Case Scenario 47
3.3 Personalized Web Search 47
3.3.1 Use Case Scenario 48
3.4 Summary 48

II Design and Implementation **49**

4 A Bird's-Eye View **51**
4.1 The Adaptation Framework 51
4.1.1 The Modeling Cycle 53
4.1.2 The Transfer Cycle 54
4.1.3 The Adaptation Cycle 55
4.2 Walkthrough Example 58
4.2.1 Design-Time Walkthrough 58
4.2.2 Run-Time Walkthrough 58
4.3 Overcoming the Research Challenges 59
4.4 Related Work 63
4.5 Summary 64

5 Obtaining the Adaptation Rules **65**
5.1 Ontology Design 67
5.1.1 Annotation Ontology 68
5.1.2 Domain Ontologies 70
5.2 Annotation of Web Applications 71
5.3 Preprocessing and Web Usage Mining 72
5.3.1 Conventional Preprocessing 73
5.3.2 Semantic Preprocessing 73
5.3.3 Pattern Discovery 77
5.4 Analysis and Rules Design 77
5.5 Evaluation 78

5.5.1 Preprocessing . . . 78
5.5.2 Annotation . . . 79
5.5.3 Mining and Analysis . . . 79
5.5.4 Discussion . . . 79
5.5.5 Privacy Concerns . . . 80
5.6 Related Work . . . 81
5.7 Summary . . . 83

6 The Adaptation Rule Language 85
6.1 Requirements . . . 85
6.1.1 Description of the Requirements . . . 86
6.1.2 Meeting the Requirements . . . 87
6.2 The Rules File . . . 89
6.3 The Repositories . . . 90
6.4 The Adaptation Rule . . . 92
6.4.1 The Event Part . . . 93
6.4.2 The Condition Part . . . 98
6.4.3 The Action Part . . . 101
6.5 Construction Wizard Example . . . 104
6.6 Related Work . . . 110
6.7 Summary . . . 111

7 The Adaptation Engine 113
7.1 Concept Level Architecture . . . 114
7.1.1 The Production Engine . . . 115
7.1.2 The Event Engine . . . 117
7.1.3 Walkthrough of the ARRIA Adaptation Cycle . . . 118
7.2 Design Level Architecture . . . 119
7.2.1 Graph Builder, Rules File and Fact Base . . . 119
7.2.2 Hybrid Inference Engine . . . 122
7.2.3 Event Sources and Event Bus . . . 126
7.2.4 Agenda . . . 127
7.3 Construction Wizard Example . . . 128
7.4 Performance Analysis . . . 129
7.4.1 Goals of the Performance Analysis . . . 130
7.4.2 Metrics . . . 130
7.4.3 Design of the Experiments . . . 130
7.4.4 Analysis, Presentation and Interpretation of Data . . . 131
7.5 Related Work . . . 133
7.6 Summary . . . 135

III Demonstration and Evaluation 137

8 Personalized E-Government 139
8.1 The Adaptive City Portal of Vöcklabruck . . . 139
8.2 Comparative Usability Test . . . 141
8.2.1 Test Hypotheses . . . 141
8.2.2 Test Planning . . . 142
8.2.3 Architecture and Implementation . . . 147
8.2.4 Execution and Test Results . . . 149
8.3 Summary . . . 155

9 Personalized Web Advertisement 157
9.1 Architecture . . . 158
9.1.1 Design-Time Framework . . . 158
9.1.2 Run-Time Framework . . . 161
9.2 Evaluation . . . 163
9.3 Related Work . . . 164
9.4 Summary . . . 165

10 Personalized Web Search 167
10.1 Architecture . . . 168
10.1.1 Design-Time Framework . . . 168
10.1.2 Run-Time Framework . . . 171
10.1.3 Use Case Scenario . . . 172
10.2 Evaluation . . . 173
10.3 Related Work . . . 175
10.4 Summary . . . 176

11 Conclusions 177
11.1 Ongoing and Future Work . . . 179

IV Appendix 181

A Grammar of the ARRIA Rule Language 183

B Usability Test Results 193

C ARRIA Search Extension Usability Study 213

References 215

List of Figures

1.1 Adaptive vs. non-adaptive user interfaces 4
1.2 Research contributions 11
1.3 Structure of the thesis 15

2.1 Research fields 17
2.2 Aggregation of web log data 26
2.3 Using frequent item-sets for personalization 28
2.4 Rete discrimination network 33

3.1 Non-adaptive service list of the building apartment 43

4.1 The ARRIA Adaptation Framework 52
4.2 Hovering over several links 60
4.3 Direct user guidance 61
4.4 Research challenge 62

5.1 Modeling Cycle 66
5.2 ARRIA ontologies 68
5.3 Annotation of web page elements 72
5.4 Examples of transforming web log data 75

6.1 Event rule 94
6.2 Event operators 96
6.3 Action rule 102
6.4 Modify rule 104
6.5 Construction Wizard 105
6.6 Construction Wizard in action 106

7.1 Conceptual level architecture 115
7.2 Design level architecture 120
7.3 Hybrid Discrimination Network 125
7.4 Construction Wizard example 129
7.5 Event frequency 132
7.6 Object modifications 132

8.1 Dynamically inserted Construction Wizard 140
8.2 Finite state machine of the Evaluation Wizard 144
8.3 Screenshot of the welcome page 145
8.4 Screenshot of the personal data questionnaire 146
8.5 Screenshot explaining the evaluation procedure 147
8.6 Screenshot describing the first task 148
8.7 Screenshot of the static home page of Vöcklabruck 149
8.8 Screenshot of the questionnaire 150
8.9 Age distribution 151
8.10 Gender distribution 151
8.11 German as mother tongue distribution 151
8.12 Familiarity with the Internet distribution 151
8.13 User satisfaction 154
8.14 Easy to find forms 154
8.15 Preference to online portal 155

9.1 ARRIA approach for personalized web advertisement 162

10.1 Architecture of the ARRIA Search Extension 169
10.2 Abstract use case scenario 172
10.3 Concrete use case scenario 174

List of Tables

5.1 E-Government domain facets 71
5.2 Discovered association rules 80

8.1 Test tasks and their solutions 144
8.2 Z-test for the correctness hypothesis 152
8.3 Two-sample T-test for the duration hypothesis 153
8.4 Z-test for the help hypothesis 154
8.5 Z-test for the search hypothesis 154

B.1 Personal Data and Questionnaire 195
B.3 Summary 198
B.4 Task 1 200
B.5 Task 2 204
B.6 Task 3 208

C.1 Questions and scales 213
C.2 Question 1 214
C.3 Question 2 214
C.4 Question 3 214
C.5 Question 4 214
C.6 Question 5 214
C.7 Question 6 214
C.8 Question 7 214
C.9 Recorded user behavior 214

List of Acronyms

AH	Adaptive Hypermedia
AHAM	Adaptive Hypermedia Applications Model
AHS	Adaptive Hypermedia System
AI	Artificial Intelligence
Ajax	Asynchronous JavaScript and XML
ANTLR	ANother Tool for Language Recognition
AOOH	Adaptive Object Oriented Hypermedia
API	Application Programming Interface
ARFF	Attribute-Relation File Format
ARL	Adaptation Rule Language
ARRIA	Adaptive and Reactive Rich Internet Application
AWAC	Adaptive Web Applications Creator
CA	Condition Action
CAWE	Computer Aided Web Engineering
CEP	Complex Event Processing
CI	Collective Intelligence
CSS	Cascading Style Sheets
DAG	Directed Acyclic Graph
DHTML	Dynamic HTML
DL	Description Logic
DOM	Document Object Model
EA	Event Action
EBNF	Extended Backus-Naur Form
ECA	Event Condition Action
ESP	Event Stream Processing
FIFO	First In, First Out
FMC	Fundamental Modeling Concepts
FSM	Finite State Machine

GUI	Graphical User Interface
HCI	Human Computer Interaction
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IFrame	Inline Frame
IR	Information Retrieval
IT	Information Technology
JSON	JavaScript Object Notation
OWL	Web Ontology Language
POA	Performance Optimization Application
PRML	Personalization Rules Modeling Language
RDF	Resource Description Framework
RFC	Request for Comments
RIA	Rich Internet Application
RSS	Really Simple Syndication
RTT	Round-Trip Time
SaaS	Software as a Service
SOA	Service-Oriented Architecture
SQL	Structured Query Language
SWRL	Semantic Web Rule Language
SWW	Semantic Web Widget
TAM	Technical Architecture Modeling
UI	User Interface
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WAI-ARIA	Web Accessibility Initiative - Accessible Rich Interactive Applications
WEKA	Waikato Environment for Knowledge Analysis
WME	Working Memory Element
WUI	Web User Interface
WUM	Web Usage Mining

WWW	World Wide Web
W3C	World Wide Web Consortium
XML	Extensible Markup Language

Part I

Problem Statement

Chapter 1

Introduction

In this chapter we motivate our research. We point out research challenges, hypothesize potential solutions approaches and list our major research contributions. Furthermore, we describe the research methodology which guided our research, and, finally, we outline the structure of the thesis.

1.1 Motivation

Every Internet user is an individual with different information needs and preferences. Current web applications[1] meet this diversity by providing one-size-fits-all user interfaces (UI) which address only the skill set of an imaginary average user. This is depicted in Figure 1.1 a). The advantage of such web UIs is their ease of design and their low total costs of ownership as only one user interface has to be designed and maintained for all potential customers. On the other hand, the major drawback is their missing support of essential needs of most individual users as pointed out by Brusilovsky et al. (2000b); Bradley et al. (2000); Kushmerick et al. (2000).

Figure 1.1 uses SAP's standardized Technical Architecture Modeling (TAM)[2] (Gröne, 2008a,b) in order to illustrate the problematic nature of one-size-fits-all UIs of web applications. A scenario from the public sector domain exemplifies the

[1] A web application is a piece of software the user interface of which can be accessed via and rendered by a common web bowser.

[2] TAM is a subset of UML 2.0, including the Fundamental Modeling Concepts (FMC) block diagrams `http://www.fmc-modeling.org/download/notation_reference/Reference_Sheet-Block_Diagram.pdf`. According to Gröne (2008a), TAM assigns diagram types and elements to two levels of abstraction. First, the conceptual level is needed to describe the system structures and the concepts. Models on conceptual level can be used to communicate with all stakeholders. Second, the design level focuses on the implementation of the system, for example the description of the code structures. These models are mainly used within development.

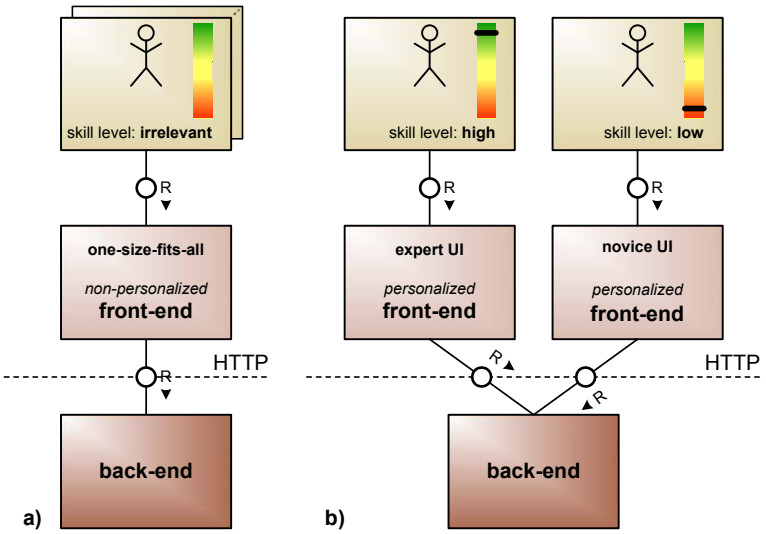

Figure 1.1: a) One-size-fits-all user interfaces. b) Personalized user interfaces according to skill level.

general problems: A typical service provided by an e-Government portal[3] is the submission of an application form related to a building project. Such a service is actually a complex process, subject to regulations that require the submission of different forms at different times, depending on the type of building project. As depicted in Figure 1.1 a) the challenge for inexperienced users starts here. With lack of background knowledge of the building regulations the users are confused and do not know which form to choose for their building project. Such users, unfamiliar with the specific online service, need guidance in order to prevent them from getting stuck in the portal shallows. On the other hand, for architects working daily with the virtual building authorities within an e-Government portal, any guidance would only hinder their smooth sailing. Therefore, there is, among other things, a need to cater for different skill levels. Moreover, the skill level may vary from service to service since for instance an expert in one service may be a novice in another. Thus, an architect, expert at building applications, may be a novice when it comes to submitting an application for marriage. Especially in the field of e-Government services, many of these will only rarely be used by any one user, so the majority of users will probably remain novices in their use.

[3]Web portals are a web sites that function as single point of access to a variety of information and services. An e-Government portal is a web site bringing these characteristics to citizens provided by their public authorities like for instance municipalities.

Due to their interactive nature web applications offer immediate direct[4] and indirect[5] user feedback, in contrast to traditional broadcast media like radio or television, which do not have a back-channel. User feedback can be recorded and stored in a model, called the user model. The information stored in a user model can be used to tailor web UIs specifically to the information needs of the individual user in order to foster the usability of the entire web application, as depicted in Figure 1.1 b). Such an adaptive web application can, for instance, hide or highlight links according to the background knowledge or experience of individual users inferred from their user models. For the architect in the example, adaptive web applications can provide ad-hoc user guidance, thus easing the discovery of the marriage certificate application form and related forms like, for instance, the birth certificate application form.

The adaptation of web applications based on contextual user models has been an object of investigation in the scientific domain of Adaptive Hypermedia (AH) for more than a decade, since the mid 90s (Brusilovsky, 1996). The theory of AH provides models for the adaptation of web applications to situations where people with different knowledge, backgrounds, experience, preferences and tasks are expected to interact with a web application. AH tries to improve the usability of hypermedia by providing methods and tools which bridge the gap between available information and information of interest.

Adaptive Hypermedia Systems (AHS) are conventionally implemented in the back-end. This means for the architect in the example that his/her web usage behavior can only be tracked, when a new page is requested. Based on such page requests the user model is built and the adaptation strategies are chosen. This pure server-side approach limits the possibilities of behavioral user tracking to HTTP requests only, which are actually only a subset of the overall user clickstream[6].

This strict server-side only approach of traditional UIs is in contrast to the new web programming paradigm of Rich Internet Applications (RIA). RIAs are web applications exhibiting characteristics of desktop applications. Driver et al. (2005) see RIAs as the next evolution of the web. RIAs provide richer and more responsive user interactions by gradually transforming the traditional thin client[7] approach of web browsers into a fat client[8] approach. In principle, RIAs

[4]Direct user feedback is obtained by explicitly asking web users for their opinions using, for instance, a questionnaire.

[5]Indirect user feedback is unobtrusively obtained from user interactions with the web application.

[6]A clickstream is composed of all interactions of a user with a web application (Mobasher et al., 2000).

[7]Traditionally, web browsers are thin clients, as they were designed as small computer programs dealing mainly with the client/server communication, while leaving the bulk of data processing on the server.

[8]Meanwhile, current web browsers are fat clients, as they allow web applications to perform the bulk of data processing directly on the client.

are state-full client applications residing inside a web browser asynchronously accessing from time to time a server-side service layer using standard Internet and web protocols.

Compared to traditional web applications relying on the web-page-paradigm[9], RIAs provide richer and on-the-fly user tracking and UI adaptation capabilities. With RIAs the range of user actions and thus the clickstream that can be tracked is extended beyond just HTTP requests. Such advanced user tracking capabilities comprise for instance any kind of mouse actions like scrolling, hovering or clicking. Thus, for instance, in the previous example the inexperience of an architect could be inferred from his/her browsing behavior of frequently hovering over hyperlinks in order to get an idea of how to navigate to the marriage form by studying the tooltips. Also by tracking the web usage behavior of an individual user directly on the client, client-side user modeling and adaptation is enabled without the need of an explicit, user-triggered request of a new web page. Thus, ad-hoc user guidance can be provided as a fading window to the architect, who is looking for the right form in an e-Government portal, although he/she did not explicitly issue an additional page request.

This directly leads to the major research question which is elaborated throughout the thesis: How to bridge the gap between the server-side adaptation approach of legacy AHSs and the client-side approach of modern RIAs in order to enhance the user experience of RIAs while retaining their responsive and rich UIs?

We addressed this research question by introducing Adaptive and Reactive Rich Internet Applications (ARRIA). ARRIAs leverage methods and techniques from interdisciplinary research fields in order to foster ad-hoc adaptivity and reactivity of RIAs. By adaptivity we mean the ad-hoc manipulation of the UI according to the current user's context and by reactivity we mean the ability of a web application to immediately respond to user interactions.

ARRIAs introduce a holistic Adaptation Framework covering the whole adaptation process, from obtaining meaningful adaptation patterns, through on-the-fly user modeling, to ad-hoc user interface adaptation. The Adaptation Framework is divided into the Design-Time and Run-Time Framework supporting the design and execution of ARRIAs, respectively. The Design-Time Framework focuses on gathering meaningful adaptation patters from the analysis of semantically enriched behavioral web usage data of a group of users by applying standard methods and tools adopted from the research field of Collective Intelligence (CI) and in particular from the research field of web usage mining (WUM). Compared to legacy WUM, we annotate the content and structure of web applications with ontologies[10] in order to mine declarative adaptation rules semantically. Assuming

[9] Conventional web applications can be viewed as a series of web pages which have do be explicitly and synchronously downloaded on behalf a user-issued distinct HTTP request.

[10] Ontologies are an explicit specification of a conceptualization (Gruber, 1993).

that, for instance, hyperlinks have been associated with concepts from an ontology, the system can now make a semantic interpretation of the user's web usage behavior. In order to convey adaptation patterns from the server to the client we encode them using the newly developed declarative Adaptation Rule Language (ARL). ARL is based on the event condition action (ECA) paradigm. The event-driven approach enables ARRIAs to adapt according to the chronological sequence of user interactions.

The ARRIA Run-Time Framework provides client-side, run-time services for on-the-fly user modeling and ad-hoc UI adaptation of web applications. Compared to legacy AHSs, the ARRIA Run-Time Framework advances the state of the art of AH by moving user tracking and UI adaptation from the server to the client, leveraging the fat client capabilities of modern web browsers, and providing a responsible execution environment for RIAs. We think the client-based approach offers advantages over the server-based approach by reducing client-server communication to a minimum and omitting latency; as there is no round-trip time (RTT) with the client-side approach, on time response to user actions in an ad-hoc fashion can be achieved.

The ARRIA Run-Time Framework is prototypically implemented using client-side JavaScript (ECMA International, 1999), as common scripting language for the development of dynamic web sites. We provide insights from a performance analysis of the prototypical run-time components, as well as from a comparative usability study showing the superiority of ARRIAs over legacy non-adaptive web applications with regard to their overall usability. Finally, the universality of the ARRIA approach is demonstrated by applying the whole framework, or parts of it, to several use cases form different domains.

1.2 Research Methodology

ARRIAs are information technology (IT) artifacts which enhance the usability of RIAs. Therefore, their design, creation and evaluation can be considered part of design science, which is defined by Hevner et al. (2004) as science that creates and evaluates IT artifacts intended to solve identified organizational problems. We based the ARRIAs research process on the design science research methodology for information systems research (Peffers et al., 2008), a commonly accepted scientific framework. The methodology consists of the following six activities:

Activity 1: Problem identification and motivation

Activity 2: Definition of the objectives of a solution

Activity 3: Design and development

Activity 4: Demonstration

Activity 5: Evaluation

Activity 6: Communication

The design science research methodology reflects the research process we followed over the past years in order to create what we now call ARRIA. We started with the analysis of already existing web and AH applications to identify problems of current solutions and in order to motivate our research (Activity 1). Based on this, and based on a comprehensive state of the art analysis, we derived the requirements and research challenges for our envisioned solution to bring adaptivity and reactivity to RIAs (Activity 2). After defining the objectives of our solution in the form of research challenges we designed and developed the concepts of ARRIAs (Activity 3). We went into iterative design and development phases, designed a holistic Adaptation Framework for the development and execution of ARRIAs, specified a declarative adaptation rule language tailored to the specific needs of client-side execution of adaptations, and, finally, implemented the ARRIA Adaptation Engine that can be hosted by common browsers. To accomplish Activity 4 we demonstrated the use of our artifacts by applying them not only to multiple use cases but also to additional application domains. We conducted performance evaluations as well as usability evaluations of the ARRIA approach in order to show that we indeed support a solution to the problem (Activity 5). Any new findings have been communicated to relevant scientific conferences and journals, completing Activity 6.

1.3 Research Challenges

In this section the research challenges which guided the research of ARRIAs are emphasized. The research challenges were obtained by studying the current state of the art in AH which is detailed in Chapter 2, by analyzing several use cases, and in particular, the e-Government use case described in Chapter 3, and, finally, by interviewing people from the SAP public sector group, the SAP design guild, and the SAP NetWeaver Portal group. The research challenges summarize yet unsolved research problems. Each challenge is justified in the course of this section by a short rationale.

Research Challenge 1

> *Research of a holistic Adaptation Framework in order to make RIAs adaptive and reactive*

Generic one-size-fits-all UIs of web applications raise usability issues for users not meeting the average skill set. Research on AH addressed this issue in the past

coming up with several solutions. So far research on AH only considered web applications based on the web page paradigm. But, as RIAs expose a richer set of user tracking and UI adaptation capabilities and demand more responsive UIs compared to legacy web applications, new AH techniques had to be researched for providing ad-hoc adaptiveness to RIAs. We hypothesize that a holistic Adaptation Framework considering both the design-time as well as the run-time aspects of providing ad-hoc adaptivity to RIAs will be able to bridge the gap between the server-side adaptation approach of legacy AHSs and the client-side approach of modern RIAs. Moreover, we hypothesize that such a holistic Adaptation Framework will retain the responsive and rich nature of the graphical user interface (GUI) of RIAs and at the same time will enhance the user experience of RIAs by providing adaptivity and reactivity.

Research Challenge 2

Extraction of meaningful adaptation patterns

In order to adapt the UIs of RIAs it is necessary to declare adaptation rules in advance. Adaptation rules can be learned either from usability or domain experts, or from recorded clickstream data describing the web usage behavior of a group of people, as for instance provided by web server access log files. WUM, as a subdiscipline of CI, is a promising approach for the acquisition of adaptation rules based on log files of web servers. WUM might benefit from the formal semantics of semantically annotated log files. Ontologies seem to be a natural fit for specifying domain concepts. Due to their formal nature additional knowledge might be inferred from explicitly stated facts. We hypothesize that by enhancing standard methods from the research field of CI with formal semantics from ontologies the acquisition of meaningful adaptation patterns from aggregated web sessions of a group of users[11], in order to formulate adaptation rules for the individual user, will be facilitated.

Research Challenge 3

Design of a client-side executable, declarative adaptation rule language

We presume that using the declarative programming paradigm eases the development of adaptive RIAs by describing only what adaptation should be performed and not how to compute it. Declarative rules are superior to imperative

[11] A user session is a sequence of pages visited by a user in one sitting, and can be reconstructed from web server access logs.

programming, as they express the adaptation logic without describing the control flow. A rule language allowing for the specification of complex events allows the rule engine to handle not only conditions testing the internal state of an application but also events signaling user actions. Furthermore, in order to get wide acceptance in the web application developer community, ARL needs to provide convenient access means to RIA functionality, like, e.g., the execution of client-side scripts. We hypothesize that a declarative adaptation rule language with convenient access to native RIA functionality that can be efficiently executed by RIAs on the client-side will enable user-centric, adaptive RIAs.

Research Challenge 4

Design and implementation of an efficient and scalable client-side adaptation engine

In order to provide punctual adaptivity to RIAs which take advantage of their rich UI capabilities, adaptation rules have to be evaluated directly on the client. Due to their rich and highly responsive user interfaces RIAs demand on time adaptivity that cannot be provided by conventional server-side AHSs, as they only provide adaptation after an explicit user-issued HTTP request. Moreover, ARRIAs shall be able to process events and detect temporal, logical and spatial interdependencies between them in order to track the complete user clickstream and not just HTTP requests. A complete clickstream is the basis for on-the-fly reactivity and adaptivity. A client-side event processor is able to construct a much more fined grained user model by additionally tracking, for instance, mouse movements or keystrokes, which in turn might trigger sophisticated adaptation rules. We hypothesize that an efficient and scalable adaptation engine capable of processing events and production rules directly on the client will bring ad-hoc adaptivity to RIAs.

1.4 Research Contributions

In this section our research contributions to the research challenges are expounded. The research contributions are mapped one-to-one to the research challenges, except the last, the fifth research contribution, which actually has no counterpart. This is because the fifth research contribution aims at the evaluation of the four preceding research contributions. All research contributions have been presented at respected scientific conferences, or published in scientific journals and books. Each research contribution presented below is briefly described, and the corresponding publications are referenced. Figure 1.2 puts the research contributions into the ARRIA context.

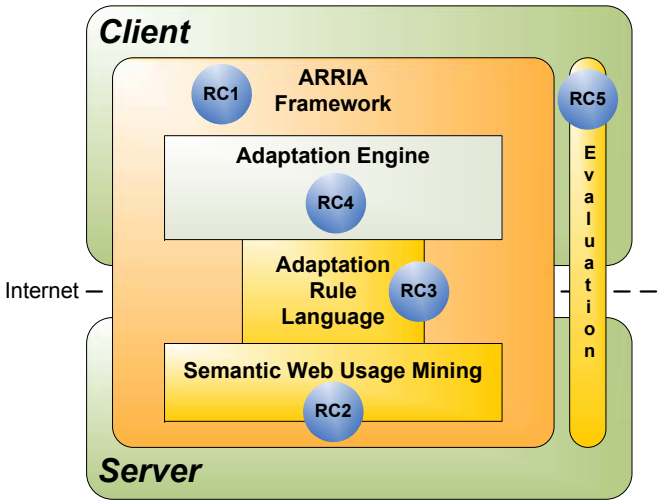

Figure 1.2: The ARRIA approach and its research contributions.

Research Contribution 1

A holistic Adaptation Framework for the design and development of ARRIAs (cf. RC1 in Figure 1.2)

Explanation. The ARRIA framework covers the whole adaptation cycle, from obtaining adaptation rules, through ad-hoc user modeling, to on-the-fly user interface adaptation. In addition to the provided methodology, we developed run-time and design-time tools facilitating the design and implementation of ARRIAs.

Publications.

- Schmidt, Stühmer, Dörflinger, Rahmani, Thomas, and Stojanovic (2010b): Adaptive Reactive Rich Internet Applications. *Annals of Information Systems: Web 2.0 & Semantic Web.*
- Schmidt, Dörflinger, Rahmani, Sahbi, Stojanovic, and Thomas (2008b): A User Interface Adaptation Architecture for Rich Internet Applications. 5^{th} *European Semantic Web Conference.*

Research Contribution 2

Usage of ontologies for the formal modeling of web applications and their application domains in order to automatically acquire adaptation rules from a group of users by mining behavioral data from semantically enriched user access logs with methods from the field of CI (cf. RC2 in Figure 1.2)

Explanation. With the help of ontologies, added to the web application in advance, we researched a semantic approach to WUM in order to find common web usage patterns. In fact, the most useful patterns can be directly modeled as adaptation patterns, which guide the user while interacting with the RIA.

Publications.

- Schmidt, Stojanovic, Stojanovic, and Thomas (2007): On Enriching Ajax with Semantics: The Web Personalization Use Case. *4^{th} European Semantic Web Conference.*
- Stojanovic, Schmidt, Stojanovic, and Thomas (2007c): Adaptive Portals Based on Combining Semantics and Ajax. *e-Challenges Conference 2007.*
- Rahmani, Thomas, Schmidt, and Stojanovic (2008): Using Semantic Web Usage Mining to Improve e-Government Websites. *7^{th} International EGOV Conference.*

Research Contribution 3

A lightweight adaptation rule language tailored to the needs of RIAs that can be executed directly on the client by most common browsers (cf. RC3 in Figure 1.2)

Explanation. Adaptation rules declaratively encode adaptation logic derived either from the web usage behavior of a group of users, i.e., the recorded interactions between users and RIAs, or from the knowledge of domain experts. The adaptation rule language's asset of being client-side executable leverages on-the-fly adaptivity of RIAs, as adaptation patterns can be executed with no latency, and the UI of a web application can be manipulated immediately.

Publications.

- Schmidt, Stühmer, and Stojanovic (2008e): From Business Rules to Application Rules in Rich Internet Applications. *Scalable Computing: Practice and Experience, Scientific International Journal for Parallel and Distributed Computing, Special Issue: The Web on the Move.*
- Schmidt, Anicic, and Stühmer (2008a): Event-Driven Reactivity: A Survey and Requirements Analysis. *3^{rd} International Workshop on Semantic Business Process Management in conjunction with the 5^{th} European Semantic Web Conference.*

Research Contribution 4

An efficient and scalable adaptation engine running directly on the client (cf. RC4 in Figure 1.2)

Explanation. As RIAs demand immediate user feedback, we decided to move the adaptation engine from the server to the client, thus, directly to the end user's desktop. To capture the manifold interactions between users and RIAs, and to provide on time and user-centric reactivity, we implemented a production rule system enhanced with event processing capabilities.

Publications.

- Schmidt and Stojanovic (2008): From Business Rules to Application Rules in Rich Internet Applications. *11th International Conference of Business Information Systems.*
- Schmidt, Stühmer, and Stojanovic (2008d): Blending Complex Event Processing with the Rete Algorithm. *1st International Workshop on Complex Event Processing for the Future Internet, collocated with the Future Internet Symposium.*
- Schmidt, Stühmer, and Stojanovic (2009b): Gaining Reactivity for Rich Internet Applications by Introducing Client-Side Complex Event Processing and Declarative Rules. *2009 AAAI Spring Symposium on Intelligent Event Processing.*

Research Contribution 5

Evaluation of the ARRIA approach by applying it to several use cases, and by conducting user-centric usability tests (cf. RC5 in Figure 1.2)

Explanation. We implemented the ARRIA approach in the e-Government portal of an Austrian local public administration. Afterwards we conducted a user-centric usability test in order to evaluate our hypotheses regarding the increased usability of the ARRIA enhanced city portal. Additionally, we verified that the ARRIA approach is applicable not only to the e-Government domain, but also to other use cases. Two which we examined are personalized web advertisement and personalized web search.

Publications.

- Schmidt, Stojanovic, Stojanovic, and Thomas (2010a): Personalization in e-Government: An Approach that Combines Semantics and Web 2.0. *Semantic Technologies for E-Government: A European Perspective.*
- Stühmer, Anicic, Sen, Ma, Schmidt, and Stojanovic (2009b): Lifting Events in RDF from Interactions with Annotated Web Pages. *8^{th} International Semantic Web Conference.*
- Stühmer, Anicic, Sen, Ma, Schmidt, and Stojanovic (2009a): Client-Side Event Processing for Personalized Web Advertisement. *8^{th} International Conference on Ontologies, DataBases, and Applications of Semantics.*
- Schmidt, Sarnow, and Stojanovic (2009a): Socially Filtered Web Search: An Approach Using Social Bookmarking Tags to Personalize Web Search. *24^{th} ACM Symposium on Applied Computing (SAC).*

1.5 Structure of the Thesis

The structure of the thesis follows the activities of the design science research methodology for information systems research. In Figure 1.3 the structure is represented. The activities of the design science research methodology are depicted at the upper right in each horizontal box, where a horizontal box represents a dedicated part of the thesis. The design science activity of communication has been carried out throughout the entire research and problem solving process, and is, therefore, depicted vertically.

The thesis is divided into three parts. In the first part the problem is identified; in the second part the solution of the problem is detailed; and in the third part the solution is evaluated. Below we detail the structure of each part of the thesis, briefly describe the content of each chapter, and associate the chapter to the research contributions, which are depicted in Figure 1.3 as RCx, where x is the number of the respective contribution.

Part I states the research problem of the thesis. It reflects the first and second activities of identifying the research problems and motivating the thesis as well as the definition of the objectives of our research. In addition to the introductory chapter the first part is made up of the two further chapters: Foundations and Use Case Analysis. In **Chapter 2** the technological foundations of ARRIAs are given. Inside is provided by discussing all involved research fields and technologies. The use case chapter, **Chapter 3**, describes the use cases considered for the research of ARRIAs. We analyzed more than one use case in order to incorporate universality into ARRIAs by design. In Chapter 3 we present use cases from several domains: the e-Government domain, the web advertisement domain and the web search domain.

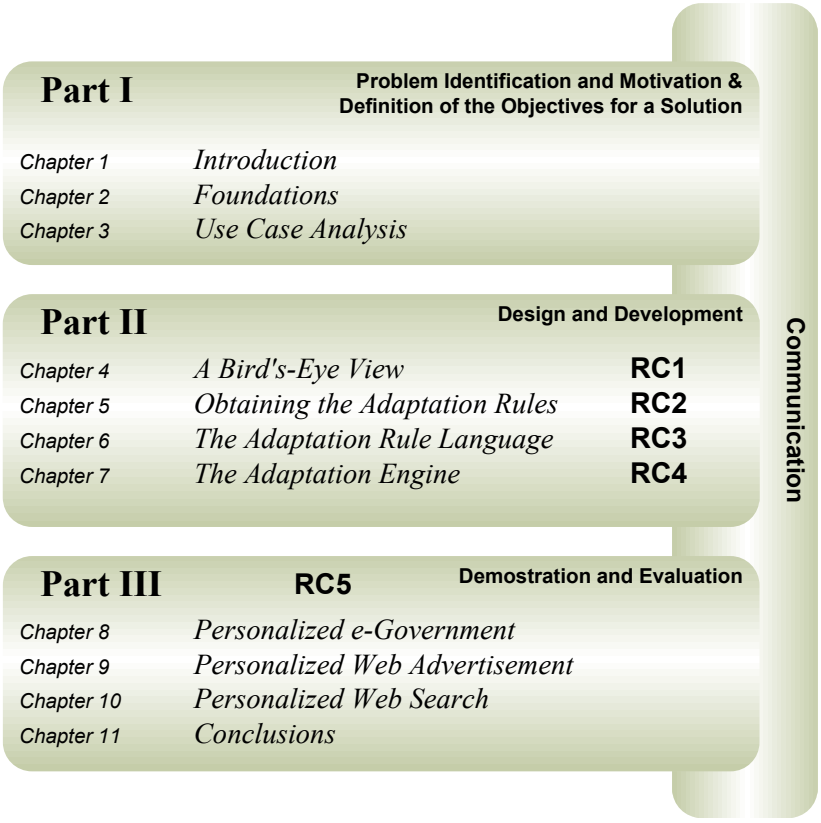

Figure 1.3: Structure of the thesis according to the design science research methodology.

Part II covers the third activity of the design science research methodology, namely, the design and development of the artifact. It is made up of four chapters: A Bird's-Eye View, Obtaining the Adaptation Rules, The Adaptation Rule Language and The Adaptation Engine. In each chapter one artifact and its research contribution to the respective research challenge is discussed (cf. RCx in Figure 1.3). Additionally, the ARRIA artifacts are compared to related scientific or commercial work for each research contribution. **Chapter 4** details our first research contribution: the Adaptation Framework of ARRIAs. The basic idea of lifting legacy, server-based AHS to the client is detailed and the overall mode of operation of ARRIAs is explained. In **Chapter 5** our second research contribution is detailed: the Modeling Cycle of the Adaptation Framework which enables to obtain the adaptation rules. After depicting the ARRIA ontologies, insights into our semantically enhanced web behavior mining approach are given. **Chapter 6** covers our third research contribution: the client-side executable Adaptation Rule Language. The formal syntax of the Adaptation Rule Language is presented in detail. **Chapter 7** explains the client-side adaptation engine of ARRIAs, which is our fourth research contribution. Its conceptual architecture is outlined and its algorithms are detailed.

In **Part III** of the thesis, we demonstrate and evaluate our research results which is in line with the fourth and fifth design science activities. This part is made up of four chapters: Personalized e-Government, Personalized Web Advertisement, Personalized Web Search, and, finally, Conclusions. The first three chapters constitute our fifth research contribution, as they evaluate the ARRIA approach in three different application domains. Related work is given only for Personalized Web Advertisement and Personalized Web Search, as related work for Personalized e-Government already is given throughout the previous part. **Chapter 8** evaluates the prototypical implementation of an ARRIA-based e-Government portal of an Austrian municipality which is presented throughout the previous part. A comparative usability test is described and its results are discussed. In **Chapter 9** the ARRIA approach is applied to the domain of web advertisement, in which ARRIAs are used to tailor web advertisements to the current working context of the user. **Chapter 10** comprises the application of ARRIAs to the web search domain. We demonstrate the use of selected parts of the ARRIA approach to personalize web search results. Finally, **Chapter 11** concludes our work of researching ARRIAs and gives insights into ongoing and future work.

Chapter 2

Foundations

In this chapter we lay the foundations for ARRIAs. We motivate and survey the six main contributing scientific research fields. The six scientific research fields, upon which ARRIAs are based, are depicted in Figure 2.1. The research fields are: RIAs, ontologies, CI, AH, production systems, and, finally, complex event processing (CEP). For each research field we motivate and introduce the technologies and algorithms relevant for ARRIAs.

ARRIAs are the objects of our research. In order to achieve ARRIAs we investigated RIAs with focus a on personalization. We enhanced RIAs by taking advantage of ontologies, CI, AH, production systems, and, CEP. We link ontologies to web applications in order to reveal domain knowledge. Based on these annotated web applications we track the user browsing behavior and analyze it by leveraging technologies from the research field of CI in order to gather meaningful personalization patterns. We use user modeling and adaptation strategies from the scientific domain of AH and lift them to the client-side in order to provide real-time, ad-hoc personalization of RIAs by eliminating client-server communication, while computing adaptations based on the user profile of the individual user. In

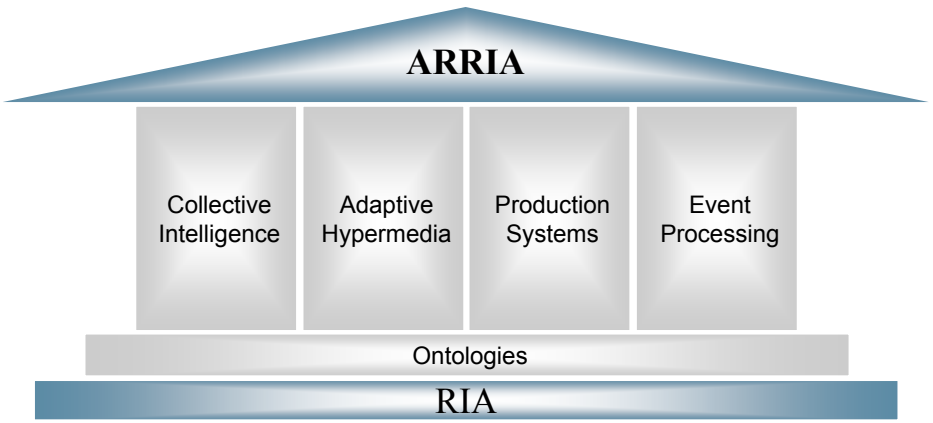

Figure 2.1: The constituting research fields.

order to collect and process behavioral data of an individual user on the client-side we decided on a declarative approach for representing the previously found personalization patterns. These personalization patterns are processed at runtime by a hybrid Adaptation Engine amalgamating and enhancing technologies from the research fields of production systems and event processing. Ontologies are not only used for collecting meaningful personalization patterns but they are also utilized by the Adaptation Engine for applying useful personalization strategies. Because the other research fields make use of RIAs and ontologies, RIAs and ontologies are depicted in Figure 2.1 as horizontal bars supporting the other research fields.

The chapter is structured as follows, first we explain the basic concepts of RIAs followed by insights into ontologies. Subsequently, the remaining four research fields are surveyed and motivated in the order depicted in Figure 2.1: CI, AH, production systems and event processing. Finally, this chapter is concluded by summarizing its content.

2.1 Rich Internet Applications

The increase of digital bandwidth and computing power of personal computers as well as the rise of the Web 2.0 came along with a new web programming paradigm: RIAs. RIAs are considered as the next evolution of the web by Driver et al. (2005). Now the time is right for the final breakthrough of RIAs, because of the broad bandwidth of today's Internet connections, as well as the availability of powerful and cheap personal computers. RIAs already enhance usability by their rich and responsive UIs, but they are not user-centric per se. That is why RIAs are the focus of our research and not traditional web applications adhering to the synchronous request/response paradigm.

In this section we discuss fundamental characteristics of RIAs and features that distinguish them from older non-rich web applications. As ARRIAs make heavy use of Asynchronous JavaScript and XML (Ajax), Ajax is presented in detail. The basics of JavaScript Object Notation (JSON) as a light-weight data interchange format are given subsequently.

2.1.1 Rich Clients

In the early 2000s the term RIA was introduced by Macromedia (Allaire, 2002). Duhl (2003) defines RIAs as software applications using common Internet technologies and asynchronous communication facilities while at the same time providing a rich UI that exhibits characteristics of desktop applications. RIAs run either in a web browser directly, in a web browser plug-in[1] or in a stand-alone

[1] Web browser plug-ins are sandboxes providing a controlled and supervised execution environment for RIAs to interact with the browser.

sandbox[2] directly on the client by connecting to the Internet asynchronously from time to time. Besides the rich UI, this asynchronous communication facility is the most distinguishing characteristic of RIAs compared to legacy web applications relying on the web-page-paradigm.

The term 'rich' indicates that RIAs are fat clients exhibiting rich desktop-like UI capabilities and the efficiency of performing computations directly on the client even without any server request at all. For instance, fading windows and drag-and-drop are rich UI capabilities which are adopted from desktop applications. Rich clients are also termed fat clients, thick clients, intelligent clients or simply clients. Rich or fat clients display rich application functionality to the user most of the time independently of a network. The counterpart to a fat client is a thin client, which is heavily dependent on the application running on a network server. Although, from time to time, even fat clients require a network connection, most of the time they perform without any network connection. A thin client on the other hand does as little processing as possible on the client-side but rather sends every input data that needs to be processed to the application server. The asynchronous communication facility of rich clients enables responsive GUIs by breaking the web page paradigm.

2.1.2 Ajax and JSON

As the RIA technology for ARRIA's Run-Time Framework we chose Ajax. Compared to other RIA technologies like Adobe Flex[3] or Microsoft Silverlight[4], Ajax is an out of the box and plug-in free RIA approach which allows the direct execution of RIAs without the need of an additional run-time plug-in. Ajax is not a proprietary RIA enabling technology but is commonly agreed upon by all major competing web browser vendors. There are many different vendors of Ajax client-side libraries and server-side development frameworks.

The term Ajax was coined by Garrett (2005) and is an acronym standing for asynchronous JavaScript and Extensible Markup Language (XML) (Bray et al., 2008). Ajax utilizes technologies which are already implemented by default in nearly all current state of the art web browsers. The following different, but related, web technologies are subsumed by Ajax: Dynamic HTML (DHTML), XML, and, finally, XMLHttpRequest. Rather then being a brand new technology Ajax combines these legacy web technologies under a uniform umbrella facilitating responsive web GUIs. The most notable property of Ajax is its asynchronous

[2]Sandboxes are security mechanisms separating and restricting the running of often untrusted programs without the need of installing them. They disallow, or, heavily restrict, the access to system resources like network, memory or hard disk. Sandboxes can be seen as a sort of virtualization.

[3]`http://www.adobe.com/products/flex`

[4]`http://silverlight.net`

communication facility. This communication approach allows web applications to reload only parts of a web page thus breaking the web-page-paradigm.

We now highlight three of the major technologies that Ajax relies upon, as ARRIAs make heavily use of them: the Document Object Model (DOM), JavaScript, and XMLHttpRequest. Le Hors et al. (2004) define DOM as a platform- and language-neutral object-oriented interface that allows programs and scripts to dynamically access and update the content, structure and style of documents. JavaScript is a dialect of the ECMAScript standard (ECMA International, 1999) used on the web as a scripting language for manipulating the DOM, and performing computations on objects within a host environment. It is object-oriented, weakly typed, prototype-oriented, and, therefore, instance-based, meaning functions are objects themselves. XMLHttpRequest is an internal browser object facilitating the asynchronous communication of Ajax web applications (van Kesteren, 2008). Via the XMLHttpRequest object RIAs can send Hypertext Transfer Protocol (HTTP) requests directly to a web server without having to wait synchronously for the response. The data interchange format can be JSON, XML or even plain text. The XMLHttpRequest object can be accessed via a browser scripting language like JavaScript.

Ajax web applications often use the JSON format as an alternative to the XML format for serializing and transmitting structured data. In the abstract and introduction to the request for comments (RFC) 4627 (Crockford, 2006b) JSON is described as a lightweight, text-based, language-independent data interchange format for the serialization of structured data that is derived from the object literals of JavaScript, as defined in the ECMAScript Programming Language Standard, Third Edition (ECMA International, 1999). Actually, JSON is a subset of JavaScript. JSON defines a small set of formatting rules for the portable representation of structured data and defines six primitive and structured types. The primitive types are strings, numbers, booleans and null, and the two structured types are objects and arrays. An excellent web resource introducing JSON is `http://www.json.org`.

2.2 Ontologies

ARRIAs use ontologies to model the domain of discourse. Gruber (1993) defines ontologies as an explicit specification of a conceptualization. This definition was slightly modified by Borst (1997) to a formal specification of a shared conceptualization, and, finally, Studer et al. (1998) came up with the well-established definition of ontologies as:

> An ontology is a formal, explicit specification of a shared conceptualization. Conceptualization refers to an abstract model of some phenomenon in the world by having identified the relevant concepts of that phenomenon. Explicit means that the type of concepts used,

> and the constraints on their use are explicitly defined. Formal refers to the fact that the ontology should be machine readable, which excludes natural language. Shared reflects the notion that an ontology captures consensual knowledge, that is, it is not private of some individual, but accepted by a group.

Staab and Studer (2004) provide detailed insights into ontology-related topics like reasoning and ontology engineering. Another excellent survey of semantic web technologies pointing out trends and research in ontology-based systems, is provided by Davies et al. (2006). The basics of ontologies, reasoning and semantic web technologies are summarized in Schmidt (2004).

Additionally, Flügge and Schmidt (2004, 2005) demonstrate how ontologies enable ad-hoc collaboration of web services, and Barnickel et al. (2006) elaborate how ontologies can be used for semantic web service composition in the e-Government domain. Furthermore, Schmidt (2007) argues for the application of ontologies in an enterprise context for establishing semantic enterprise service-oriented architectures (SOA).

There are strong reasons to build our approach upon semantic technologies. Ontologies enable semantic interpretation of user behavior in web applications, which enables meaningful, effective and context-aware adaptation. The building permission example from the introductory Section1.1 is elaborated next to show how RIAs and semantics together enable such context-aware adaptation. The architect wants to request a building permission. He/she goes to the appropriate e-Government web portal. On a navigation page in the web portal site the architect finds a list of hyperlinks to forms related to building permissions. In order to find the appropriate link the he/she hovers the mouse over the presented hyperlinks. The web application provides mouse-over help for these links. As the architect expects precisely this behavior, he/she places the mouse over the hyperlinks for a time to make the help appear. Assuming that the hyperlinks have been associated with concepts from an e-Government domain ontology, the system can now make a semantic interpretation of the architect's behavior. In this case, the conclusion would be that the user has a strong interest in the concepts associated with the hyperlinks for which he/she read the tooltips, and that he/she needs help choosing a form. In response to the determined user context, the system can offer specifically tailored help. The help can be tailored to the user by taking into account the concepts in which the user showed interest, deduced from his/her current navigation path and behavior.

Another reason is, that ontologies facilitate sharing knowledge between web applications, especially for those offering similar services (e.g. the online-services of two municipalities in one state are similar). For example, the best practices gathered in issuing building permits in one portal (e.g. inexperienced users need an additional explanation regarding the hyperlink *required documents*) can be easily transferred to other portals that implement the same regulations for issuing

building permits. This sharing is greatly facilitated by the fact that all of the terms used (e.g. additional explanation, hyperlinks, required documents etc.) are well defined. It is clear that the benefits for the users and the e-Government web applications are enormous, since the public administration can improve its performance at much less expense.

2.3 Adaptive Hypermedia

ARRIAs lift AH from the server to the client in order to provide on time adaptivity to RIAs. In this section we define the basic concepts of Adaptive Hypermedia (AH) as a computer science discipline closely linked to the research field of Human Computer Interaction (HCI).

Before we define the focus of AH, we start with the disambiguation of the terms *adaptability* and *adaptivity*. According to Goy et al. (2007) and Oppermann (1994) adaptable systems provide means which leverage user specific customization of the system. In contrast, adaptive systems preform the adaptation autonomously based on a user model. ARRIAs are adaptive systems.

According to Jameson (2007) the focus of AH is research on interactive systems that adapt to the current needs of the individual user. These interactive systems are called AHSs or user-adaptive systems (Schneider-Hufschmidt et al., 1993). Other historical labels for user-adaptive systems are: adaptive interfaces, user modeling systems, software/intelligent agents or personalization systems. We note that the terms adaptation and personalization are used alternately throughout the thesis as both denote the concept of user-centric adjustment of web applications.

AH promises to increase the usability of hypermedia by making it personalized (Brusilovsky, 1997). It protects users from the complexity of unrestricted hyperspace, and it reduces their cognitive overload (Kirsh, 2000). AH tries to improve the usability of hypermedia by providing methods and tools that bridge the gap between available information and information of interest. This gap is often referred to as the lost in hyperspace syndrome (Nielsen, 1995; Theng et al., 1996; Edwards and Hardman, 1999). According to Conklin (1987) the lost in hyperspace syndrome can be classified as either disorientation or cognitive overload. Disorientation refers to the loss of orientation in a web application e.g., users lost track of where they are in the web application, where they came form and where to go. Balasubramanian (1994) stated that cognitive overload refers to the difficulty of making decisions as to which links to follow, and which to abandon, given a large number of choices.

But there are also usability challenges for AH, as usability studies have shown that different users develop different navigation strategies, e.g., Fukuda and Bubb (2003), to accomplish the same goal. Fukuda and Bubb (2003) also observe that users tend to re-use a learned navigational strategy, even when it is not the

most efficient. Moreover, Tsandilas and Schraefel (2004) argue that AH affects landmarks on which users base their navigational and reading tasks. Landmarks may be an important part of a user's mental model, and when altered or deleted may result in disorientation. Vinson (1999) defines landmarks as distinctive environmental features functioning as reference points. In web pages such reference points are images, textual elements, graphics, layouts, etc.

Brusilovsky (1996) defines AHSs as:

> ...all hypertext and hypermedia systems which reflect some features of the user in the user model and apply this model to adapt various visible aspects of the system to the user. In other words, the system should satisfy three criteria: it should be a hypertext or hypermedia system, it should have a user model, and it should be able to adapt the hypermedia using this model.

Jameson (2007) approaches the definition of AHSs from a different angle. He defines AHSs from a process point of view. AHSs are composed of two processes, first, the process of user model acquisition, and, second, the process of user model application. Subsequently, we further elaborate on the three components of the definition of AHSs: hypermedia system, user model and adaptation.

2.3.1 Hypermedia System

A hypermedia system is a non-linear medium of information, interwining by hyperlinks different media like text, graphics, audio or video. The World Wide Web (WWW) itself is a classical example of a hypermedia system. As RIAs are nowadays an integral part of the WWW, we also consider ARRIAs as hypermedia systems, and, therefore, we use the AHS definition of Brusilovsky given in the previous subsection unchanged, as a working definition throughout the thesis.

2.3.2 User Models

Brusilovsky and Millán (2007) define user models as:

> ...a representation of information about an individual user that is essential for an adaptive system to provide the adaptation effect, i.e., to behave differently for different users.

A user model typically represents features of an individual user. Such features might be interests, knowledge, goals, tasks, background, or individual traits. What is modeled and stored in a user model is application specific. Only individual data needed for subsequent adaptation has to be collected.

The most promising user modeling approach for ARRIAs is that of modeling user interests. An individual user's interests were originally used in information

retrieval (IR) and filtering systems in order to profile users. This also gave this kind of user model its name: user profile. Throughout the theses we use the term user model and user profile synonymously as the underlying user model of ARRIAs is actually a user profile. According to Gauch et al. (2007), user profiles might be constructed based on explicitly or implicitly collected information. A hybrid approach is also feasible. Modifiable user profiles are considered dynamic, whereas profiles maintaining the same data over time are considered static. Dynamic user profiles can be further divided into short-term and long-term user models (Kim and Chan, 2003). Short-term user profiles represent the user's current interests, while long-term profiles represent lasting interests not subject to frequent changes.

The user model of ARRIAs is a short-term, dynamic profile based on implicitly collected information about the working context of the individual user (cf. Chapter 4). We define the user's working context, or, more simply the user's context, as the working situation that the individual user is currently engaged in. This situation consists of the current task or goal he/she wants to achieve in conjunction with the already accomplished tasks. This definition is broader than the commonly used definition, which defines context as the location of the individual user and the environmental dimensions including spatio-temporal aspects and physical conditions (Schmidt et al., 1999).

2.3.3 Adaptation Technologies

Adaptation technologies make use of the user model of individual users to adapt the web application according to user needs. The adaption space is quite limited in adaptive hypermedia. There are only a few features which can be altered (Brusilovsky, 1996). According to Brusilovsky (2007) AHSs adapt to preferences and goals of the current user by applying some of the following adaptation technologies: direct guidance, link ordering, link hiding, link annotation or link generation. ARRIAs support all adaptation technologies. In particular, direct guidance and link ordering are heavily used in our prototypical use case implementations.

Other adaptation technologies are the management of personalized views in information spaces (Waterworth, 1994; Thomas, 1995), the limitation of the navigation space (Mathe and Chen, 1996) or the presentation of the most relevant or similar links to follow (Armstrong et al., 1995; Kaplan et al., 1993).

2.4 Collective Intelligence

In order to find meaningful web usage behavior patterns from a group of users, and, based on this, useful adaptations for a single user, ARRIAs leverage research results from the scientific domain of CI. CI is an often used term in social and

computer science with different meanings. We use CI according to Segaran (2007) as the combination of behavior, preferences, or ideas of a group of people to create novel insights. This is exactly what is done by the ARRIAs, deriving new conclusions from independent contributors.

The general aim of CI is to casually collect data by tracking what people do, and, to perform calculations in an intelligent way to generate new information, potentially enhancing user experience. The analysis of data is performed by using methods, models and technologies from two subfields of artificial intelligence (AI), namely, machine learning and data mining.

In accord with Witten and Frank (2005) we consider machine learning as the ability of computers to acquire knowledge and to use it in order to change their behavior in a way that makes them perform better in the future. Closely related to machine learning is data mining, which is defined by Witten and Frank (2005) as a process of discovering useful patterns, automatically or semi-automatically in very large quantities of data.

The application of data mining techniques in order to discover patterns from the web is called web mining (Cooley et al., 1997). Web mining is divided into three subtypes differentiated by their analysis targets: WUM, web structure mining and web content mining.

Web content mining, sometimes also called web text mining, is the processing of text, image and other media types from the web to discover useful information. It is closely related to natural language processing and information retrieval. Web structure mining focuses on the analysis of the document structure of web applications as well as on the hyperlinks interweaving several web pages. The interesting web mining subtype for ARRIAs is WUM, as this discipline captures and models user behavioral patterns and the profiles of users interacting with a web site (Mobasher, 2004). A comprehensive survey of WUM and its related technologies is given in Schmidt and Thomas (2007); Stojanovic et al. (2007b).

2.4.1 Web Usage Mining

As an underlying approach for web personalization WUM has been proposed by Srivastava et al. (2000); Mobasher et al. (2000); Baraglia and Silvestri (2007). Mobasher (2004) states that the goal of WUM is to capture and model the behavioral patterns and profiles of users interacting with a web site.

The process of WUM consists of two phases (Mobasher, 2004): the data preparation and the pattern discovery phase. The data preparation phase explained by Cooley et al. (1999) consists of several sub-activities like filtering irrelevant web server log entries, robot detection, page view detection and session reconstruction. The heuristics on which session reconstruction is based are detailed by Spiliopoulou et al. (2003). The pattern discovery phase consists of mainly two sub-activities, namely, the actual usage mining and the analysis of the discovered patters. The actual usage mining activity applies data mining algorithms like

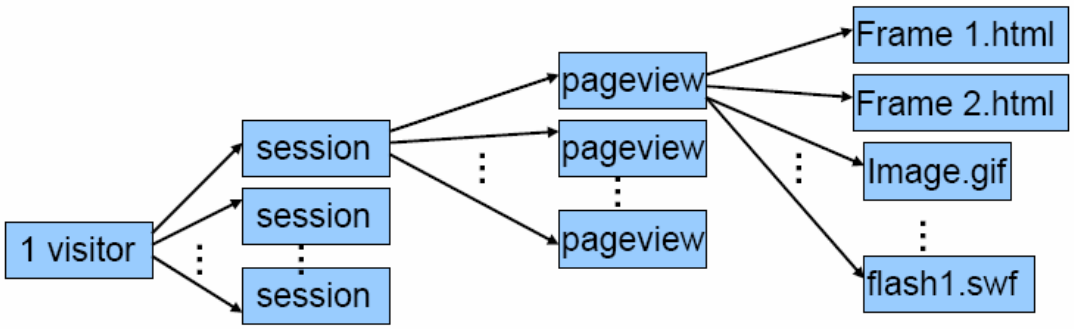

Figure 2.2: Aggregation of web log data.

association rule mining to the preprocessed historical data. By analyzing the discovered patterns aggregated usage profiles can be constructed.

The primary data sources used in web usage mining are the server log files, which include web server access logs and application server logs. The log data collected automatically by the web and application servers represents the rough navigational behavior of visitors. Depending on the goals of the analysis, this data needs to be transformed and aggregated at different levels of abstraction. In web usage mining, the most basic level of data abstraction is that of a pageview. Physically, a pageview is an aggregate representation of a collection of web objects contributing to the display on a user's browser resulting from a single user action. These web objects may include multiple pages (such as in a frame-based site), images, embedded components or scripts that populate portions of the displayed page. Conceptually, each pageview represents a specific user action, e.g., reading a news article, browsing the results of a search query, viewing a service page, executing a service, and so on. On the other hand, at the user level, the most basic level of behavioral abstraction is that of a server session (or simply a session). A session, which is also commonly referred to as a visit, is a sequence of pageviews by a single user during a single visit.

Figure 2.2 shows the aggregation of web log data (Stojanovic et al., 2007b). Page views are lowest level data for web usage mining. They represent series of requests for pages received by the web server that hosts a site. Hits are records of every GIF, JPEG, and HTML file requested by the user's browser. These hits can be aggregated into page views. Page views can be aggregated into sessions.

2.4.2 Semantic Web Usage Mining

Semantic WUM as a subcategory of web usage mining which enables tracking of user behavior at a conceptual level and can enhance collaborative filtering and personalization systems by providing concept-level recommendations in contrast to item-based or user-based recommendations. With ontologies the new item problem can be solved. The new item problem denotes the problem of rec-

ommender systems that they can only recommend what they have mined from historical data. They cannot recommend completely new items which were unknown at learning time. A comprehensive overview of semantic WUM approaches is given by Berendt et al. (2004); Stumme et al. (2006).

Semantic web usage mining involves the integration of domain knowledge into web usage mining (Dai and Mobasher, 2004). The use of semantic knowledge can lead to deeper interaction of the visitors or customers with the site. Integration of domain knowledge allows such systems to infer additional useful recommendations for users based on more fine grained characteristics of the objects being recommended, and provides the capability to explain and reason about user actions.

In general, it has been shown by (Dai and Mobasher, 2002) that it is useful to combine web usage mining with semantics in order to 'make sense' of observed frequent paths and the pages on these paths. The prerequisite for doing that is to develop the ontology describing the content of pages and subsequently to establish a mapping of individual pages to this conceptualization.

Domain Knowledge Integration

Domain knowledge can be integrated into the web mining process in many ways. This includes leveraging explicit domain ontologies or implicit domain semantics extracted from the content or the structure of documents or web sites. In general, however, this process may involve one or more of three critical activities: domain ontology acquisition, knowledge base construction and knowledge-enhanced pattern discovery.

Domain Ontology Acquisition. The process of acquiring, maintaining and enriching the domain ontologies is referred to as *Ontology Engineering*. For small web sites with only static web pages, it is feasible to construct a domain knowledge base manually or semi-manually. The outcome of this phase is a set of formally defined domain ontologies that precisely represent the web site. Good representation should provide machine understandability, the power of reasoning, and computation efficiency.

Knowledge Base Construction. While the first phase generates the formal representation of concepts and relations among them, the second phase, the construction of the knowledge base, can be viewed as building mappings between concepts or relations on the one hand, and objects on the web. The goal of this phase is to find the instances of the concepts and relations from the web site's domain, so that they can be exploited to perform further data mining tasks. Information extraction methods play an important role in this phase.

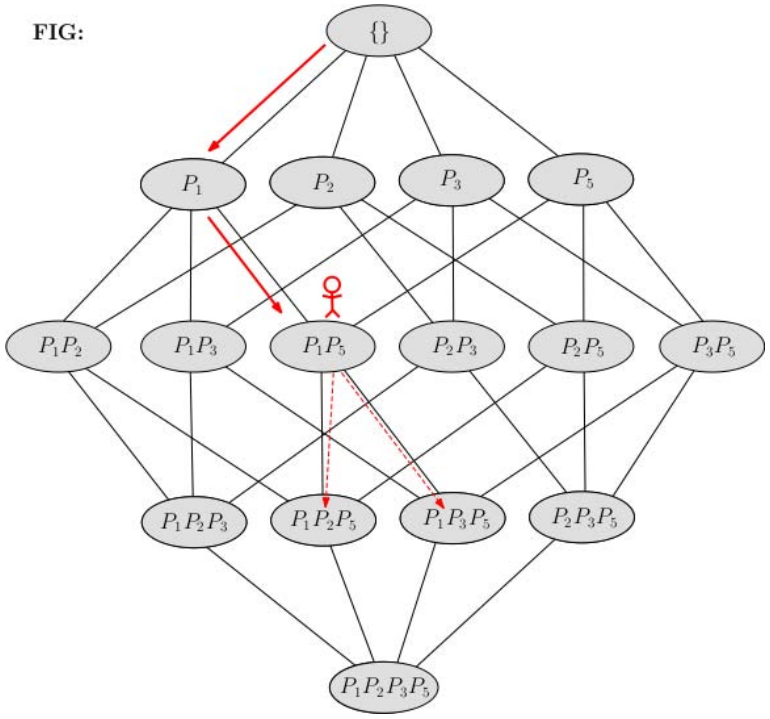

Figure 2.3: Using frequent item-sets for personalization.

Knowledge-Enhanced Pattern Discovery. Domain knowledge enables analysts to perform more powerful web data mining tasks. For example, semantic knowledge may help in interpreting, analyzing, and reasoning about usage patterns discovered in the mining phase.

2.4.3 Association Rules

On order to acquire adaptation patters, we require web usage mining that discovers relations between pages, or between concepts, if semantics are used. Therefore, the primary algorithms of interest are association rules. Association rules can be used for instance to personalize links. They specify that the occurrence of one set of pages in a user session implies the occurrence of another disjoint set of pages (or concepts). For example, the expression $\{P_1, P_5\} \Rightarrow \{P_2\}$ means that if the visitor went to pages P_1 and P_5, it is implied that he/she might also be interested in P_2. The right side of such rules is called the antecedent, the left side the consequent.

Association rules are derived from frequent item sets, that is, items (e.g. pages) that frequently occur together, where the meaning of frequently is given by the data analyst, who sets a threshold on the percentage of sessions in which the item set must occur before it is considered to be frequent. A frequent item-set graph taken form Stojanovic et al. (2007b) is shown in Figure 2.3.

Each node in the graph represents a frequent-item set, where the items are portal pages. Assuming that the analyst set the support threshold such that the frequent-items sets shown were found. As the user clicks through pages, the frequent item-sets he/she has visited change. Assuming that the user has clicked through page $P1$ and is now on page $P5$, as depicted. Since, as seen in the graph, $\{P1, P2, P5\}$ and $\{P1, P3, P5\}$, are also frequent-item sets, the system can suggest that the user should visit $P2$ or $P3$.

Note that this graphical representation of frequent-item sets is one way of representing the corresponding association rules such as: $\{P1, P5\} \Rightarrow \{P2\}$ and $\{P1, P5\} \Rightarrow \{P3\}$.

Support and Confidence

Usually, before starting the usage mining process, the analyst specifies values for two parameters that control the algorithms: support and confidence.

Support. The support $supp(X)$ of an itemset X of pageviews signifies how many sessions must support the itemset of the rule, that is, how many sessions must contain the itemset as a subset. Support is usually expressed as a percentage of the total sessions, e.g., 0.01 means 1% of the sessions support/contain the itemset.

Confidence. Confidence, on the other hand, says how much confidence can be placed in the rule, that is, what percentage of the supporting sessions confirm the rule, that is, contain the consequent as well as the antecedent. Confidence is expressed as a value between 0 and 1, where zero means no supporting sessions contain the consequent, and one means all supporting sessions do. The confidence of an association rule is defined as: $conf(X \Rightarrow Y) = supp(X \cup Y)/supp(X)$.

2.5 Production Systems

In order to personalize web applications on time ARRIAs utilize a client-side production rule system for the execution of adaptation rules. Production rule systems, or production systems for short, are problem-solving systems that execute declarative rules, also termed productions, based on the evaluation of one or more conditions (Waterman and Hayes-Roth, 1978; Brownston et al., 1985). In addition to productions, the knowledge base of an object-oriented production system includes objects, also termed facts. At run-time all objects are loaded into a memory. This memory is called *working memory*. The working memory constitutes the current state of the application. An object loaded into the working memory is called a *Working Memory Element (WME)*.

Algorithm 1: Match-Resolve-Act Cycle.

```
repeat
    Match conditions against WMEs
    Resolve conflicts if more then one rule is satisfied
    Execute actions
until No conditions match
```

We decided in favor of a production engine for executing adaptation rules mainly because of two reasons: declarative programming and efficiency of pattern matching. A production system can be programmed declaratively. This means that productions state what to do, and not how to do it. Production systems are also able to provide an explanation of how a solution was computed. Moreover, the declarative representation helps domain experts to inspect, understand and even modify application and adaptation logic. When combined with the declarative nature of ontologies the benefit of production systems for ARRIAs is even higher. For example, the hierarchical organization of e-Government services allows a domain expert to model adaptation rules on a more abstract level, i.e., covering more than one concrete service, e.g., building permission service, independently of the type of building such as house, office, etc. This reduces significantly the number of rules and makes maintenance of the system much easier. Additionally, the separation of logic and data in production systems can be advantageous, as separated application logic is easier to maintain.

The Match-Resolve-Act Cycle. Production systems are forward-chaining inference engines which constantly evaluate WMEs and productions by performing the *match-resolve-act cycle*. The match-resolve-act cycle is depicted in Algorithm 1. The match-resolve-act cycle is executed by the inference engine of the production systems until no further WMEs match the conditions. In the first phase of the cycle, the match phase, the conditions of the productions are matched against the working memory. Whenever the conditions of a rule are satisfied it is selected and put on the agenda. The agenda is the conflict resolution component of the inference engine and decides in the resolve phase in which order the rules from a set of rules are executed. There exist many conflict resolution strategies based on, for instance, rule priority. Another very basic resolution strategy is the first in, first out (FIFO) strategy, which orders the rules according to the time of their appearance on the agenda. The last phase of the match-resolve-act cycle is the act phase. In the act phase the actions of the rules are fired according to their order on the agenda. If different sets of WMEs satisfy the conditions of a rule the rule action is executed for all matching sets. After all actions of the selected rules are fired the match-resolve-act cycle starts anew until no conditions are matched.

The most prominent algorithm for production systems is Rete (Forgy, 1979). As we the Rete algorithm is utilized in our ARRIA approach for processing adaptations, it is detailed next. To the best of our knowledge there has been no client-side rule engine, so far, that can be used for adapting RIAs directly on the client-side.

2.5.1 The Rete Algorithm

Algorithms like Rete (Forgy, 1982), and its variations like Gator (Hanson and Hasan, 1993), Leaps (Batory, 1994) or Drools' Reteoo (Proctor, 2007) provide a very efficient and industry proven way of matching productions. Especially, when only some attributes of WMEs are altered, they are promising candidates for efficiently evaluating declarative adaptation rules. Rete is a natural fit for adaptation scenarios where forward chaining and inferencing is used to calculate new facts from existing facts.

Rete is an efficient pattern matching algorithm for productions. Productions are condition action (CA) rules consisting of one or more conditions and a set of actions. Actions are executed for the complete set of objects that match the conditions. Conditions perform tests on object attributes including type tests. The major advantage of the Rete algorithm over naive condition evaluation approaches is that by caching partial matches it avoids the complete re-evaluation of all objects each time they change.

Rete as a Directed Acyclic Graph. In the ARRIA approach the Rete network is constructed mostly at run-time by translating the production rules set. It is represented by using a directed acyclic graph (DAG) of in-memory objects. Thus, Rete can be described by using the formal notions of graph theory. A graph is a pair $G = (V, E)$ of sets such that $E \subseteq [V]^2$; thus, the elements of E are 2−element subsets of V (Diestel, 2005). The elements of V are the *vertices* or *Rete Nodes* of the Graph G. E is a set of pairs of vertices. These pairs are called *edges* (or *lines* or *arcs*). Directed Graphs are defined by Diestel (2005) as a pair (V, E) of disjoint sets of vertices and edges together with two mapping functions, init:$E \to V$ and ter:$E \to V$ assigning to every edge e an *initial vertex* init(e) and a *terminal vertex* ter(e). Finally, a directed graph not containing any cycles is called a DAG.

The Pattern Matching Network. The Rete pattern matching network, which in fact is a DAG, is divided into two parts: the alpha network and the beta network. The division of Rete into alpha and beta network is depicted in Figure 2.4. The alpha network is depicted above and the beta network on the bottom in Figure 2.4.

Alpha Network. The alpha network contains *Alpha Nodes* with 1-input edge. In the alpha network only attributes of one object are tested at the same time against constants and other attributes of the same object. This is in contrast to the beta network which performs tests between the attributes of two or more objects. The alpha network is a discrimination network. A discrimination network selects individual WMEs based on simple conditional tests. WMEs are fed into the alpha network via the unique entry point called the *Root Node*. The *Root Node* does not perform any tests on the objects but passes them directly to the *Type Nodes*. A *Type Node* tests the type of an object, that is, its class membership. Hence, all WMEs belonging to the same class typically traverse the same branch of nodes in the discrimination network. After checking for their types, WMEs are passed to *Selection Nodes*. *Selection Nodes* match WME attributes against constant values or against other attributes of the same WME. If a WME is successfully matched against the conditions represented by a node, it is passed to the next node and so one. As depicted in Figure 2.4, each branch of alpha nodes terminates at an *Alpha Memory*. *Alpha Memories* are sets of WMEs matching each condition in each node in the branch from the root to the *Alpha Memory*. Objects that fail to match at leased one condition are not stored.

Beta Network. The beta network contains 2-input *Join Nodes* or *Beta Nodes* which typically perform joins between WME lists stored in *Beta Memories* and individual WMEs stored in *Alpha Memories*. Each *Join Node* sends the result of the operation as a WME list to its corresponding *Beta Memory*, as depicted in Figure 2.4. The WME lists traverse the network by passing through further *Join Nodes*. In this process additional WMEs may be added to the lists and stored in *Beta Memories*. Every WME list stored in a *Beta Memory* represents a partial match for the conditions of a given production. When WME lists reach the end of a beta network branch a single production is completely matched. The lists are passed to the corresponding *Terminal Nodes*. A *Terminal Node* puts a new production instance on the *Agenda* for each WME list it receives.

After matching the production instances against the working memory the other two phases of the match-resolve-act cycle are executed as described above, objects are asserted or retracted from the working memory and the match-resolve-act cycle starts again.

2.6 Complex Event Processing

ARRIAs are RIAs which are adaptive, and, reactive. We define reactivity, based on the definition of Bry and Patranjan (2005), as the ability of an AHS to act in response to events in order to provide a measure of punctual adaptivity. In this section the foundations of event processing as a scientific discipline that researches reactivity are given. First, the basic concepts of event processing according to

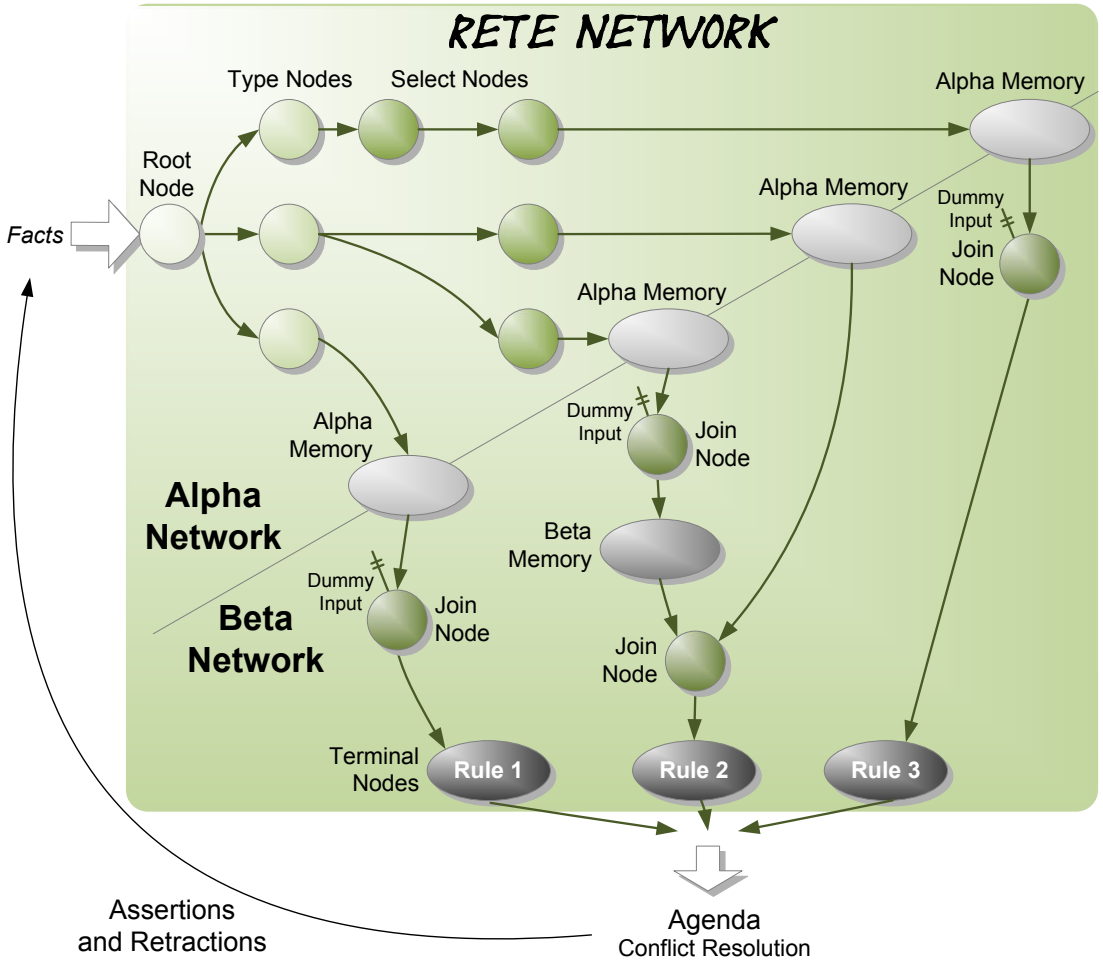

Figure 2.4: Rete discrimination network based on a figure by Charles Young (`http://upload.wikimedia.org/wikipedia/commons/9/92/Rete.JPG`).

Luckham and Schulte (2008) are described and then event specification languages and event processing algorithms are surveyed.

CEP is an ambitious new research area studying the methods, algorithms and algebras for processing complex events, which in turn can be used to define temporal, logical or spatial constraints on adaptation rules. We propose client-side CEP to enable the tracking of fine-grained user interactions, on-the-fly user modeling and the initiation of sophisticated UI adaptations.

CEP, since its advent in the 1980s, and still today, is mostly a feature of enterprise applications residing on the server-side (Dayal et al., 1988). However, depending on the sources of events, it is preferable to move the point of the detection as close as possible to the origin of events. In a web-browsing scenario we therefore propose client-side CEP. This means moving the task of CEP to the client, in this case the browser. In doing so, we reduce the latency which would otherwise be incurred by transmitting events over the Internet. Furthermore, the

volume of transmitted events is decreased because most events might not take part in any patterns, and, hence, relaying them would be counterproductive.

Use of CEP generally involves challenges such as having expressive operators, access to the necessary event sources, as well as efficient detection algorithms (Luckham, 2001). For RIAs there are specific further challenges such as the choice of an appropriate client-side programming language.

2.6.1 Basic Concepts of CEP

An event is anything that happens or is contemplated as happening (Luckham and Schulte, 2008). In computer science events are observable and typically asynchronous actions outside the scope of the program that handles the events, the event handler. Usually, the event handler is notified by software messages indicating the occurrences of events. In object-oriented software systems events and their payload are represented, encoded, carried and stored as objects. In fact event-driven computer systems process the representations of events namely the event objects rather than the events themselves. More then one event object may present the same real world event, called action. Several events of the same action represent different attributes, views or abstractions of that dedicated action. Once events are created they are immutable[5]. Event processing systems achieve the manipulation of immutable events by creating new events. The original events are not altered nor deleted. Events might also be virtual. Virtual events are not caused by actions in the physical world but appear to signal a real world event, e.g. simulated or modeled events. In this thesis we use the term event for event objects whereas we call real-world events actions. Sometimes, we also use the term event in order to describe an action. In such ambiguous cases the context of each use indicates which meaning is intended.

Luckham and Schulte (2008) insist that an event has to be strongly typed. The event type[6] determines the inner structure of an event instance that should at least consists of a unique identifier, the name of the event type, the time stamp of the event's creation and the event source. Besides these predefined attributes an event might have additional properties.

Closely linked with the notion of events is the notion of time. As events are objects in time they carry timestamps storing their creation time interval. Early work on events in computer science views events as having instantaneous occurrences, even if their creation or detection spans a time interval. As a simplification, the time point of the event detection is used. The consideration of only the time point of detection is termed detection-based semantics. But, detection-based semantics poses problems with nested sequences as pointed out in Galton

[5]The ARRIA event processing engine also treats events as immutable. But this is not mandatory. Different event engines might allow the manipulation of events after their creation.

[6]Other terms for event type are: event class, event definition or event schema.

and Augusto (2002). Therefore, Adaikkalavan and Chakravarthy (2006) argue for interval-based semantics which solves the problems of detection-based semantics.

A totally ordered sequence of events, of possibly many different types, is called an event stream. Usually events are ordered by time. Event sets exhibiting only a partial order are called event clouds. The difference between a stream and a cloud is its ordering. In a cloud there is no relationship that totally orders the events hence no assumptions about the arrival order of events can be made. An event cloud can be composed of many event types and streams.

Events are categorized into two basic types[7]: simple and complex events. Simple events are indivisible event occurrences in event streams or clouds. Usually, their detection is performed at run-time, directly after their occurrence but also a delayed detection based on historical event records is common practice. In general, simple events are distinguished by means of their different types and their occurrence times. In RIAs typical simple event types are: *mouseover* or *keypress*.

Complex or composite events are high-level artificial events composed of either simple events or other complex events. They are an abstraction of other events, derived or synthesized as a result of processing other simple or complex events. Coming back to our motivating example. The *LostInHyperspace* event indicates that a user, in our case an architect, does not find the web page he/she is looking for. The *LostInHyperspace* event is a complex event, and as such composed of simple events like the simple event *BackButtonPressed* which indicates that the user pressed the back button. When pressing the back button frequently without staying for a predefined time on the requested page, the *LostInHyperspace* event is raised. It consists of all the constituting simple events.

In order to detect simple events in streams or clouds, and to derive complex events, the behavior of an event processor[8] has to be specified. This is done by a high level declarative computer language called an event processing language or reaction language. The rules defined in the event processing language prescribe both event patterns, using an event specification language, and the actions to be taken whenever such a pattern is matched and the corresponding condition holds. Such rules are called reaction rules or ECA rules. On the other hand, event action (EA) rules are also reaction rules where the condition part is omitted. ECA rules generalize several methods to achieve reactive behavior, such as triggers and

[7]Luckham and Schulte (2008) categorize events into five types: simple, complex, derived, composite, and, finally, raw events. They define complex events as an abstraction of other events, derived events as results of applying a method to other events, composite events as the union of derived and complex events, simple events as events that are neither an abstraction nor a composition of other events, and, finally, raw events as records of a real-world event. For the sake of effectively working with these concepts we simplified them and defined a complex event in such a way that it also embraces the definitions of derived and composite events. The same holds for simple and raw events. A raw event is in our terminology also a simple event.

[8]Event processors also go by the following names: event processing components, event processing units, event processing systems, event processing engines, event mediators or event processing agents.

production rules, which had been in prior existence but treated separately. Many active database systems not only specify an event specification language but also a reaction rule language. The term ECA rule was first used in conjunction with the HiPAC active database (Dayal et al., 1988).

An event specification language, or, for short, event language, defines event patterns which are templates containing event descriptors, relational operators and variables. Event sets are matched against the template and the variables are replaced with values. Typical relational operators used in event languages are: `And`, `Or` or `Sequence`. Using the `Sequence` operator for instance stipulates that the matched events must occur in sequence.

Computing with events is called event processing. The operations performed on events include read, create, transform and delete. Event processing is divided into CEP and event stream processing (ESP). CEP and ESP are two research fields concerned with the computing of events. Traditionally, they research different problems, using different approaches.

Events may happen in a stream or in several streams, a cloud. The field of ESP is concerned with the extraction of events from a stream. Thus, ESP handles events that are totally ordered in time. Further emphasis of ESP is on efficiency for high throughput and low latency. Processing is done by analyzing the data of events, and, based on this analysis , appropriate events are selected. Appropriate events are events the ESP system has subscribed to because they are of a certain interest for the application.

CEP, on the other hand, is more focused on the detection of complex patterns of events. To detect these patterns CEP takes more time and memory than ESP. CEP is concerned with event clouds, that are only a partially temporally ordered. In comparison to ESP, CEP is more about higher level situational inferencing than about signal processing. However, because CEP and ESP nowadays adopt each other's approaches, the two paradigms become mingled, and, e.g., Bass (2007) declare them as one and the same scientific field.

Finally, in order to complete the basic event processing concepts we define an event-driven architecture as an architecture consisting of at least one event-driven component that reacts in response to the detection of events inside or outside the system boundaries.

2.6.2 Snoop, SnoopIB and Sentinel

ARRIAs are very much inspired by SnoopIB (Adaikkalavan and Chakravarthy, 2006). The definition of complex event patterns in our adaptation rule language is to a large extent based on the SnoopIB operators. Snoop (Chakravarthy et al., 1994) and its successor SnoopIB define an easy to use event algebra including operators like *And*, *Or* and *Sequence*. Early work on Snoop views events as having instantaneous occurrences. Interval-based semantics for Snoop is introduced by SnoopIB.

ARRIA's detection algorithms for complex events is a further development of the graph-based algorithm introduced in Sentinel (Chakravarthy, 1997). Sentinel is an active object-oriented database implementing complex event detection for the Snoop operators. Event detection follows a directed acyclic graph based approach. The graph is constructed from the event expressions. Complex expressions are represented by nodes with links to the nodes of their subexpressions, down to nodes of simple events. Event occurrences enter the bottom nodes and flow upwards through the graph, being joined into composite occurrences.

Compared to other CEP systems like Ode (Gehani et al., 1992b) based on automata, or SAMOS (Gatziu and Dittrich, 1994) based on Petri nets, Sentinel's graph-based approach does not represent, or even clarify the semantics of the event expressions. The semantics of the different Snoop operators is hidden in the implementation of each graph node. However, the semantics of Snoop is defined externally, using event histories, and describing the operators as mappings from simple event histories to complex event histories.

2.7 Summary

In this section we detailed and motivated the basic technologies ARRIAs rely on. The research fields from which these technologies come from are: RIAs, Ontologies, AH, CI, Production Systems and CEP. The different research fields are discussed separately each in its own section. We defined the basics concepts and surveyed the state of the art of each research filed.

Chapter 3

Use Case Analysis

In this chapter we detail several use cases which are excellent candidates for applying ARRIAs. To guarantee the universality of the ARRIA approach we analyzed use cases from three different application domains: e-Government, web advertisement and web search. The selected use cases originated from the FIT project[1] and from discussions with SAP internal groups like the SAP public sector group, the SAP design guild or the SAP NetWeaver Portal group. Out of the three investigated use cases, the e-Government use case scenarios influenced the ARRIA design and implementation most. The e-Government use case is used to illustrate the ARRIA approach throughout the whole thesis. In Chapters 8 to 10 the ARRIA approach is evaluated in the context of all three use cases.

In accord with the method of Hevner et al. (2004) we show that we address yet unsolved, important and real-world business problems. The use case analysis guided out before we started with our research of ARRIAs. Actually, we derived the research challenges presented in Section 1.4 based on the use case analysis and the study of related work.

The chapter is structured as follows: In Section 3.1 we analyze the personalization potential of municipal portals in the e-Government domain. Section 3.2 investigates which steps are necessary in order to deliver personalized web advertisement. In Section 3.3 we analyze our third use case which is about personalizing web search results. The chapter finishes with a summary.

3.1 Personalized E-Government

The European Commission (2007) defines e-Government as the use of information and communication technologies in public administrations in order to raise their

[1]FIT project: Fostering self-adaptive e-Government service improvement using semantic technologies — `http://www.fit-project.org`. The FIT project was co-funded by the European Commission under the *Information Society Technologies* Sixth Framework Program (2002-2006).

efficiency. For a survey of additional definitions of the term e-Government consider Palvia and Sharma (2007). According to the EU definition of e-Government, the long term goal for public bodies is to improve the image of governance by providing more efficient, easy to use and flexible service delivery, tailored to individual users (Webber et al., 2005). However, as documented by a number of studies (Rambøll Management, 2004; Nielsen and Loranger, 2006), this goal is not being achieved because poor usability plagues many of these web sites. Often citizens are not able to find the needed services or information; they fail because of difficult to use online services, or because of language understandability issues like jargon. These problems have arisen because the needs of public administrations and the technologies themselves were the rationale behind the development of public online services. The usability of the web applications was often overlooked. Many of the current online services were simply converted from offline paper-based services without considering the possible usability problems of web applications.

As the Internet, specifically the WWW has become one of the most important media in the past decade[2] (Morris, 1996), citizens expect public administrations to provide the same quality of e-Government services, which they are accustomed to when using online banking, flight booking or electronic shops. Increasingly, they also expect the types of personalization and user adaptation offered by such commercial services (Center for Democracy and Technology, and InfoDev, 2002). Typical users of e-Government services are all kinds of usually non-trained and unskilled citizens who do not regularly maintain an explicit application specific user profile.

The current norm for e-Government portals, which is to confront different citizens with a one-size-fits-all web interface, is not the optimum way to deliver public sector services because every person is an individual with different knowledge, abilities, skills and preferences. The conventional brick-and-mortar office has a more human face because the clerk can respond to different people in different ways. That is why people tend to use the conventional office rather than the e-Government services. To transfer some of the humanity of conventional services to e-Government portals, it is necessary to build adaptive portals for public services. Such user-adaptive portals will increase the usability, and, thus, the acceptance of e-Government, enabling administrations to achieve the, as yet, elusive efficiency gains and user satisfaction that are the primary goals of e-Government projects.

[2]Recently, in Europe the number of Internet users rose between 2000 and 2008 by around 366% from 105 million to 385 million with nearly constant population. In 2008 the percent of the European population who were Internet users was about 48%; nearly every second person was familiar with the Internet. In North America the Internet penetration was even higher and reached almost 74% (United Nations Population Division, 2006; Miniwatts Marketing Group, 2008).

Webber et al. (2005) analyzed seven major U.S. federal e-Government web sites and found that usability is in general poor, much poorer than commercial sites. Every site had significant design flaws that could only be removed by a user-centric redesign to identify and support the most important user goals. To solve the problems, they advocate persona-centric design. Briefly, a persona is a make-believe person, who is representative of a group of users with specific goals and interests. Personae are used to focus discussions and to guide decisions about the site design. Use of personae in design was popularized by Cooper (1999). Having recognized the usability problem, the U.S. government has a comprehensive web site dedicated to the topic of creating usable e-Government web sites. It offers, for example, guidelines for designing usable sites (Koyani et al., 2006). Member EU states have also developed usability guidelines. An example is the Austrian guideline to designing online forms (Wiesner and Pacnik, 2006).

The major usability challenge in e-Government portals is *Finding the Right Form*. This usability problem is a general problem in e-Government portals, as is also acknowledged by the SAP public sector group. The SAP public sector group also reports that public administrations are often overwhelmed by wrongly chosen forms or by unnecessary submissions of forms.

Wrongly submitted forms lead to increased costs for the public administration and to increased work and dissatisfaction for the citizens:

Increased costs for the public administration
: Every received form has to be checked for validity by the public administration. Not choosing the right form causes not only delays in the approval process, but, also generates costs as the clerk has to contact the citizen and to process the resubmitted form. Unnecessary submitted forms also increase the work load of clerks as every form as to be checked and answered.

Dissatisfaction for the citizens
: Without explicit user guidance it is hard to find the right form for an inexperienced client. This already causes confusion while searching for the right form. Clients feel unsure about the correctness of the form they have chosen. But, in the case of a wrongly submitted form they are even more dissatisfied after receiving the request to submit another request because they used the wrong online form.

Another major usability challenge of online services provided by the public sector is the perceived decreased quality in service provisioning because of non-personalized online services. This challenge seems to be contradictory to one of very important goals of e-Government portals: to enable the provision of easy and convenient access to public services for everybody. But citizens are accustomed to consulting clerks sitting in a brick-and-mortar environment who can be talked to either directly face to face or via telephone. Mardle (2007) argues that citizens

most often are not only looking for information but also for immediate help from public authorities.

3.1.1 Use Case Scenarios

The e-Government use case scenarios come from the FIT project. The goal of this project was to foster self-adaptive e-Government service improvements using semantic technologies. The FIT project acted as an incubator for the research of ARRIAs. The setup of the project gave us direct access to the knowledge and the domain expertise of the use case partners. The use case partners who contributed substantially to the ARRIA use case scenarios were the governmental officials of the Austrian city of Vöcklabruck[3]. In addition, the administration bodies committed to also provide web usage data in the form of web access log files for web usage behavior analysis. There has been a very close cooperation going on during the analysis, design, development, implementation and testing of the ARRIA approach. The web server was put up about 8 years ago and there has been little change since then. The IBM Lotus Domino web server[4], has been running reliably since 1999.

The official online city portal of Vöcklabruck is reachable via the web address `http://www.voecklabruck.at`, and is hosted and developed by the administration itself. The online portal serves as an Internet resource for accessing many kinds of information related to the town, for downloading forms related to public services like the application for a building permit or birth certificate, and, finally, also for the complete online handling of public services like, for instance, the submission of meter readings. We chose the city portal of Vöcklabruck for the use case analysis because of the administration's commitment to work with us in order to raise the usability of the portal.

The city portal of Vöcklabruck is a plain-old web application without any explicit nor implicit user guidance. Thus, it does not support the user in finding the right form, one of the major usability challenges of e-Government portals. As example the services of the building department are presented as simple list as depicted in Figure 3.1. Furthermore, the figure shows that the online services are categorized according to the responsible department. This categorization is not always intuitive as a citizen might not be aware which department is for instance responsible for ordering for instance meals on wheals for elderly people.

In order to identify an online service that is frequently used, complex, hard to use for a non-skilled citizen, and, thus, well suited for providing online adaptation and personalization we followed a three stage process. First we asked the e-

[3]Vöcklabruck is an Austrian town regional center with 13,000 inhabitants at the northern edge of the Salzkammergut region. The municipality of Vöcklabruck employees 216 persons all together, governmental work is done by 70 persons working on 60 computers in a state of the art network.

[4]`http://www.ibm.com/software/lotus/products/domino`

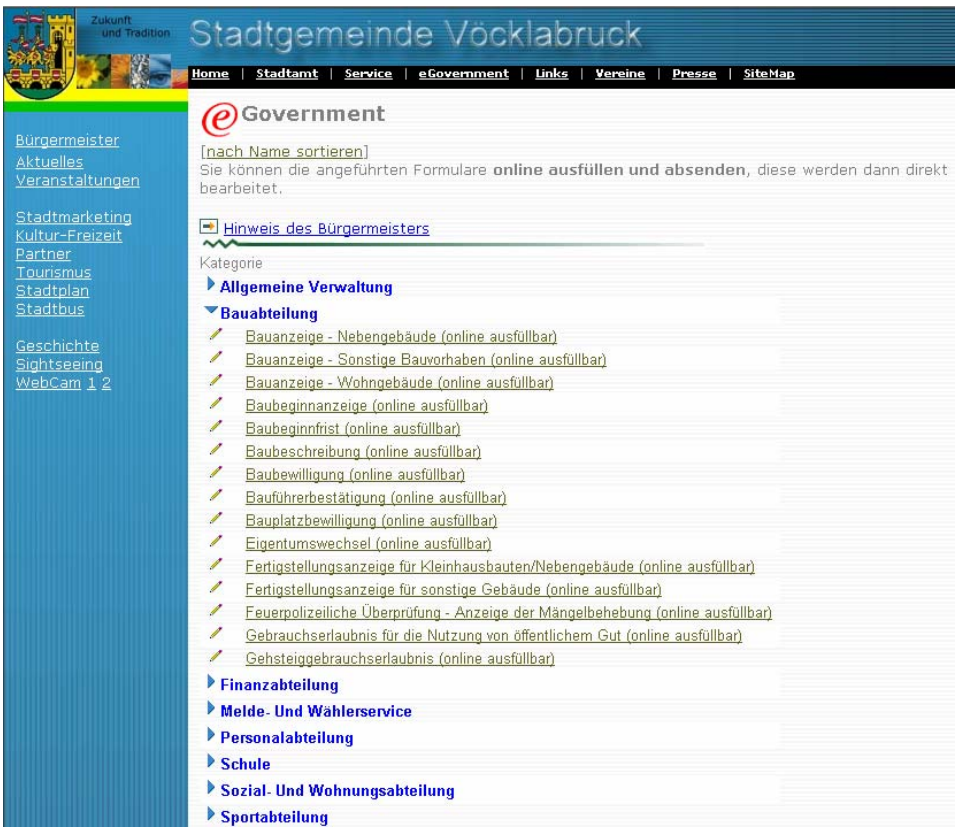

Figure 3.1: Simple list of services of the building apartment in the original non-adaptive web portal of Vöcklabruck.

Government domain experts of the city of Vöcklabruck to list all eligible online services, then we ranked them in a second step according to their frequency of use and their potential of being adapted. In a third step we requested a detailed description of the highest ranked process to serve as the basis to derive the requirements for the ARRIA approach. This process of identifying important and adaptable e-Government services was accompanied by an end user study to identify possible usability problems of the current portal of Vöcklabruck. The results of executing the processes are detailed in Feldkamp et al. (2006).

Public service experts of Vöcklabruck ranked two services highest amongst the 40 online services of Vöcklabruck, regarding their importance for the public administration and their potential for usability improvements by means of real-time adaptation of their web UI: building permission service and registry office service. In Austria, as well as in Germany, the building process is constrained by building law. The law regulates the administrative process for changing, building or demolishing any kind of building. The registry office offers services dealing with the civil status of citizens. Services offered are, for instance, the provision

of certificates of birth or of marriage. We selected both processes as examples for further analysis.

The Building Permission Process

Citizens have to apply for a building permit at the local department of housing and urban development, or over the Internet, via the e-Government portal of the responsible municipality, Vöcklabruck in our case. The trigger of this process is the request for a building permit or the submission of a notification to the authorities of an intention to build, modify or demolish an edifice. There exist several processes with distinct online forms for different kinds of construction. The Austrian legislator defines three types of building permissions: building permit[5], building notification[6] and construction projects for which notification of the authorities is unnecessary[7].

We analyzed the following three building permission process scenarios portrayed by Vöcklabruck's public sector experts. They are also used for conducting a comparative usability test in Section 8.2.

Demolition of a Double Garage. Mr. Schneider wants to demolish his part of a double garage in order to construct a new single garage. The two garages of the double garage are structurally connected along the property boundary. Mr. Schneider already knows that he has to apply for a building permit.

He decides to apply online for the permit. Thus, he connects to the Internet, browses to the city portal of his home town Vöcklabruck, clicks the public services link, navigates to the forms provided by the building department, and, finally, after some time of searching and browsing, still does not find a link for demolishing a building. Angry about that, he has to go directly to the Citizen's Advice Bureau. Due to the lack of user guidance he is not aware that he has to apply for a building permit instead of a demolition permit, as the demolition case is regarded as a construction of a building under Austrian and German law.

Building Law Rule 1: *IF* the destruction of structurally interconnected parts of a building is planned, *THEN* the building permit form has to be submitted.

Construction of a Swimming Pool. After a hot summer Mrs. Schulze decides to construct a swimming pool in her garden in Vöcklabruck. She plans to construct a swimming pool with a height of 1.40 meters and an area of 30 m^2. Mrs. Schulze thinks that all planned constructions have to be announced to the

[5] Official German term for building permit: *Baubewilligung*

[6] Official German term for building notification: *Bauanzeige*

[7] Official German term for not notifiable building project: *Nicht anzeigepflichtiges Bauvorhaben*

responsible public authorities. Thus, she visits the online portal of Vöcklabruck, and after navigating the web site of the public construction authority, she is confronted with two forms that seem redundant: the building permit form and the building notification form. Because of the absence of deeper knowledge of the construction law, she decides to fill in and submit the building permit.

With the submission of the building permit, she causes increased costs for the public construction authority, since according to Building Law Rule 2, the planned construction of a swimming pool with the dimensions of 1.40 meters in height and 30 m^2 in area may be done without.

Building Law Rule 2: *IF* the construction of a swimming pool with a depth of less than 1.50 meters and an area of less than 35 m^2 is planned, *THEN* the project is allowed without notification, meaning no form has to be submitted.

Construction of a Noise Barrier. After the railway line running alongside the estate of Mr. and Mrs. Schmidt was upgraded to an inter city express line, the married couple decided to build a noise barrier. The noise barrier shall have a height of three and a half meters, to protect their bedroom on the ground floor of their house from noise.

Not knowing the details of the building law, they applied online for a building permit in order to get their planned construction authorized. But, according to the building law, noise barriers higher than three meters require a building notification instead of a building permit. Thus, Mr. and Mrs. Schmidt caused additional costs to the public administration, as it has to refuse their wrongly chosen application for a building permit with the advise to apply for a building notification.

Building Law Rule 3: *IF* the height of a planned noise barrier exceeds 3 meters, *THEN* the building notification form has to be filled in.

Services of the Registry Office

The registry office issues certificates for several life events like birth, marriage, death, civil partnership or adoption. Some of these are likely to be filled in on the same occasion, but at the Vöcklabruck e-Government portal there is no support for finding associated forms. All available forms are displayed as a simple list like depicted in Figure 3.1. The user is not guided, e.g., after filling in one form as he/she does not know which form he/she should fill in next, as there is no predefined workflow implemented in the e-Government portal. Thus, for instance, the applicant for a marriage certificate has to provide as prerequisite a valid birth certificate.

Get Married. Please consider the case of Mr. Meyer, who wants to get married. After entering the city portal of Vöcklabruck, Mr. Meyer browses to the

comprehensive list of all available online forms with the aim of triggering the administrative process of issuing a marriage certificate. After some time spent searching, he finally finds the corresponding online form for applying for a marriage certificate. Without further guidance, he would leave the portal immediately after submitting the marriage certificate form. Thus, he would trigger an incomplete administrative process causing additional communication, leading to a time delay for Mr. Meyer, and additional administrative costs for the citizens advice bureau.

Registry Office Rule: *IF* an application for a marriage certificate has been made, *THEN* suggest that the applicant apply for a birth certificate.

3.2 Personalized Web Advertisement

The market for web advertisement is continuously growing[8]. Correspondingly, the number of approaches that can be used for realizing web advertisements are increasing. The main challenge in all these approaches is to personalize ads as much as possible. Google's AdSense[9], by far the most popular program for ad placement, automatically analyzes the content of web pages to dynamically determine which ads are the most relevant to each page. If a user is on a site reading about LCD televisions, AdSense shows him/her ads for retailers who sell them — without the web publishers or advertisers having to explicitly specify anything.

Focusing only on determining the content of a web page for contextualizing/personalizing ads leads to over-generalization of ads, which decreases the probability that the user will pay attention to an ad, even though it seems to be very relevant for the web page the user is visiting. In order to get the attention of a user, an ad should be as specific as possible to his/her current interest. For example, there is a big difference between the current interests of two users who are visiting the same web site about LCD televisions, if one is reading about technical characteristics of the device, and another is focusing on the text related to energy saving issues. Obviously, different ads should be delivered to these two users.

Consequently, differentiating between interests of the users who are visiting the same web page becomes the key issue for the successful advertisement. Indeed, the concept of *differentiation* is crucial for determining what is very relevant for a user, in the following sense: what distinguishes a user from other users (i.e. an average user) is a valid descriptor of his/her interests.

[8] IAB/PwC Internet Advertising Revenue Report 2008/Q2; `http://www.iab.net/media/file/IAB_PWC_2008_6m.pdf`.

[9] `https://www.google.com/adsense`

3.2.1 Use Case Scenario

Let us consider the case of Mr. Wagner, who browses a news web site to get informed. As Mr. Wagner is politically engaged he browses several articles about politics. Then he browses an article about the rain forest in which several hypothetical consequences of the climate change to the rain forest are discussed. To his surprise an online advertisement is provided linking to a bioenergy provider next to him. As Mr. Wagner is an environmentalist he is very interested in consuming bioenergy and clicks on the advertisement.

3.3 Personalized Web Search

Personalized web search has become increasingly important in the daily work of today's knowledge workers. Knowledge work comprises all kinds of acquiring, searching, analyzing and organizing of information (Drucker, 1959). The ever increasing information demands of knowledge workers lead to an information consumption paradox. On the one hand, more information means that the needed information might be out there, but, on the other hand, it becomes potentially more difficult to find the proper information. Non-personalized web search engines were successfully implemented by numerous vendors in the past, but they do not consider the current working context of information workers in order to narrow the information space. Only 20% to 45% of the common search results are relevant for the user (Machill et al., 2003).

Recent attempts to personalize search results primarily aim at the improvement of server-side ranking algorithms (Google Inc., 2007; Sun et al., 2005). But, naturally, a server-side user model can only reflect a fragment of an information worker's current working context. Therefore, scientific research on web search came up with client-side web search engines like (Shen et al., 2005; Teevan et al., 2005). The advantage of client-side engines lies in the comprehensive user model that can be built up, reflecting nearly all facets of user behavior, even across domain and application boundaries. Pitkow et al. (2002) state that the search can be personalized either by refining search queries or by re-ranking search results. Current approaches (Shen et al., 2005; Teevan et al., 2005) rely on the extraction from documents of additional terms describing the user context in order to refine search queries or re-rank search results. These systems are rather monolithic and hard to expand which restricts their usage. Furthermore, it is not proved that adaptive search results are preferred by users over static search results (Findlater and McGrenere, 2004), which is an issue, since they are used to working with a static result set that does not change depending on previously visited web pages.

3.3.1 Use Case Scenario

We consider the following use case scenario as a motivating example. The software developer Mrs. Weber is looking for guidelines on how to test some database specific code snippets. Currently, she is programming an IBM DB2 database, and, therefore, she browses to the corresponding web page *www.ibm.com/db2*. At the same time, in a different browser tab, she also reads the latest news at *www.hsqldb.org* as she is additionally interested in pure Java database solutions. We assume that these pages build the current working context of Mrs. Weber. As she is looking for testing guidelines, she issues the following query in Google[10]: *test*. The original search results returned by Google range from personality tests over web-based testing software to intelligence quotient tests, but what she is looking for are Java-based testing guidelines for the DB2 database management system.

3.4 Summary

In this section we analyzed several use cases, some of them taken from recent software projects addressing the implicit personalization of web interfaces. We analyzed the e-Government portal of Vöcklabruck, and we discussed that searching for the right form is a widespread problem in e-Government portals. Subsequently, we analyzed the use case of personalized web advertisement and personalized web search. By demonstrating the relevance of ARRIAs for several use cases we showed the universality of our approach.

[10] `http://www.google.com`

Part II

Design and Implementation

Chapter 4

A Bird's-Eye View

In this chapter a holistic view of our first research contribution, the ARRIA Adaptation Framework, is given. Important architectural issues are highlighted so that it becomes clear how they are linked together, and how they provide solutions to the first research challenge established in Section 1.3. Below we explain how the holistic ARRIA Adaptation Framework bridges the gap between the server-side adaptation approach of legacy AHSs and the client-side approach of modern RIAs. We published the ARRIA Adaptation Framework in the Journal *Annals of Information Systems: Web 2.0 and Semantic Web* (Schmidt et al., 2010b) and on the 5^{th} *European Semantic Web Conference* (Schmidt et al., 2008b).

This chapter is structured as follows: First we sketch the Adaptation Framework of ARRIAs. In order to provide a compact and yet complete picture of the Adaptation Framework of ARRIAs we only outline the processes of the Adaptation Framework and their phases in this chapter. The design and implementation of the Adaptation Framework is detailed in the following chapters. The continuation of the motivating example of Section 1.1 illustrates ARRIA's mode of operation. Subsequently, we explain how the research challenges of ARRIAs are met by the Adaptation Framework. Before we finally conclude with a brief summary of the chapter, we compare our approach to related work.

4.1 The Adaptation Framework

The ARRIA Adaptation Framework depicted in Figure 4.1 is a framework for managing the life cycle of user-centric, adaptive web applications. The Adaptation Framework supports the design, development, execution and maintenance of ARRIAs.

Design, development and maintenance are covered by the Design-Time Framework, as they are carried out offline. The iterative, offline process of designing, developing and maintaining ARRIAs is represented by the Modeling Cycle. At design-time ontologies are used to annotate web applications with domain knowl-

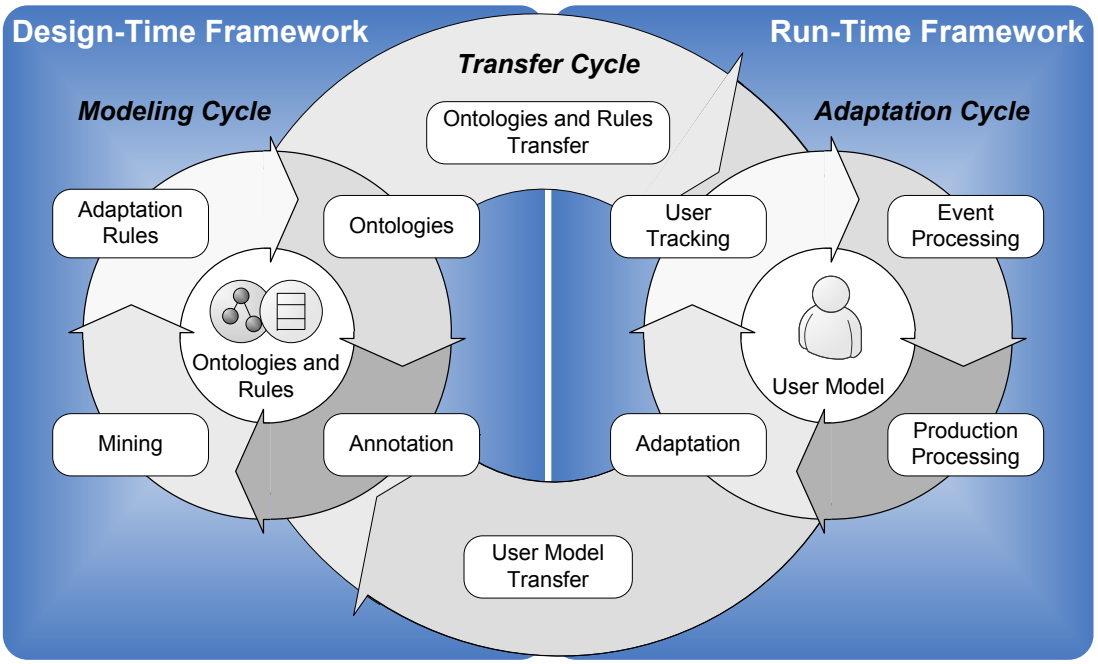

Figure 4.1: The ARRIA Adaptation Framework.

edge. Subsequently, these annotations are used in order to mine meaningful behavioral patterns from a group of users based on their logged web usage data. These patterns are then used to define meaningful and declarative adaptation rules.

The Run-Time Framework, which is our main research contribution, is responsible for executing these adaptation rules on the client-side at run-time. The involved iterative steps of executing adaptation rules in order to enable user-centric and adaptive web applications are represented by the Adaptation Cycle at the right in Figure 4.1. The web usage data of a single user is tracked online by the web application directly on the client in order to build up a user model of that particular user. This user model is continuously matched against the personalization patterns presented by the declarative adaptation rules. This is accomplished by a client-side Adaptation Engine relying on event and production processing. When the individual user model matches one, or many, personalization patterns the web application is adapted accordingly. After that the Adaptation Cycle starts over and continues user tracking. In fact, user tracking never stops; it continues during the evaluation of the adaptation rules.

The Transfer Cycle links the Modeling Cycle to the Adaptation Cycle. First, adaptation rules are transferred to the client-side at run-time in order to be executed by the Adaptation Engine on the client. Second, the web usage data and user model of the individual user can be transferred back to the Modeling

Cycle. In turn, these data can serve as further input for improving adaptation rules.

4.1.1 The Modeling Cycle

The Modeling Cycle comprises the design-time phases in charge of obtaining and constructing adaptation rules for the on-the-fly personalization of web applications. It consists of the four phases depicted on the left in Figure 4.1: ontology design, annotation, semantic WUM, and, finally, adaptation rules design. The first three phases correspond to domain ontology acquisition, knowledge base construction, and knowledge-enhanced pattern discovery as identified by Dai and Mobasher (2004). These three phases are enhanced by a forth phase dealing with the design of meaningful adaptation rules. We consider it as part of the Modeling Cycle, since the representation of personalization patterns is necessary for their run-time execution.

Ontologies

We use ontologies in order to gather meaningful personalization patterns from access log files by using semantic WUM. Dai and Mobasher (2004) argue that enhancing web usage data like access log files with metadata from ontologies can be advantageous to conventional data mining technologies, as it can enhance personalization systems by providing concept level recommendations.

We use the scientific method of faceted analysis for the design of ontologies for domains like, for instance, e-Government. According to Broughton and Slavic (2007) facet analysis provides an established scientific methodology for the conceptual organization of a subject field, and the structuring of an associated classification or controlled vocabulary.

Faceted classification allows the assignment of multiple facets to elements of a web application. Thus, for instance, in an e-Government portal a web object like a form could be annotated with the concept *Senior Citizens* from the facet *Target Group*, and, additionally, with the concept *Meals on Wheels* from the *Subject* facet. Rosati et al. (2004) demonstrates the advantage of faceted classification over a taxonomy hierarchy for the e-Government domain. We analyzed forms from a number of Austrian municipal portals in order to design a faceted E-Government Ontology for electronic forms.

At design-time ontologies can be used for gathering meaningful personalization patterns. As they provide a platform and programming language independent way to model class hierarchies, they can be used at run-time to infer subsumption hierarchies and the meaning of web elements. The ontologies ARRIAs rely on are detailed in Section 5.1.

Annotation

Creating up ontologies is only half the story. In order to take advantage of the formal semantics of the ontologies, the objects and structural elements of web applications must be linked to ontological concepts. This process is known as mapping or annotating and is usually supported by tools which simplify the explicit annotation of web applications, as explained by Stojanovic et al. (2007e). How we annotated a real web portal using the E-Government Ontology developed during the previous phase is described in Section 5.2.

Semantic Web Usage Mining

After annotating mainly the content of a web application, we need to relate these metadata to the already recorded non-semantic web usage behavior of individual users stored in web access log files. We developed a novel concept for relating ontologies to access log files of web servers. Semantically enhanced access log files can now be mined on a concept level in order to mine useful web usage patterns that can be used for adaptations. We achieved this by applying technologies of the discipline of semantic WUM (Dai and Mobasher, 2002; Rahmani et al., 2008). Our approach to semantic WUM is elaborated Section 5.3.

Adaptation Rules

After the discovery of web usage patterns, the discovered patterns have to be analyzed by a domain expert, for instance, an e-Government expert for the public sector domain, and, if available, by a usability expert. The experts have to judge, whether the patterns are useful or not. Patterns, which have been judged useful, are cast into adaptation rules using our client-side executable ARL. The design phase of adaptation rules is detailed in Section 5.4.

ARRIAs rely on the client-side execution of adaptation rules. Therefore, we specified a new ARL which can be easily processed by modern web browsers. ARL is a declarative ECA rule language (cf. Subsection 2.6.1) specified in extended Backus-Naur form (EBNF)[1] (cf. Appendix A). In Chapter 6 our new ARL is described.

4.1.2 The Transfer Cycle

At run-time the adaptation rules and the transformed ontologies are transferred to the client when the user requests an ARRIA-enhanced web application. This phase is depicted in Figure 4.1 as the ontologies and rules transfer phase. Optionally, the collected behavioral data and the inferred user model of an individual user can be transferred to the back-end. This phase is depicted in Figure 4.1

[1] EBNF is a notation used to formally express context-free grammars.

as the user model transfer phase. As the Transfer Cycle does not involve any modeling or processing this is the only place in the thesis where it is explained. In short, the Transfer Cycle simply transmits data by using standard Internet protocols like HTTP.

Ontologies and Rules Transfer

The personalization patterns in the form of adaptation rules written in ARL are simply transferred to the client in conjunction with the web application. But as web browsers have only limited processing capabilities in contrast to web servers, we decided to not directly deal with ontologies on the client. We propose to materialize ontologies beforehand on the server, and, consequently, transform them into an easy to process format. How we transform ontologies into a client-readable format is explained in Chapter 6.

User Model Transfer

The user model transfer phase is responsible for transferring back the user model created at run-time on the client to the back-end where all individual user models can be collected. Subsequently, the collected individual user models can be fed back into the Modeling Cycle in order to gain new adaptation rules or to fine-tune existing ones.

4.1.3 The Adaptation Cycle

The Adaptation Cycle, the core of the run-time part of the Adaptation Framework, is depicted rightmost in Figure 4.1. It aims at the on time adaptation of web applications to the current user based on predefined adaptation rules. In order to achieve this we designed and implemented an autonomous adaptation engine that operates directly on a user's client machine inside a web browser. We use Ajax as RIA technology as it is widely-used and does not require the installation of an additional browser plug-in (cf. Section 2.1). The Adaptation Cycle consists of four phases: user tracking, event processing, production processing, and, finally, adaptation.

User Tracking

Online user tracking is the prerequisite for on time personalization. By online user tracking comprehensive user data for the ad-hoc creation of short-term user profiles are gained. We decided to follow the approach of implicit user profiling in contrast to explicit user profiling in order to free users from the burden of providing and maintaining profiles by themselves.

While a user interacts with the web applications his/her clickstream is traced onsite on-the-fly. This is in contrast to current approaches of user tracking. State

of the art approaches usually log user behavior inside the web server providing the web application (cf. Section 2.3) for example in a web browser access log file. The design and implementation of the user tracking phase is explained in detail in Chapter 7.

Event Processing

In the event processing phase user interactions are captured by the Adaptation Engine. While interacting with a web application each user action issues one or many corresponding internal simple events like, for instance, *mouseover* or *key-press*. In order to derive more meaningful events we process simple events and compose complex events according to the guidelines specified by the adaptation rules. We derive the current work situation from the behavior of an individual user, that is, from his/her interaction history, the clickstream of the user within the web application. The user model that is built is a short-term dynamic profile based on implicitly collected information (cf. Section 2.3). Thus, users can be profiled and later on matched against the mined behavioral patterns collected in the semantic WUM phase of the Modeling Cycle. Matching is solely based on interactions with the web application. In the case of EA rules (cf. Subsection 2.6.1), that is, rules omitting the condition part, events can directly trigger adaptations.

We realized client-side complex event processing by utilizing a modified graph based approach based on Sentinel and SnoopIB (cf. Subsection 2.6.1). The design and implementation are detailed and evaluated in Chapter 7.

Production Processing

Often events trigger UI adaptations only under certain conditions. Conditions check attributes of the user profile or the application state. Typically, production rules are CA rules and their actions are fired, if the state of the application, which is called working memory, changes. Both the event part and the condition part of an ECA rule may consist of complex patterns which must be fulfilled for the rule to fire (cf. Subsection 2.6.1). The event pattern must be detected from one or more event sources and the condition pattern must be matched from among business objects (facts) pertaining to the application state. Business objects are objects that encapsulate real world data and business behavior associated with the entities that they represent (Heidasch, July/August 2007).

In Chapter 7 we show how we apply Rete as an efficient pattern matching algorithm for client-side production rule processing. Furthermore we detail how we combined complex event processing and Rete in a single holistic algorithm. Moreover, we provide an performance analysis of our prototypical implementation in Section 7.4.

Different Roles of Event Processing and Production Processing. Conceptually, events and non-temporal facts exhibit important differences. First, it must be noted that they serve different purposes. Event detection is concerned with transient, temporal data, i.e. events. The production processing, on the other hand, is concerned with persistent data, representing the system state, i.e. business objects. The two types of data are to be separated in order to avoid making unnecessary events persistent, and thereby imposing a storage burden on an web application.

Events and facts carry semantically different information. Therefore, we propose to distinguish temporal events from non-temporal facts in ECA rules engines.

- Events are transient and are consumed after they are delivered to all consumers. Facts, on the other hand, are persistent and must explicitly be deleted.
- Events are usually immutable, whereas facts may be changed.
- Facts are evaluated by an ECA rules engine using a pull strategy. In contrast events are usually pushed via event streams into the engine.

As a thought experiment, let us consider an ECA rules engine as a finite state machine (FSM), where events represent the transitions (Etzion, 2008). After the detection of a simple or complex event, the appropriate transition is taken. The event may be discarded afterwards. The facts, e.g., the attributes of the business objects, on the other hand, represent the state of an application. ECA rules engines have to ensure that the transition can only be taken if the condition and, thus, a specific application state holds. The involved facts remain valid and are retained. They can only be removed explicitly in a state change by calling the appropriate removal command. For the implementation of an ECA rules engine, on the other hand, there is no strict obligation to keep events and facts separated. However, as events are transient, the engine has to provide an efficient means to discard them after they are not used any further.

Another analogy from Bry and Eckert (2006) compares facts to written text. Text may be altered or deleted, it does not expire by itself and text is available for anyone to retrieve it. Events, on the other hand, are like the spoken word. They are available only to the people who listen at the right time, and are immutable.

Adaptation

During the adaptation phase the actual UI personalization takes place. When an adaptation rule fires, the corresponding actions are executed and the RIA adapts itself directly on the client-side without any further server requests. In this last phase of the adaption cycle the actions selected in the previous phases are executed, thus, achieving adaptation of the web application based on the profile of the individual user.

The adaptation phase corresponds to the agenda evaluation phase of modern rule engines. Single user actions can result in multiple eligible, but, potentially, conflicting rules. We implemented the simple FIFO conflict resolution strategy. Details can be found in Chapter 7.

4.2 Walkthrough Example

To illustrate the application of the ARRIA Adaptation Framework we use a variant of the motivating example from Section 1.1. We consider the already married couple Mr. and Mrs. Schmidt who want to construct a noise barrier (cf. the third building law rule in Subsection 3.1.1). Additionally, we consider the design-time process of adding adaptivity to an already existing e-Government portal.

4.2.1 Design-Time Walkthrough

Mrs. Lehmann is a public servant working for the Austrian municipality of Vöcklabruck. Her task is the electronic provision of e-Government services like the services of the registry office or construction office. In order to provide a user centric e-Government portal she decides to apply the ARRIA Adaptation Framework. According to the methodology of the Modeling Cycle of the Design-Time Framework she annotates the already existing web portal with concepts from ontologies especially designed for Austrian and German e-Government. Afterwards, during the mining phase, she mines the semantically enhanced web access log files of the e-Government portal in order to acquire common web usage patterns. Mrs. Lehmann is able to interpret the discovered patterns because of their semantic annotations. One out of many patterns that also have been confirmed by e-Government experts is the registry office rule stating that whenever a user submits a marriage certificate the system should also guide the user to the birth certificate form. This adaptation rule is illustrated in Subsection 3.1.1. In a last step she enlists the aid of Mr. Müller, the administrator of the e-Government portal, who encodes this rule in ARL and embeds the ARRIA Adaptation Engine in the portal. With the ARRIA system plugged into the city portal of Vöcklabruck incomplete administrative processes are prevented by design.

4.2.2 Run-Time Walkthrough

Mr. Schmidt enters the e-Government portal of the municipality he is living in. It is the municipality of Vöcklabruck for which Mrs. Lehmann is also working for. Mr. Schmidt's goal is to register online for his building project of constructing a noise barrier. As he is not an expert in Austrian building law, he does not know which form to fill in. Additionally, there might be more than one form to fill in.

In order to find the right form he browses to the web page of the building department. As Mr. Schmidt does not know immediately which link to follow he hovers over the displayed unordered list of hyperlinks pointing to several online forms (cf. Figure 4.2). Each hovering causes a dedicated mouse event which is caught by the Adaptation Engine's event processing component. At the same time the Adaptation Engine causes the portal to display a tooltip showing a short description of the link's destination. All links have been annotated by Mrs. Lehman during design-time with concepts from ARRIA's E-Government Ontology. The links over which Mr. Schmidt hovers are all annotated with subconcepts of the *Construction* facet. By evaluating the adaptation rules and exploiting the subsumption hierarchy of the E-Government Ontology the Adaptation Engine infers that Mr. Schmidt is looking for a link to an online form related to a building project. This conclusion becomes part of Mr. Schmidt's user profile. As a result the Adaptation Engine displays a dialog asking Mr. Schmidt to start the *Building Wizard* (cf. Figure 4.3). The Building Wizard is a software assistant to guide Mr. Schmidt directly to the building form he is looking for by asking questions which narrow the problem space (cf. Section 8.1). In summary, while browsing Vöcklabruck's e-Government portal, each of Mr. Schmidt's actions is tracked by the ARRIA Adaptation Engine during the Adaptation Cycle by leveraging the Run-Time Framework of ARRIAs.

4.3 Overcoming the Research Challenges

Based on the research challenges (cf. Section 1.3), we designed the concepts and developed a prototypical implementation of ARRIAs and the corresponding Adaptation Framework. In this section we outline how the proposed solution meets the research challenges.

Research Challenge 1

> *Research of a holistic Adaptation Framework in order to make RIAs adaptive and reactive*

The basic idea of ARRIAs, which is depicted in Figure 4.4, can be stated in one sentence: In order to enhance the user experience of RIAs we relocate the server-side run-time of traditional AHSs to the client. Performing user modeling and web application adaptation directly on the client is facilitated by leveraging and enhancing cutting-edge technologies like ontologies, and methods and tools from the research fields of CI, AH, production systems and event processing (cf. Figure 2.1). The continually enhanced processing power of modern desktop computers and the broad bandwidth of contemporary Internet connections established the basis for the client-side, dynamic and implicit personalization of web applications.

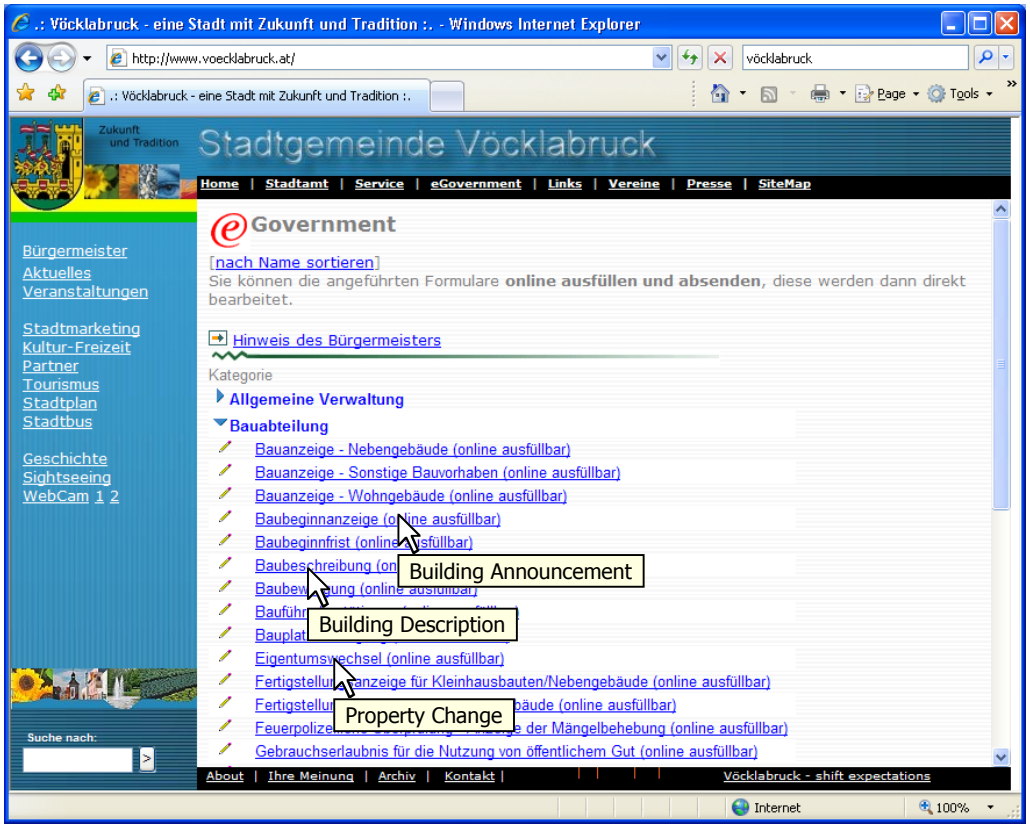

Figure 4.2: Mr. Schmidt hovers over several links to online forms.

The basic idea of ARRIAs is a direct outcome of our research towards overcoming the *first* research challenge. By adding the process of adapting a web application to an individual user's current context based on his/her recognized interests, we directly enhanced the user experience of RIAs. Client-side user-tracking and adaptation processing offer accurate, but unobtrusive user guidance while retaining the responsiveness and richness of RIAs. The superiority of AR-RIAs compared to non-adaptive, non-reactive web applications is substantiated by the results of a comparative usability test described in Section 8.2.

Research Challenge 2

Extraction of meaningful adaptation patterns

The *second* research challenge demands methods and tools for gathering meaningful adaptation rules. This challenge is addressed by the Modeling Cycle of the Design-Time Framework. During the ontology design phase mainly domain specific ontologies are designed which are used in the subsequent phase for the annotation of already existing web applications. Additionally, we developed supporting tools for easing the annotation of web applications. For the third phase,

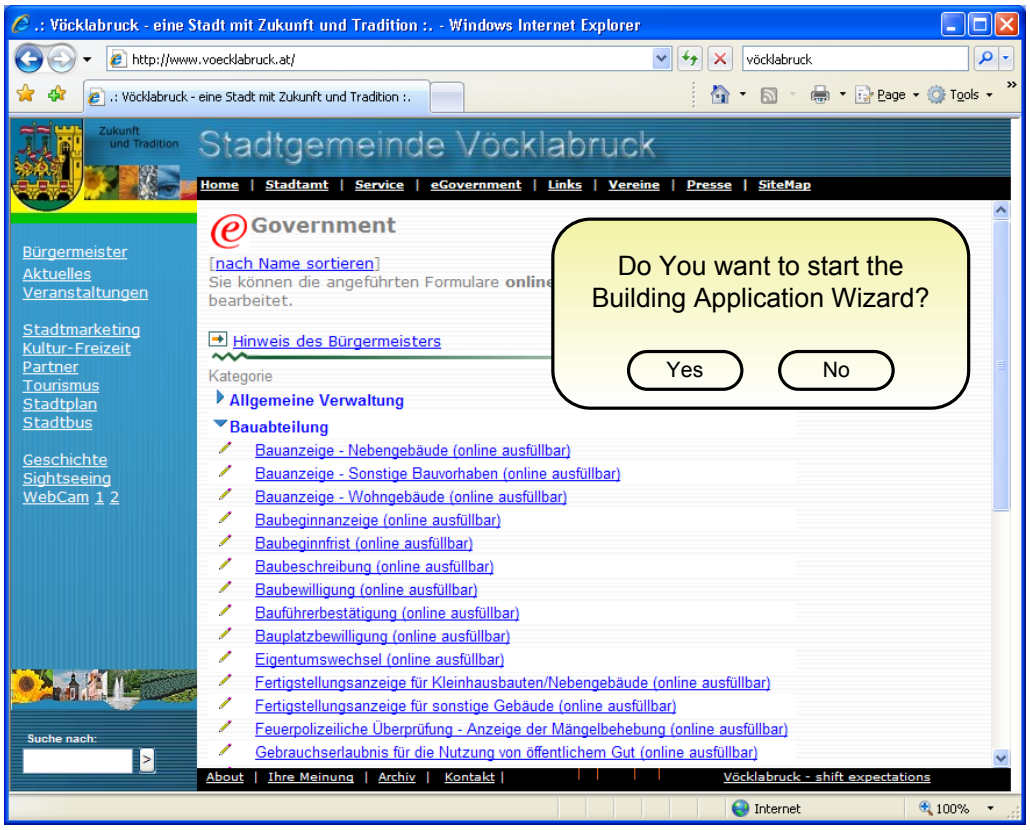

Figure 4.3: Direct user guidance is about to start.

the semantic WUM phase, slightly modified standard web mining algorithms are used for mining semantically annotated web server access log files. Finally, the discovered web usage patterns are evaluated by domain and usability experts in order to extract really useful adaptation rules. The Modeling Framework supports the discovery of useful adaptation rules and their design. These services are run offline in the back-end (on the server-side).

Research Challenge 3

Design of a client-side executable, declarative adaptation rule language

We defined the expressive, yet notionally simple ARL. The aim of this language is to declaratively express personalization patterns. To meet the *third* research challenge the ARL is client-side executable and declarative. As adaptation rules have to be executed on the client in a web browser we chose a browser-friendly and directly executable format. ARL is able to express temporal, logical and spatial interdependencies between user actions (events) and to evaluate conditions over business objects (facts).

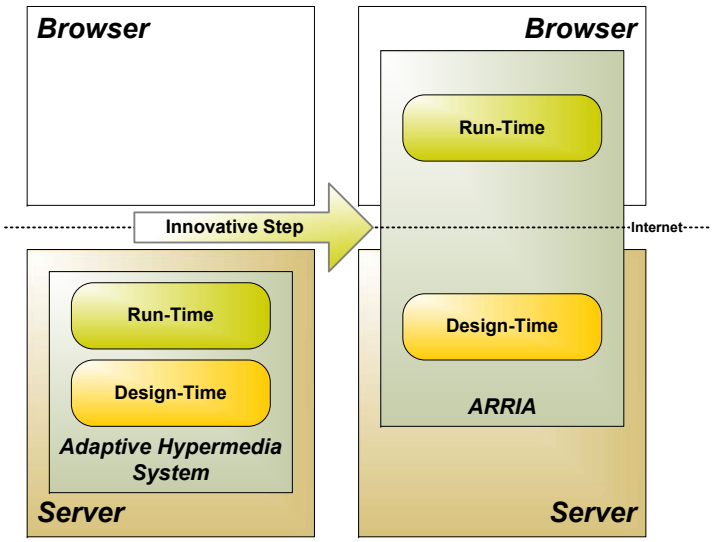

Figure 4.4: Research Challenge 1 — the basic idea of the ARRIA approach.

Research Challenge 4

Design and implementation of an efficient and scalable client-side adaptation engine

We realized our vision of adaptive and reactive RIAs by developing an autonomous Adaptation Engine which runs directly on the client, and, thus, enables the punctual personalization of web applications implicitly based on the current working context of the individual user. The Adaptation Engine can be implicitly embedded in a single web application, or, explicitly installed in a user's web browser. Compared to the per application delivery of the Adaptation Engine, the installation of the Adaption Engine directly in a user's web browser enables more fine-grained detection of a user's working context, as the user can now be tracked across several browser tabs, and, thus, across application boundaries.

We designed and implemented the Adaptation Engine as a client-side CEP engine which is capable of detecting not only HTTP requests but also the complete clickstream of an individual user. The fully fledged event processing engine is based on the event detection algorithms of Sentinel (cf. Section 2.6). Furthermore, we designed and implemented a production rule system based on the Rete algorithm (cf. Section 2.5). Finally, we researched an efficient approach for combining Sentinel's event detection algorithm with the Rete algorithm. The prototypically implemented Adaptation Engine combines the event processing and production rule engine. The whole Run-Time Framework is designed to support an efficient solution to the *fourth* research challenge, as the client-side Adaptation Engine grants adaptivity and reactivity to RIAs by collecting fine grained user data, ad-hoc user modeling and on time adaptation rule execution.

4.4 Related Work

We compare the overall ARRIA approach to the following related work: the Adaptive Hypermedia Applications Model (AHAM) by Romero et al. (2005), the Personal Reader by Dolog et al. (2004), the approach of personalizing the presentation of hypermedia content to the user by Frasincar and Houben (2002), the Web Accessibility Initiative - Accessible Rich Interactive Applications (WAI-ARIA) by Craig et al. (2009), WebMate by Chen and Sycara (1998), and, finally, Vistabar by (Marais and Bharat, 1997).

Comparing our work with standard models for AHSs like, e.g., the AHAM, introduced by Romero et al. (2005), we follow the same approach of dividing AHS models into domain and user models, omitting however, the underlying teaching model, as ARRIAs gather the interests of the individual user and not his/her knowledge about a particular domain.

A prominent example of a AHS is the Personal Reader (Dolog et al., 2004). The Personal Reader provides a pure server-side framework for the design of personalized static views on web resources based on their semantic descriptions. Again, ARRIAs follow a different approach, as they collect user behavior at run-time directly on the client-side, and, based on this, personalize the web application on-the-fly.

Similarly, Frasincar and Houben (2002) focus on content adaptation, or, more precisely, on personalizing the presentation of hypermedia content to the user. Like the Personal Reader this approach does not focus on the on-line discovery of the profile of the current user, as ARRIAs do.

The transformation, the accessibility and interoperability of RIAs for persons with disabilities is addressed by a World Wide Web Consortium (W3C) initiative (Craig et al., 2009). It focuses on the transformation of widgets developed with Ajax, Hypertext Markup Language (HTML), JavaScript, and related technologies. WAI-ARIA statically changes the representation of single UI widgets in order to provide better readability. In contrast, ARRIAs focus on the adaptation of a web application as a whole, that is, its UI, structure and content based on collected behavioral data of the individual user, which is not utilized in the WAI-ARIA approach.

Chen and Sycara (1998) present *WebMate*, a personal agent that helps users while browsing and searching. WebMate provides personalized help based on a incrementally and continuously learned user model. In contrast to ARRIAs, it follows a proxy-based approach which needs additional installation effort. Due to this proxy-based approach the WebMate approach has only limited capabilities of tracking users clickstream data. Actually, only explicit user requests can be tracked. Here ARRIAs are advantageous, as they are able to track nearly any user actions like, for instance, mouse movements. Another difference is that WebMate does not enable web applications providers to use the user model in order to personalize their applications accordingly. Moreover, ARRIAs differ

from WebMate in their method of gaining metadata from web pages. WebMate utilizes the TF-IDF[2] method for the content assessment of web pages, where as, ARRIAs utilize semantic annotations.

Marais and Bharat (1997) propose *Vistabar* as a desktop assistant supporting cooperative and personal surfing. In contrast to ARRIAs Vistabar does not adapt web pages according to a collected user model, but focuses on commenting upon web pages and bookmarks. It is a hybrid client-side and server-side approach. The Vistabar can be attached to any web browser. Like WebMate it cannot be used by web application providers in order to deliver personalized RIAs like ARRIAs do.

4.5 Summary

In this section we clarified the main features of ARRIAs from a bird's eye view. We explained that lifting the Adaptation Engine from the server up to the client is an innovative and evolutionary step in adding adaptivity and reactivity to RIAs. We explained the different parts and processes of the Adaptation Framework and why it consists of two different frameworks: the Design-Time and the Run-Time Framework. We demonstrated the application of ARRIA's Adaptation Framework in the light of a modified version of the motivating example from Section 1.1. By detailing our approach to the Research Challenge 1 we demonstrated how a holistic Adaptation Framework, considering both the design-time as well as the run-time aspects of providing ad-hoc adaptivity to RIAs, is able to bridge the gap between the server-side adaptation approach of legacy AHSs and the client-side approach of modern RIAs.

[2]The TF-IDF method is an algorithm which represents each document as a vector in a vector space. Similar vectors correspond to documents with similar content. TF stands for term frequency and IDF for inverse document frequency.

Chapter 5

Obtaining the Adaptation Rules

In this chapter we detail our approach of obtaining adaptation rules by elaborating the four phases of the modeling cycle: ontology design, annotation of web applications, knowledge-enhanced mining of web usage behavior and adaptation rule creation. The process of gathering adaptation patterns is explained by means of the e-Government portal of Vöcklabruck (cf. Section 3.1). We presented this approach to the second research challenge (cf. Section 1.3) at the *4th European Semantic Web Conference* (Schmidt et al., 2007), at the *7th International EGOV Conference* (Rahmani et al., 2008), and, at the *e-Challenges Conference* (Stojanovic et al., 2007c).

In Section 1.3 we hypothesized that by enhancing standard methods from the research field of CI with formal semantics from ontologies the acquisition of meaningful adaptation patterns will be facilitated. Background knowledge is needed because the raw data of user access log files consists of administration-oriented URLs, whereas adaptation analysts are interested in events/entities in the application domain, i.e. in the semantics underlying a web session. Thus, the log data has to be characterized at a deeper semantic level using entities from the domain ontology. Consequently, it is important to create a mapping between the web log entries and the domain entities. The semantic information can be leveraged at various steps in the knowledge discovery process, namely in the preprocessing phase, in the pattern discovery phase, or during the post-processing of the discovered patterns. Special attention is paid to the preprocessing phase that determines which information should be used for learning.

A primary expected general benefit of semantic web usage mining is ease of interpretation of the results, since the patterns are more meaningful. Rules are easier to interpret, since they are expressed using concepts rather than uniform resource locators (URL). This makes it easier to see if rules make sense. Furthermore, semantic WUM enables the analyst to focus the search for rules by semantic annotation. For example, the search for rules can be limited to a particular subject or set of subjects. Similarly, the search might focus on a target group or set of related target groups.

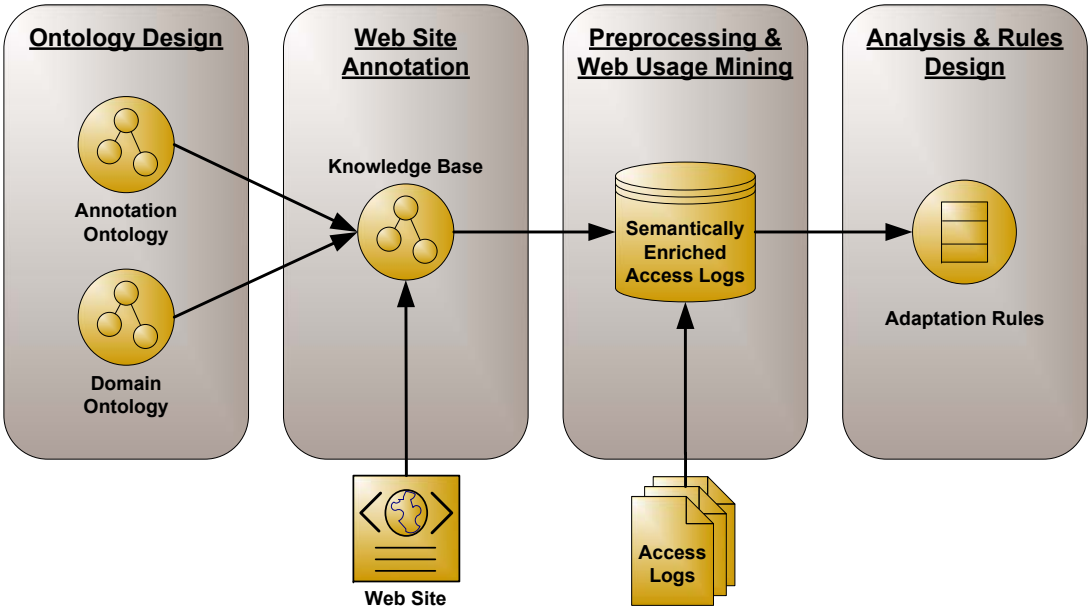

Figure 5.1: The four phases of the Modeling Cycle and their artifacts.

Figure 5.1 provides a conceptual overview of the Design-Time Framework and its artifacts. The framework has been designed with the objective to leverage recorded user behavior in a web portal and semantic knowledge about that portal in order to gain web portal usage patterns of high quality. The phases correspond to the chronological execution order of the building blocks and to the phases of the Modeling Cycle of the Design-Time Framework depicted in Figure 4.1.

During the ontology design phase the Annotation Ontology and the Domain Ontology are created iteratively. This phase is followed by the web Site Annotation phase. In the web site annotation phase existing web sites are annotated with concepts from the ontologies. The resulting annotations are stored in the Knowledge Base. During the preprocessing and mining phase these annotations are mapped to the reconstructed sessions of web access logs, preprocessed, and stored in a database named Semantically Enriched Access Logs. The actual mining process applies standard web mining methods to the Semantically Enriched Access Logs. In the last phase, the analysis and rules design phase, the mined behavioral patterns are examined and evaluated by domain experts, and, finally, meaningful adaptation rules are extracted.

This chapter is structured according to the four phases of the Modeling Cycle (cf. Subsection 4.1.1) also depicted in Figure 5.1. In Section 5.1 the ARRIA ontologies, the paving stones of the personalization highway, are introduced. The Annotation Ontology and the e-Government Ontology are detailed. We explain the concepts and properties of these two ontologies and their interrelations. Thereafter, in Section 5.2, we show how to link existing web applications to ontological concepts. In the next section, Section 5.3, we demonstrate our approach

to mining web usage behavior based on semantically annotated web access log files. Thereafter, in Section 5.4, the analysis and rules design phase is described, in which the discovered web usage behavior patterns are checked for reasonability, and in which the adaptation rules are formalized. This section is followed by an evaluation section, Section 5.5. Section 5.6 compares our design-time approach to related work in this field. We conclude this chapter by summarizing and discussing our approach to gaining meaningful adaptation rules.

5.1 Ontology Design

Suitable ontologies are crucial for mining common web usage behavior patterns (Berendt et al., 2002). The ARRIA ontologies serve the purpose of making web usage behavior machine interpretable at design-time and of recommending concept-level personalizations at run-time.

The ARRIA ontologies and their interrelations are depicted in Figure 5.2. We designed two ontologies: the Annotation Ontology and the e-Government Ontology. The domain specific E-Government Ontology was developed jointly with public sector experts from Vöcklabruck.

According to Corcho et al. (2003) our iteratively constructed ARRIA ontologies can be categorized as lightweight ontologies as we modeled mainly taxonomies. During the construction phase we followed a faceted classification approach as proposed by Díaz (2003) and Broughton (2006). By using the faceted classification approach an object can be classified in multiple ways. Assuming that faceted classification is used to design a portal, it makes easier to find the web object because it appears in multiple places.

We captured the Annotation Ontology using a top-down approach[1]. The domain-specific E-Government Ontology was developed by using a bottom-up approach[2]. The domain ontology is constructed by combining semi-automatic ontology learning methods provided by an extension of *Text2Onto* (Cimiano and Völker, 2005) with a semantic tagging approach realized in the Annotation Editor (Stojanovic et al., 2007e,a, 2008).

As a knowledge representation language we first used the Web Ontology Language (OWL) (Bechhofeer et al., 2004) and then in June 2009 we switched to the newly developed OWL 2 (Grau et al., 2008) as soon as it became a W3C Candidate Recommendation (Motik et al., 2008).

[1] A top-down development process starts with the definition of the most general concepts in the domain and subsequent specialization of the concepts (Noy and McGuiness, 2001).

[2] A bottom-up development process starts with the definition of the most specific classes, the leaves of the hierarchy, with subsequent grouping of these classes into more general concepts (Noy and McGuiness, 2001). The bottom-up approach is the natural fit for the development of faceted ontologies.

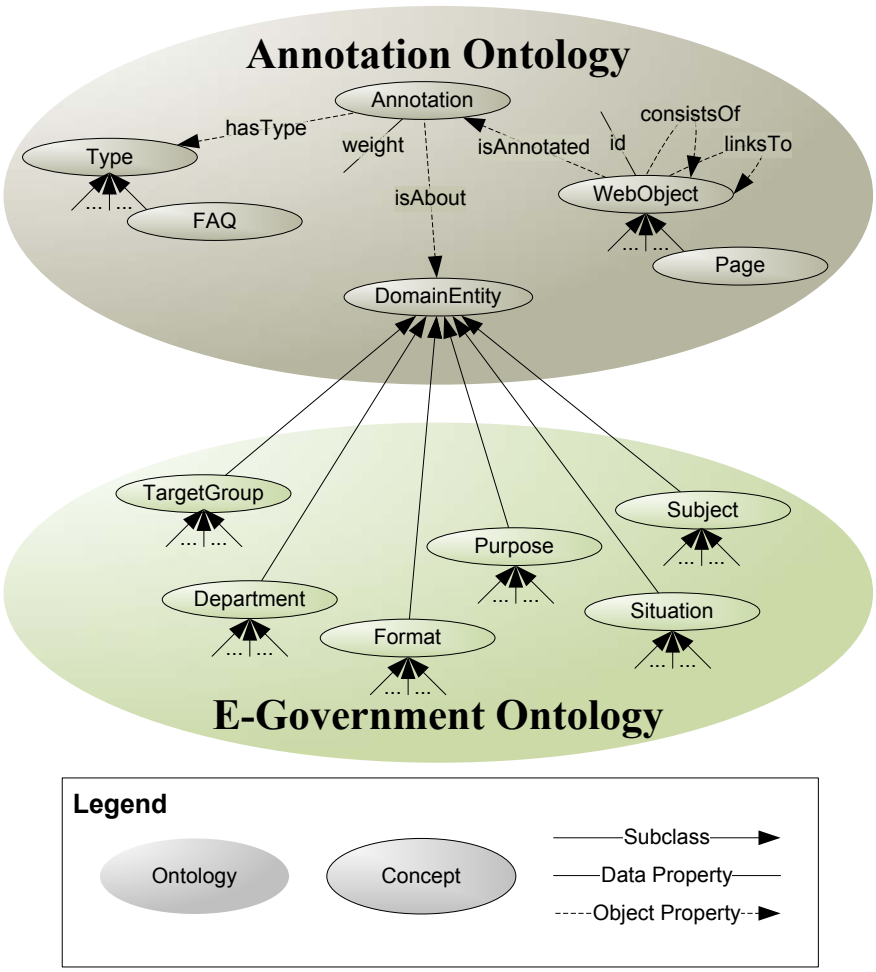

Figure 5.2: Parts of the ARRIA ontologies showing several entities of the Annotation Ontology and the domain-specific E-Government Ontology as well as dependencies between them.

5.1.1 Annotation Ontology

The Annotation Ontology defines the top-level concepts for linking the domain specific concepts to real web applications. These top-level concepts are: `WebObject`, `Annotation`, `DomainEntity` and `Type`.

The WebObject Class

The `WebObject` class describes all web objects of a web page and their interrelations. It is the top-level concept for describing the internal, physical structure of web applications in general. The way that the web site is physically laid out as well as the structure of each page can be useful for understanding usage behavior. Thus, the `WebObject` is a superconcept of any HTML entity representing

a structural element of a page such as `Page`, `Frame`, `Table`, `Hyperlink`, `Form`, `Image` etc.

The `WebObject` has the three object properties `isAnnotated`, `consistsOf` and `linksTo`, and, additionally, the data property `id`. `isAnnotated` is the central property of the Annotation Ontology. It links web objects and annotations. `consistsOf` is used to model a web object consisting of several parts which again are web objects themselves. We constrained this property by excluding the case of self-inclusion. `ao:linksTo` models the transitive relationship of referencing other web objects via hyperlinks. `id` links `WebObject` instances to web objects residing inside a web application via HTML's universal *id* attribute. We assume unique ids within one and the same application.

The Annotation Class

The `Annotation` class models the annotation of a web object. The `isAbout` object property links domain concepts to web objects. Web objects are annotated with any subclasses of `DomainEntity` as modeled in the domain-specific E-Government Ontology. Additionally, the type of a web object can be described by using the `hasType` object property. Thus, a page containing mainly hyperlinks can be annotated with the concept `Navigational` which in turn is a subconcept of `Type`. An annotation might have a weight assigned by the `weight` data property. Weight is a measure to identify the importance of a facet for the annotated web object. A greater numeric value means higher importance. Weights are important for multi-level mining in order to find adaptation patterns based on higher level concepts which is explained later on in this chapter.

The DomainEntity Class

The `DomainEntity` class is the top-level concept for modeling different aspects or viewpoints of a web object and the domain entity it represents. `DomainEntity` is the superclass of all domain specific concepts like, for instance, of the `TargetGroup` concept defined in the E-Government Ontology.

The Type Class

The `Type` concept is a category corresponding to different functions of web objects (Pirolli et al., 1996). We predefined the common types that can be found in the literature (Cooley et al., 1999) like `Navigation`[3] or `Content`[4]. We refined the classical selection by additional facets as some web objects of the web portal

[3] A navigation web object is annotated as such if it provides links to guide users on to content objects.

[4] A web object categorized with the Content concept contains a portion of the information content that the web application is providing.

of Vöcklabruck needed a more fine grained description. Thus, for instance we added `FAQ`[5] and `Search`[6].

We expect that information about page structure can be used to derive or to verify the type of a web object. For example, a navigational web object is a web object with small content/link ratio, short time spent on page and which is not a part of a maximal forward reference[7]. We note that types are not exclusive. A page could consist of more than one type.

5.1.2 Domain Ontologies

In IT, domain ontologies model specific aspects of a domain with respect to their intended role in a particular software system. Exemplarily, we show by means of the personalized e-Government use case, without loss of generality, how to construct an ontology specifically for the domain of e-Government portals and their personalization. The domain-specific E-Government Ontology consists of concepts modeling the meaning of services/information offered by an e-Government portal from different viewpoints such as residential affairs, residential permissions, identification, certifications, naturalization citizenship, moving, education, etc.

In order to gain the most specific classes to start from we analyzed several existing e-Government portals of Austrian municipalities[8]. Additionally, the ontology has been developed based on existing standards for modeling life events such as the Swiss Standard eCH-0049[9].

In Table 5.1 the six top-level facets and their documentation are listed. Each of the top-level concepts is subclassed by other concepts. Generally, each facet is the starting point of a three to four level class hierarchy. In order to publish the domain facets to the Annotation Ontology they are subsumed by the `DomainEntity` class (cf. Figure 5.2).

[5] An FAQ web object contains frequently asked questions and answers to them.

[6] An web object annotated with this facet provides some sort of user query driven IR.

[7] A maximal forward reference of a web user is a longest consecutive sequence of web pages visited by the user in a session without revisiting some previously visited page in the sequence (Chen et al., 1998; Cooley et al., 1999).

[8] Concretely, we analyzed the web sites of Vöcklabruck (`http://www.voecklabruck.at`), Villach (`http://www.villach.at`), Graz (`http://www.graz.at`) and Linz (`http://www.linz.at`).

[9] Swiss Standard eCH-0049 *Themenkataloge für E-Government-Portale* (`http://www.ech.ch/index.php?option=com_docman&task=doc_download&gid=2773&Itemid=25&lang=en`) gives an overview of all relevant e-Government services in Switzerland, and, therefore, provides a consistent and standardized classification of public administration services.

Table 5.1: E-Government domain facets.

Type	Description
Department	This facet gives the type of department responsible for the displayed information or service.
Format	Public administration services appear in e-Government portals in formats like information services, online service, online service demanding a digital signature, download forms service etc. This facet is very similar to the Type facet of the Annotation Ontology but specialized for the e-Government domain.
Purpose	The Purpose facet gives the general intention of the web object like for instance notification, authentication or money transfer. Additionally, it harmonizes different terms with the same meaning in a way a thesaurus (Fellbaum, 1998; Kunze and Rösner, 2004) does like for example declaration, announcement, statement, notification, report etc.
Situation	The Situation facet describes life events like birth, marriage, travelling, living etc.
Subject	The raison d'être of a web object can be annotated with this facet. Subjects of e-Government forms are for instance education, construction, public security, family, local authority etc.
TargetGroup	The intended audience of a web object can be annotated with one or many subclasses of this facet like for instance singles, couples or senior citizens.

5.2 Annotation of Web Applications

The linkage of ontologies to real web applications and their components is carried out in the annotation phase. So far we have developed the *TBox* [10] for the ontologies. What comes next is their instantiation, that is the construction of the *ABox* [11]. We store the *ABox* individuals forming the semantic annotations of a web application in the Knowledge Base. Thus the Knowledge Base stores all annotations describing the different facets of concrete web objects.

During the annotation phase public sector experts annotate the web applications of interest in order to take advantage of the formal semantics of ontologies. Stojanovic et al. (2007d) present a supporting tool which eases the annotation of web applications, called the Annotation Editor, which also has been developed in the course of the FIT project.

[10] Nardi and Brachman (2003) define the TBox for description logic (DL) as the box that contains intensional knowledge in the form of a terminology or taxonomy and is built through declarations that describe general properties of concepts.

[11] Nardi and Brachman (2003) define the ABox for description logic (DL) as the box that contains extensional knowledge that is specific to the individuals of the domain of discourse.

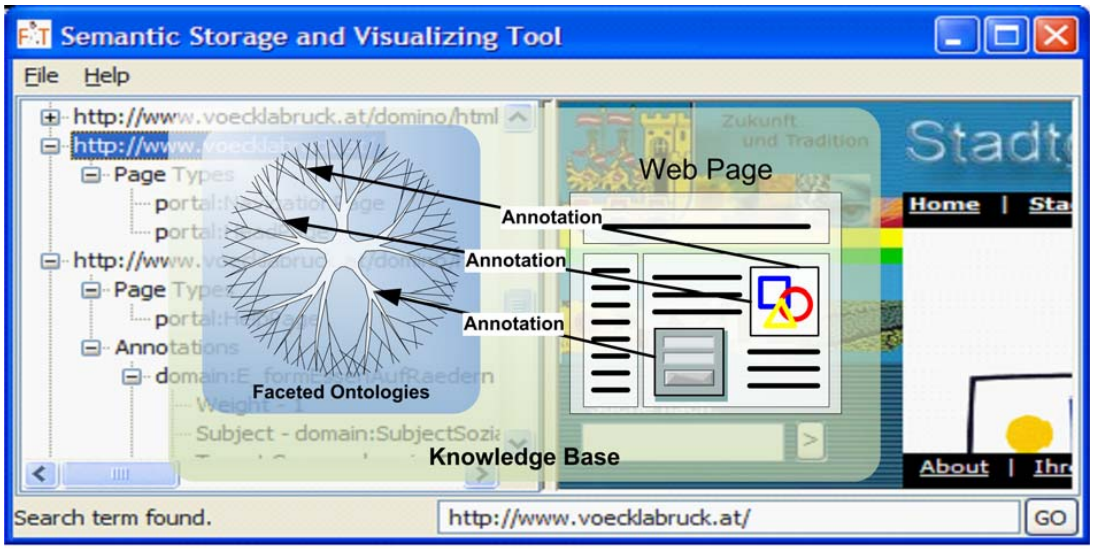

Figure 5.3: Annotation of web page elements to concepts.

For instance a web page to apply for a building permission is described by the following concepts from the facets *Department*, *Purpose*, *Subject*, *Target-Group* and *Situation*: *BuildingDepartment*, *Application*, *GeneralBuildingApplication*, *Builder* and *BeginOfBuilding*, in that order.

A screenshot showing the Knowledge Base for the e-Government portal of Vöcklabruck is given in Figure 5.3. The split window displayed in Figure 5.3 shows on the left the URLs of web objects and their annotations, and, on the right, it displays the corresponding web page in a web bowser pane. After the construction of the Knowledge Base, it has to be transformed into a format suitable for semantically enhanced web usage behavior mining. Furthermore, it has to be linked to the recorded web usage data of individual users and their reconstructed sessions. This is explained in detail in the next section.

5.3 Preprocessing and Web Usage Mining

We designed a two step approach to preprocessing. The two steps are: conventional preprocessing of user access log files and preprocessing of semantic annotations. We implemented them as Java applications in the *Preprocessor* and *Semantic Preprocessor*, respectively. In the conventional preprocessing step implemented in the Preprocessor, the original log files are cleaned by marking irrelevant entries. Additionally, user web sessions are reconstructed in order to group the page view entries of the log files. Many interesting basic questions about usage intensity, session duration and page popularity or unpopularity can be answered by analyzing sessions. When combined with semantics this becomes even more powerful. Semantics can be used to discover not only popular pages,

but also popular topics. These might not show up as popular pages if the topic is distributed across many pages. Therefore, in the semantic preprocessing step, the annotations stored in the Knowledge Base are preprocessed and linked to the sessions reconstructed from the web access log files.

After preprocessing the access log files and the annotations, WUM methods can be applied to the semantically enriched access logs database in order to obtain meaningful adaptation rules. Whether the mined web behavior patterns are meaningful or not is determined in the Analysis and Rules Design phase.

5.3.1 Conventional Preprocessing

In the phase of conventional preprocessing the user access log files are cleaned[12]. This includes marking all irrelevant document formats like pictures but also removing robot, duplicate and bad requests. Based on a cleaned log file user sessions are reconstructed. Here the major challenge is user identification. User identification is no problem if the user logs in. But with ARRIAs we assume no explicit user login. Therefore heuristics are used. Heuristics apply a combination of IP address, machine name, browser agent and temporal information to identify users. Our solution relies one standard heuristics as introduced by Cooley et al. (1999) (cf. Section 2.4). The concrete realization of the preprocessing of user access log files is described in detail by Schmidt and Thomas (2007).

After cleaning the log file and reconstructing sessions all log entries (marked and unmarked) are stored in a database together as user sessions. The database schema that stores the log entries and the user sessions is described by Magoutas et al. (2007). We decided in favor of a database because a database is superior in flexibility and performance compared to a plain file.

5.3.2 Semantic Preprocessing

Cleaning a log file and reconstructing user sessions are an indispensable prerequisite for finding web usage patterns by data mining techniques (Srivastava et al., 2000). But conventional web usage mining can only find patterns based on URLs. This has two major drawbacks: the patterns cannot be generalized and in the end are hard to interpret by a human operator. To overcome these limitations we followed the approach of semantically enriching the log data with domain knowledge.

For combining user access log files and concepts from the ARRIA ontologies we developed the Semantic Preprocessor. The Semantic Preprocessor is a user-centric, easy to use and comprehensive tool that considers implicit knowledge

[12] We are able to process the most common access log formats: Apache's common and combined log format (`http://httpd.apache.org/docs/1.3/logs.html`). Additionally, we are able to process log files from the IBM Lotus Domino Web Server (`http://www.ibm.com/software/lotus/products/domino`).

in many different ways. As output not only database tables are produced, but also Attribute-Relation File Format (ARFF) files, as required by well-known open-source data mining tools e.g. Waikato Environment for Knowledge Analysis (WEKA)[13]. The Semantic Preprocessor enables the user, e.g., a domain expert, to select the semantic annotations needed by the web usage mining algorithms via a convenient user interface, to set up the weights for annotations, to control the representation of training examples, etc. The Semantic Preprocessor is described in great detail by Stojanovic et al. (2007b).

Goals and Challenges

Our goal is to assign domain entities to user navigational paths/patterns by mapping the pageview names or URLs (i.e. web log entries) to the instances in the knowledge base. To be more specific, instead of describing a user's navigational path as: $(p_1, p_2, \ldots, p_n)$ (where p_i is a URL pointing to a web page), we need to represent it using the domain entities used for annotating these pages in the knowledge base, such as: `Apartment`, `OutBuilding`, ..., `DomainEntity`.

However, the mapping of user activities at the clickstream level, stored in the Web Log, to concepts derived from the explicit annotation of pages is very complex, since there is no 1:1 mapping. Indeed, one Web Log entry may be transformed into several entries in the semantically-enriched log (e.g. in the case that a visited page is annotated with several domain entities). An example for such a mapping is shown in Figure 5.4. Since page *P1* is annotated with two domain entities (cf. `Apartment` and `OutBuidling`), the first Web Log entry indicating that *P1* was visited is transformed into two entries in the resulting Semantically Enriched Web Log table. Additionally, several Web Log entries may be conceptually grouped into one entry in the Semantically Enriched Web Log (e.g. in case several entries belonging to the same session represent visited pages annotated with the same domain entity). This is demonstrated with the session *S2* in Figure 5.4. Even though session *S2* contains two entries in the standard Web Log (i.e. pages *P2* and *P3* belonging to this session were visited) it could be also represented in the Semantically Enriched Web Log as only one entry labeled with the `Homestead` domain entity, since both pages are annotated only with this entity. An alternative way of computing concept weights in a class hierarchy was described by Aleman-Meza et al. (2003).

The integration of domain knowledge generates semantic usage patterns, introducing great flexibility as well as challenges. We have identified the following two research challenges: First, how to obtain implicit knowledge for transforming user sessions into semantically enhanced sessions containing the semantics of the visited pages. Second, how to calculate the importance of an ontology entity for an entry in the semantically enriched log.

[13] `http://www.cs.waikato.ac.nz/ml/weka/`

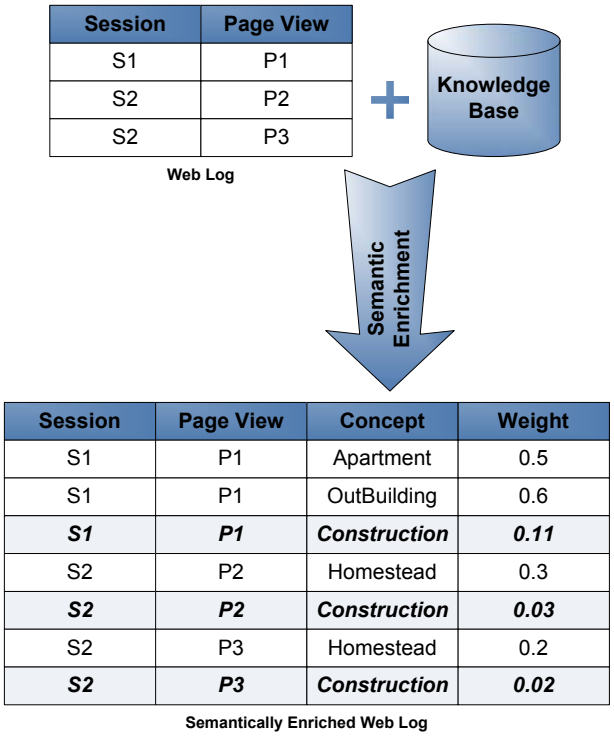

Figure 5.4: Examples of transforming web log data into semantically enriched log data.

Meeting the Challenges

Reasoning. Preprocessing based on the ontology entities used for annotation can be performed by applying reasoning. Indeed, one of main advantages of using ontologies for web usage mining is reasoning, which can be defined as an act of deriving a conclusion based solely on what is already known. By applying reasoning to the domain ontology and to the Knowledge Base, new, implicit information can be derived and the Knowledge Base can be extended automatically.

For example, consider the class hierarchy of `Construction` as superclass and `Apartment`, `OutBuilding` and `Homestead` as subclasses of `Construction`. Given this subsumption hierarchy we can infer that the class `Apartment` is a subclass of `Construction`. By assuming that a page is about a parent entity of an entity it is annotated with, the semantically enriched log shown in Figure 5.4 will be extended with the new entries illustrated in bold and italic in the Semantically Enriched Web Log in Figure 5.4.

The complexity of semantic enrichment can be decreased by restricting the hierarchy level. The hierarchy level can be restricted by specifying an arbitrary concept, for instance, of the faceted and domain-specific E-Government Ontology (cf. Subsection 5.1.2). However, controlling the way in which the hierarchy is considered provides the ability to filter the input for semantic enrichment. The classification of ontology entities based on a class hierarchy can be used to

limit the log entries to those containing pageviews only up to a certain hierarchy level. Thus, for instance, by restricting the hierarchy level to the `Construction` concept of the E-Government Ontology only those sessions are considered for the subsequent pattern discovery phase which contain log entries annotated with the `Construction` concept or its subconcepts.

Weights. The importance of an entity for semantic preprocessing, is indicated by a weight which can range between [0,1], where 1 is most important. Usually different weights are associated with ontology entities when annotations are created. For annotations defined by domain experts, weights are usually provided as a part of the domain knowledge specified by the experts. Such weights may reflect the relative importance of certain concepts.

The computation of weights for the class hierarchy of an ontology can be of arbitrary complexity. There may be several different subsumption paths between two classes. Additionally, diverse paths between the same two classes may be of different lengths, where the length is determined by the number of the intermediate classes. Moreover, it is possible to have cycles in the hierarchy; there is no need to specify the subsumption relation explicitly — this can be derived e.g. based on properties defined for the concepts, etc.

To cope with cycles in the hierarchy, we consider each class only once. Implicit subsumption relationships are discovered by applying reasoning. Thus, the weight definition problem is reduced to a path calculation in a directed graph. The following metric with a value range between [0,1] is proposed:

$$weight(P, L) = \frac{\sum_C \frac{weight(C,L)}{numberOfDirectParents(C)}}{numberOfDirectChildren(P)} \tag{5.1}$$

P is a parent class, L is a web log entry (i.e. pageview or session), and C is a direct subclass of a class P. Thus, the weight of a class is propagated only to direct parents, used for the annotation of a page visited in the considered web log entry. Additionally, the impact of the child on the direct parents it has is determined by the number of other parents/children. By applying Formula 5.1 recursively the weights of all superconcepts can be computed.

In the bold plotted rows of the Semantically Enriched Web Log in Figure 5.4 the computed weights for the `Construction` concept are shown. They are calculated by taking into account the subsumption hierarchy of `Construction` as the only superclass of the annotations `Apartment`, `OutBuilding` and `Homestead`. In addition to these three classes the `Construction` concept subsumes another seven classes. The weighs of the `Construction` concept depicted in bold in Figure 5.4 are calculated with the Formula 5.1 by using the actual weights of `Apartment`, `OutBuilding` or `Homestead`.

5.3.3 Pattern Discovery

In the pattern discovery phase meaningful patterns are identified based on a cleaned web user access log file by exploiting semantic annotations. The found patterns are used to find ideas for adaptation rules for use in the adaptive front-end (this use is personalization). Based on an analysis of this field of research, we concluded that the most useful techniques for our purposes were association rule mining (Agrawal and Srikant, 1994).

The input-data for pattern discovery are gained either directly from the database by firing Structured Query Language (SQL) queries or by using the output of the Semantic Preprocessor. The results are again stored back into several result tables or into files (e.g. in ARFF format) as required by well-known open-source data mining tools (e.g. WEKA). A number of algorithms to find association rules were considered:

- Apriori and AprioriTid, described by Agrawal and Srikant (1994),
- FP-Growth, described by Han et al. (2000) and
- Closure and ClosureOpt, described by Cristofor et al. (2000).

The FP-Growth algorithm was chosen because its performance has been shown (Han et al., 2000) to be better for generating rules with low support value. This was important to us for several reasons. First, we want to find problems or opportunities for which the support value might be small. Second, rather than re-run the algorithms, we want to set a small support value and use SQL, if necessary, to reduce the number of rules shown. Generally, we set a support and confidence of 0.01, which generates a large number of rules, which are written into a database with their support and confidence values. Then we can use SQL queries to display only those with higher support and confidence values. As the implementation, we choose the one in the ARtool, mostly because of its performance, but also because the code is easy to understand.

5.4 Analysis and Rules Design

Based on the results of the Preprocessing and Web Mining Phase usability experts are able to design rules for on time personalization of web applications. In the case of inadequate discovered patterns like too many, too few, not meaningful, etc. the usage mining expert goes back to phase two and restarts the Pattern Discovery building block with adjusted attributes. E.g. to find more or less patterns he/she can change the confidence and support values of the data mining algorithms. After restarting the data mining algorithms the newly found patterns can be evaluated. This leads to an iterative pattern discovery approach. How

exactly adaptation rules, that can be executed directly on the client at run-time, are formulated is detailed in the next chapter.

In some cases, it makes sense to use pages, which are usually identified by Uniform Resource Identifiers (URI), as the input to mining algorithms, and then to use semantics to interpret the discovered patterns. For example, in the simplest case, to help interpret a discovered rule, the URIs can be replaced by the names of the concepts used to annotate the pages. These names are generally much more meaningful than URIs.

5.5 Evaluation

We conducted a practical evaluation on the e-Government portal of Vöcklaburck, whose service delivery mainly comprises submission of electronic forms. These form pages, along with some other pages, were annotated using the faceted E-Government Ontology that describes such forms according to their subject, target group, etc.

Vöcklabruck provided us with $1\frac{1}{2}$ years of log data. To perform web usage mining the log data for the first 5 months of 2007 were used. This seemed like enough data, and would be more likely to correspond to the current web site and the annotations made for it. This data yielded 205,482 sessions, for these sessions there were 2,267,957 log entries.

5.5.1 Preprocessing

In data mining the quality of the input data is crucial (Witten and Frank, 2005), so, we did an initial sessionizing run and then checked the quality. A number of problems were discovered. One was that some pages had multiple URLs. We were able to cope with this by creating equivalent annotations. We were, however, concerned about the quality of the log because there were many URLs in the log that pointed to pages that contained a note saying they had expired. And these URLs were in sessions with a date after the expiration date. There were also some URLs that were no longer available on the site.

As it turned out, sessionization was almost impossible. The goal of sessionizing is to recreate the user click stream, which involves filtering out all the non-click files, e.g. the pictures that are loaded as sub-elements of a page. One click might create 50 log entries, because 50 different files are loaded to draw the page. Normally the .htm and .html files in a log correspond to a user click, so by filtering out the files with other extensions, like .gif, it is possible to reconstruct a user click stream. But in the Vöcklabruck log, some sub-elements of a page, like pictures had .htm or .html as an extension, or had no extension at all, so were not filtered out. The end result was that it was not possible to reliably filter down to user clicks.

Since we could not really properly sessionize the log, and, since many pages were on other web sites, with no access to the log files, we reduced the data mining to sessions which contained at least one page that had been annotated. This reduced the sessions down to 6817. Clearly, the annotated URLs are entire pages, and therefore, do represent user clicks.

5.5.2 Annotation

The Annotation Editor enables the annotation of whole pages, text on a page and links. For this evaluation we decided to annotate just pages and links, since we were most concerned to find association rules between pages and associated concepts. But, we were unable to annotate links due to the dynamic nature of the pages on the portal.

The portal has a relatively low level of use, since it is a smallish community. To save time, we first asked what was used frequently, and tried to verify this in the log. This lead to the discovery of the previously mentioned problem of multiple URLs for the same page. However, in the end, we decided to annotate everything that we could with the domain ontology we had. We annotated the start page, the site map page, some contact pages, some help pages, and, of course, all the pages that contained forms. In addition, we annotated all the navigation pages leading to the forms. The total number of pages annotated was about 80.

5.5.3 Mining and Analysis

We confirmed that replacing URIs with concept names aids interpretation. Assuming concept names are well-chosen and human-readable, the concept-based rules are easier to interpret.

We also confirmed that, in general, mining the concepts rather than the pages results in more rules being found, because multiple pages are often annotated with the same concept. With multi-level mining we found some rules at higher levels of generalization, when none were to be found at the more specific levels. Multi-level rules (Han and Fu, 1995, 1999) can be mined when a generalization hierarchy, like our faceted E-Government Ontology, exists. For example, we mined the registry office pattern as described in Subsection 3.1.1. In Table 5.2 an excerpt of the discovered association rules, which have also been confirmed by a public sector expert, is listed. The concepts are subconcepts of the `Subject` facet of the E-Government Ontology.

5.5.4 Discussion

Based on our experience a quality-check tool to run before usage mining would be extremely valuable. Such a tool would create a graph of the pages in the web

Table 5.2: Discovered association rules.

Antecedent		Consequence	Support	Confidence
wedding day	⇒	marriage certificate	0.01218	0.35483
marriage certificate	⇒	birth certificate	0.01203	0.30400
registration form	⇒	meals on wheels	0.01519	0.30967
building notification	⇒	sidewalk usage permit	0.01139	0.30379

site (called the site topology), and then filter out any sessions that contain pages no longer in the site topology. There are some tools that crawl a site and create such a topology. In fact, we attempted to use them, but none could cope with the dynamic nature of the Vöcklabruck portal.

Developing the ontology using the approach of faceted classification is very promising. However, it turned out, that the public administration needs help creating the ontology, which implies that they need higher-level tools, problem-specific tools that are simpler than general ontology editors like Protégé[14].

Annotation should be built into the process of creating the portal content. Although the Annotation Editor is easy to use, our experience leads us to the conclusion that annotation is best done when content is created, rather than after it has been deployed to the web server. The primary reason is that web pages are increasingly dynamic, as we saw with the Vöcklabruck site, and such dynamic content can only be accurately annotated in the content management system used to create it.

5.5.5 Privacy Concerns

Gathering behavioral data of individual users raises privacy concerns (Jameson, 2007), especially, when these data are stored in the Internet on the server-side, and when they can be related to real persons. Amongst others the following scientific papers address privacy concerns: Kobsa (2007); Terveen and McDonald (2005); Cranor (2004); Kobsa (2001).

The ARRIA Design-Time Framework is affected by privacy concerns. The web access logs generated by web servers can pose a threat to the privacy of individuals, and may also present a legal risk to corporations. Therefore, the privacy policy should be taken into account: Tracking users through cookies requires disclosure and consent in advance before their use. The data collected should be handled correctly according to EU privacy legalisation[15,16]. We ap-

[14]Protégé is an ontology editor and knowledge acquisition system (`http://protege.stanford.edu`).

[15]Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the Protection of individuals with regard to the processing of personal data and on the free movement of such data.

[16]Datenschutzgesetz 2000 - DSG 2000, BGBl. I Nr. 165/1999.

proached this issue by providing a privacy statement in web applications which must be confirmed by the individual user. This includes informing the user which data is collected and for what purposes it will be used. For instance, in the e-Government use case we informed users that the web server records their Internet protocol address, the time and duration of their visit, and the time and duration of viewing specific pages on our web site. Even if the user denies the usage of his/her personal data, the web application is able to deliver services, which are not personalized.

5.6 Related Work

The related work discussed in this section focuses on the use of ontologies in AHS and in WUM as well as on WUM in general. We start with a comparison of the ARRIA ontologies to existing ontologies of modeling web applications and public sector online services.

Noy and McGuiness (2001) argue that after determining the domain and scope of an ontology existing ontologies should be considered for reuse. Therefore, we investigated several annotation and e-Government domain ontologies. We examined the NEPOMUK Annotation Ontology Specification that enables users to attach custom descriptions to resources on their desktop (Scerri et al., 2007) and the Photo Annotation Ontology that specifies a domain independent annotation structure (Schreiber et al., 2001). Furthermore, we researched the following two scientific works targeting the classification of e-Government services: Faceted Classification for Public Administration (Rosati et al., 2004) and the Simple Life-Events Ontology in *SU(M)O-KIF* (Bercic and Vintar, 2004). All four ontologies did not meet the intention and scope of the ARRIA ontologies. The NEPOMUK Annotation Ontology Specification as well as the Photo Annotation Ontology have not been built with the scope of making web usage behavior mining more precise. Actually, the same holds for the examined e-Government ontologies. Furthermore, the appraised e-Government classification schemes do not consequently follow a faceted approach to describing public services. After assessing the candidate annotation and e-Government ontologies we decided to build new ones.

In the ARRIA approach we use ontologies for deriving meaningful user models from clickstream data representing the web usage behavior of individual users. In order to take advantage of the domain knowledge, formalized as an ontology, it has to be linked to the web application of interest. This process is called *annotation* of web applications. There are several ways of annotating web applications. First, annotations can be stored separately from the web application in an extra knowledge base, as realized for the personalized e-Government use case (cf. Section 3.1 and Chapter 8). Second, annotations can be embedded directly in

the web application by using microformats[17]. This approach is detailed by Khare and Çelik (2006). We use this approach for the personalized web advertisement use case (cf. Section 3.2 and Chapter 9).

The applicability of ontologies for modeling web users is also demonstrated by Yang et al. (2007). Our semantic user model differs from the description logic (DL) semantic user model for web personalization presented by Yang et al. (2007), as we use a feature-based user modeling approach instead of stereotypes and as the ARRIAs build the user model at run-time directly on the client.

Duc Thanh Tran (2006) suggests the use of ontologies and rules in order to find related content on the web, based on the content currently displayed to the user. We enhance this work by adapting content on the basis of accumulated web usage data. Furthermore, we show how to link semantics and content. Still, the main difference remains the introduction of the autonomous client, as we are dealing with RIAs and not with common dynamic web applications executed on a web server.

Several studies have considered different approaches to integrate content-based semantic knowledge into traditional collaborative filtering and personalization frameworks. An overview of the existing approaches as well as a formal framework for integrating full domain ontologies with the personalization process based on WUM is given by (Dai and Mobasher, 2004). Dai and Mobasher (2002) apply the integration of semantics in WUM techniques to a movie web site. ARRIAs utilize semantic WUM in order to incorporate knowledge stored in ontologies to WUM to find more precise and meaningful adaptation patterns. We enhance related work by considering different web usage mining algorithms and by implementing an alternative way of linking ontologies to web applications.

WUM (Liu, 2007) analyzes log files on the server at certain intervals or possibly in a continuous fashion. It is important, however, to stress that our approach detects events like expanding and collapsing widgets directly on the client. Even when no communication to the server takes place, and, hence, no events can be logged. Thus, our approach extends clickstream analysis to regions which were previously invisible to server-based mining techniques. Moreover, our approach is truly event-driven and we detect events in time. In contrast, traditional mining techniques function in a query-driven manner where results are only created at intervals, such as the daily analysis of log files.

The goal of web usage mining, in particular, is to capture and model web user behavioral patterns (Mobasher, 2004). Indeed, web usage mining is the application of data mining methods to the analysis of recordings of web usage, most often in the form of web server logs. One of its central problems is the large

[17]Definition from `http://microformats.org/`: Microformats are designed for humans first and machines second, microformats are a set of simple, open data formats built upon existing and widely adopted standards.

number of patterns that are usually found: among these, how can the interesting patterns be identified?

A common approach to resolve this problem has been to integrate content characteristics of pages with the user ratings or judgments (Cooley et al., 1997). Generally, in these approaches, keywords are extracted from the content of the web site and are used to either index pages by content or classify pages into various content categories. In the context of personalization, this approach would allow the system to recommend pages to a user, not only based on similar users, but also (or alternatively) based on the content similarity of these pages to the pages the user has already visited.

Keyword-based approaches, however, are incapable of capturing more complex relationships among objects at a deeper semantic level based on the inherent properties associated with these objects. For example, potentially valuable relational structures among objects such as relationships between services, municipalities, and people may be missed if one can only rely on the description of these entities using sets of keywords. To be able to recommend different types of complex objects using their underlying properties and attributes, the system must be able to rely on the characterization of user segments and objects, not just based on keywords, but at a deeper semantic level using the domain ontologies for the objects. For instance, in a traditional personalization system, an e-Government web site might recommend services for passport issuance to a person, simply because that person has previously used or shown interest in this service. On the other hand, a system that has knowledge of the underlying domain ontology, might recognize that the person should first satisfy the prerequisite requirements for a recommended service (e.g. foreigner), or be able to recommend the nearest municipality, and so on.

5.7 Summary

In this chapter we explained our approach of obtaining adaptation patterns from semantically enriched access log files. The ARRIA ontologies were detailed and we demonstrated how to annotate existing web applications in order to instantiate the ontologies. Special attention was given to the semantic preprocessing of the annotations which, together with the reconstructed user sessions from the access log files, serve as input for semantic web usage mining.

As regards annotation, our experience was that after-the-fact annotation of a portal is somewhat time consuming and tedious. So, ideally, the annotation should be created as part of page creation. Given, however, that this is not often the practice, annotation by hand is at least one way to associate pages with concepts. Although, the quality of the mined rules was not overwhelming, such annotations are quite useful. We were able to confirm that annotations can serve

to limit the analysis to sessions or pages associated with specific semantics, for example, certain topics, target groups or types of pages.

Chapter 6

The Adaptation Rule Language

Adaptation Rule Language (ARL) is a general purpose ECA rule language (for details on the ECA paradigm cf. Section 2.6) intended to specify adaptation rules that can be executed in a web browser. ARL is a declarative language, in which for the event, condition and action part the adaptation logic can be expressed declaratively without describing the control flow. We decided to integrate easy access to native RIA functionality as we wanted ARL to be easy to learn by skilled JavaScript programmers and web developers. As JavaScript is already a convenient language for the dynamic adaptation of web user interfaces (WUI) we incorporated it in all parts of our ECA rule language.

We published ARL, our approach to the third research challenge (cf. Section 1.3) in *Scalable Computing: Practice and Experience, Scientific International Journal for Parallel and Distributed Computing, Special Issue: The web on the Move.* (Schmidt et al., 2008e) and presented it at the *3rd International Workshop on Semantic Business Process Management in conjunction with the 5th European Semantic Web Conference* (Schmidt et al., 2008a).

In the course of this chapter we document the formal grammar of ARL step by step. The complete EBNF of ARL is listed in Appendix A. We start by listing and explaining the requirements for ARL in Section 6.1. Also in this section we show how ARL meets the requirements. In Section 6.2 the general structure of an ARL rules file is explained. The subsequent section, Section 6.3, examines the several repositories which are provided by ARL. In Section 6.4 the structure of an adaptation rule is explained. A detailed application of ARL is given by example in Section 6.5 and related work is summarized in Section 6.6. Finally, the chapter is concluded by summarizing its content.

6.1 Requirements

We derived the ARL requirements by analyzing different use cases (cf. Chapter 3). Additionally, we took into account the needs of web developers regarding

the design of ARRIAs. ARL shall enable an easy design and convenient implementation of ARRIAs. Another source which influenced the ARL requirements was the design decision to utilize Ajax as RIA technology for ARRIA's (cf. Section 2.1). Basically, all requirements are intended to raise the acceptance of ARL within the web developer community. Consequently, we set up the following requirements for ARL:

1. ARL shall be a declarative language supporting adaptivity and reactivity.
2. ARL shall be generated by a context-free grammar.
3. The semantics of ARL shall be described.
4. The data exchange format of ARL shall be a lightweight one.
5. ARL shall intrinsically expose access to RIA functionality.
6. ARL shall be extensible.
7. A web developer shall be able to reuse previously defined ARL language artifacts.

6.1.1 Description of the Requirements

Requirement 1. Requirement 1 of declarative language support is defined since adaptation rules shall lack side effects by only describing which adaptations should be performed and not how to compute these adaptations. ARL shall natively support the declarative nature of the adaptation rules discovered during design-time (cf. Section 5.3). As web applications are event-driven ARL shall also provides means to encode complex event expressions.

Requirement 2. Requirement 2 of providing a context-free grammar for ARL aims at the automatic construction of efficient lexers[1] and parsers[2]. A context-free grammar enables the use of lexer and parser generators to automatically generate efficient lexer and parsers.

Requirement 3. The provision of semantics for ARL is the subject of Requirement 3. Semantics reflects the meaning of adaptation rules, and ensures predefined adaptation effects triggered by user interactions.

[1] A lexer is a software that converts a sequence of characters into a sequence of words (tokens).

[2] A parser checks the input tokens created by a lexer for correct syntax, and builds a data structure implicit in the input tokens.

Requirement 4. Requirement 4 aims at the provision of a light-weight data exchange format, since adaptation rules have to be transferred to the client at run-time, and need to be parsed on the client efficiently. The web client must support the interpretation of the ARL data exchange format natively in order to assure efficient rule parsing.

Requirement 5. Requirement 5 ensures the practical applicability of ARL. By providing means for seamlessly accessing core RIA functionalities, like the XMLHttpRequest application programming interface (API), ARL will be easily applicable to web applications that shall expose adaptivity.

Requirement 6. Extensibility is a crucial factor for the applicability of ARL since different application domains might demand, for instance, additional or modified event operators for describing complex event expressions. This is defined as Requirement 6.

Requirement 7. The reuse of language artifacts is the subject of Requirement 7. Reuse of language artifacts allows web developers to save time by reducing redundant work.

6.1.2 Meeting the Requirements

In this section we show how the design of ARL meets the requirements. We start with Requirement 1.

Meeting Requirement 1. ARL follows the declarative ECA rules paradigm. Basically, ARL rules consist of three parts, the event, the condition, and, finally, the action part. The event part enhances the condition and action part of conventional production rules with reactivity. The ECA rules paradigm is explained in detail in Section 2.6.

Meeting Requirement 2. ARL is a domain specific language for the domain of ARRIAs. It is specified as context-free grammar[3] in EBNF. We designed and tested the rule language grammar with the parser generator tool Another Tool for Language Recognition (ANTLR)[4] and its GUI development environment called ANTLRWorks[5]. ANTLR is a sophisticated parser generator for building domain specific languages using a predicate-*LL(*)* parsing mechanism in order to parse

[3] A context-free grammar in contrast to arbitrary grammars exposes only and always a single nonterminal symbol on the left hand side of a production rule rather than a string of terminal and/or nonterminal symbols.

[4] `http://www.antlr.org`

[5] `http://www.antlr.org/works/index.html`

LL(*) conformant grammars like ARL. JavaScript is supported by ANTLR as a target language[6]. This means that ANTLR grammars can be compiled to JavaScript code. ARL is based on the JSON grammar of Taro L. Saito developed in the XerialJ project[7].

Meeting Requirement 3. The semantics of ARL is constituted by the semantics of the event, condition and action part respectively. The semantics of each constituent is defined by reduction to their respective underlying languages. For the event part we rely on the interval-based semantics of SnoopIB (cf. Subsection 2.6.2). For the condition part we use the semantics of comparison operators already defined by JavaScript. The state-changing actions *Assert*, *Modify* and *Retract* adhere to the semantics implemented by common rule engines like Drools[8] or Jess[9].

On top of the component semantics, the overall semantics of ARL defines the relationships between events, conditions and actions. We use an interval-based semantics for ARL based on ECA rules research, e.g., Berstel (2002). An interval-based semantics states that the complete condition of a rule has to be satisfied during the whole detection time of the composite event, i.e. from the beginning of the occurrence of the first constituent event up to the end of the occurrence of its last constituent event. This understanding of ECA rules conforms to the notion of interval-based semantics established for complex events. Interval-based semantics views an event as having a duration, instead of viewing it as an instant at detection time (for more details about interval-based semantics cf. Section 2.6). The duration lasts from the start of the first constituent event to the end of the last constituent event. Therefore, an accompanying condition should span the entire interval of the event duration.

Meeting Requirement 4. RIAs mainly use JSON or XML as data serialization formats for interchanging data (cf. Section 2.1). We decided in favor of JSON as our data and rules interchange format, and against XML, as events, objects and rules have to be easy-to-parse and effortlessly executable on the client-side. We chose JSON as a lightweight data serialization format for ARL because of its simple syntax and client-side readiness. By using JSON, ARL rules can be written by humans with simple text editors and their raw data format is human readable as well. Furthermore, all state of the art browsers support JSON as a subset of JavaScript.

We consider JSON superior to XML, as XML is, in comparison to JSON very verbose, and, thus inflates the payload of HTML requests. By contrast, JSON

[6] `http://code.google.com/p/antlr-javascript`

[7] `http://www.xerial.org/trac/Xerial`

[8] Drools is a business logic platform; `http://jboss.org/drools`.

[9] Jess is a rule engine for the Java platform; `http://www.jessrules.com`.

uses a very lean syntax compared to XML. Tags do not need to be named if, for example, they are just used to provide structure like nesting. Furthermore, XML cannot be executed directly on the client, but has to be parsed beforehand, at an additional expense. Like XML, JSON provides a structured representation of data with deep nesting. Unlike XML it is readily usable in JavaScript because JSON syntax is the subset of JavaScript used to denote object literals and array literals in the programming language. An exhaustive comparison between JSON and XML under the title "JSON: The Fat-Free Alternative to XML" is given by Crockford (2006a), who provides strong arguments for the superiority of JSON over XML.

Meeting Requirement 5. As far as the possible acceptance of a new rule language goes, it is be very important that the language closely fits the environment in which it is to be used. To accomplish this, ARL is easy to deploy in an Ajax environment and honors JavaScript programming practices, where possible. ARL allows the calling of JavaScript functions in the event part, condition part as well as in the action part of adaptation rules.

Meeting Requirement 6. ARL is extensible. By extending its context-free grammar arbitrary language elements can be added, and, in an extra step, implemented. This includes the possibility of adding further operators for the event, condition and action part. Additionally, ARL permits the future use of JavaScript features which are not known today.

Meeting Requirement 7. In addition to extensibility, reusability is supported by ARL. For example, complex event expressions which are repeated in several rules can be made reusable at design-time. Application programmers have the possibility of creating a set of named event expressions. These predefined expressions can be incorporated into further event expressions of different rules. Methods of reuse are also provided for condition expressions and actions. For the latter it is possible to offer a library of predefined actions. UI patterns as defined by Tidwell (2006) might help a web developer to select suitable actions to be reused.

6.2 The Rules File

The rules file is the logical container for all ARL artifacts, and is defined in the following code snipped ranging from Line 020 to 029. The rules file consists of several parts: the prelude and repositories for events, conditions, actions, widgets, and, finally, rules. The prelude part of an adaptation rule is the place to put metadata describing the rule file. The events, conditions, actions and widgets repositories store expressions for later use, like for instance complex event

specifications. These repositories intended to ease the specification of adaptation rules as they contain expressions for reuse in the last and mandatory repository, the rules repository. The rules repository stores the results of the Modeling Cycle (cf. Chapter 5), the adaptation rules.

```
020 rulesFile
021   : LBrace
022       ( prelude Comma )?
023       ( eventsRepository Comma )?
024       ( conditionsRepository Comma )?
025       ( actionsRepository Comma )?
026       ( widgetsRepository Comma )?
027       rulesRepository
028     RBrace
029   ;
```

The Prelude part of a rule file holds all kinds of metadata describing the rule file. Metadata are arbitrary key value pairs where the key is a `String` token and the value an arbitrary but valid JSON expression. Thus, for instance `"Version":"1.0"` is a valid Prelude entry which can be used for the version management of a rule file.

```
031 prelude
032   : '"PRELUDE"' Colon keyValueParams
033   ;
034
035 keyValueParams
036   : LBrace ( keyValueParam (Comma keyValueParam)* )? RBrace
037   ;
038
039 keyValueParam
040   : id Colon value
041   ;
042
043 id
044   : String
045   ;
```

6.3 The Repositories

The Prelude section of a rule file is followed by the following repositories storing items for reuse: the Events Repository, the Conditions Repository, the Actions Repository, the Widgets Repository, and, finally, the Rules Repository. Reposi-

tories are our answer to the non-functional requirement of reusing events, conditions, actions and widgets in different rules. Repository ease the life of application programmers as events, conditions, actions and widgets can be defined once and reused in different parts of adaptation rules. Without repositories enabling the reuse of events, conditions, actions and widgets, they have to be defined anew each time they are used in adaptation rules. Repositories, storing artifacts for later reuse, have not been introduced yet by other rule languages, we are aware of (cf. Section 6.6).

Events Repository. In the Events Repository the application programmer can predefine an arbitrary number of named complex events to be reused in adaptation rules. Event specifications from the Events Repository are referenced in adaptation rules by their name. The Events Repository is defined from Line 047 to Line 053 (cf. Appendix A).

Conditions and Actions Repository. The Conditions Repository and the Actions Repository share the task of the Events Repository namely storing items for reuse. What the Events Repository is for the event part of ECA rules the Conditions Repository and the Actions Repository are for the condition and action part, respectively. In the ARL grammar the Conditions Repository and the Actions Repository are defined from Line 055 to Line 079 (cf. Appendix A). Both repositories share the same grammatical structure except for the items they store. The Conditions Repository stores named conditions and the actions repository stores named actions, as their names would suggest. In contrast to the Events Repository, which allows naming only one single event specification, the Conditions Repository as well as the Actions Repository allows to naming vectors of conditions or actions, respectively. For a named condition array this means that all conditions are related implicitly by the logical AND operator. For a named action array this means that all actions are executed according to their position in the array.

Widgets Repository. Lines 081 to 083 (cf. Appendix A) show the definition of the Widgets Repository in an ARL rule file. The Widgets Repository is meant as a container for arbitrary predefined UI widgets like, for instance, buttons, help windows, wizards, images etc. These widgets can be used in rule actions in order to show, update or hide UI elements. The entries of a Widgets Repository are key value pairs where the key is the name of the predefined widget and the value the widget itself encoded as a valid JSON expression.

Rules Repository The setup of the Rules Repository is defined in Lines 085 to 093 of the ARL grammar (cf. Appendix A). The Rules Repository contains an arbitrary number of adaptation rules. In contrast to the Conditions Repository,

Actions Repository and Widgets Repository, the Rules Repository is more a rules base than a repository storing rules for reuse. The Rules Repository is a container storing the rules encoding the logic for adapting an ARRIA.

6.4 The Adaptation Rule

Starting with Line 095 the structure of adaptation rules is specified. Adaptation rules expose the typical ECA structure plus an optional prelude section. The prelude is a set of key value pairs meant to attach metadata to each individual adaptation rule. Actually, it follows the above defined rules of the prelude for describing metadata for the whole rule file. Metadata could be, for instance, a rule name, a rule version, the name of the rule author, etc.

Adaptation rules consist of optional event and condition parts and the mandatory action part. This leads to the typical structure of ECA rules: `ON` event detection, `IF` all conditions are satisfied, `DO` some actions. In the case that both the event and the action part have been omitted the specified actions are executed only once during the initialization phase. This places web developers in the position to, for instance, assert facts to the working memory (cf. Section 2.5) before the pattern matching network is constructed. So, for instance, the valid adaptation rule `{"DO":["initAction"]}` is called before event detection and pattern network generation is started. After retrieving the rules file the Adaptation Engine (cf. Chapter 4) directly executes the action predefined in the actions repository under the name `initAction`. If it is the only action defined in the rules repository the whole repository would look like this: `"RULES_REPOSITORY": [{"DO":["initActions"]}]`.

```
095 adaptationRule
096   : LBrace
097       ( prelude Comma )?
098       ( eventPart Comma )?
099       ( conditionPart Comma )?
100       actionPart
101     RBrace
102   ;
103
104 eventPart
105   : '"ON"' Colon event
106   ;
 ⋮
206 conditionPart
207   : '"IF"' Colon conditions
208   ;
 ⋮
245 actionPart
246   : '"DO"' Colon actions
247   ;
```

6.4.1 The Event Part

Events are crucial for ARRIAs, as web applications are event-driven by nature. Web applications can react on internal or external events. DOM events[10] or timer events are internal events. They are fired when a user interacts with a web application. External events are signaled to web applications from other systems by pushing them actively to the client or by passively polling them from the server. Thus, for instance, share prices can be signaled to web applications as external events by using server push architectures like Comet (Russell, 2006).

As depicted in Figure 6.1, the event part consists either of a predefined named complex event from the events repository, a simple event expression or a complex event expression. Predefined complex events from the events repository are referred to by their name. Simple events are referenced by their type. Complex events are events constructed by using event operators. Application programmers put complex event expressions in the Event Repository, when they are used more than once.

[10] Pixley (2000) defines DOM events as a platform- and language-neutral interface that gives to programs and scripts a generic event system.

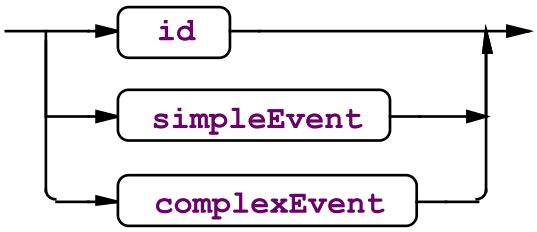

Figure 6.1: Syntax diagram for the event rule.

```
104 eventPart
105   : '"ON"' Colon event
106   ;
107
108 event
109   : id | simpleEvent | complexEvent
110   ;
```

Simple Events

Simple events are either external events from other applications or internal events. We derived this division from analyzing the use cases which were described in Chapter 3. External events can be used in adaptation rules by plainly referring to the appropriate event type like, for instance, politics news from a news ticker. Temporal and DOM events are internal, essential and therefore predefined simple events.

```
112 simpleEvent
113   : eventType | temporal | dom
114   ;
115
116 eventType
117   : LBrace '"TYPE"' Colon String RBrace
118   ;
```

Temporal events are generated by the Adaptation Engine itself and are then directly fed back into the Adaptation Engine. When, for instance, a temporal event specifying a time period is used in an adaptation rule, the Adaptation Engine starts a timer[11] at the specified point in time with the specified duration. When the timer expires, an event is fired which in turn can be detected by the Event Engine of the Adaptation Engine (cf. Chapter 7). A temporal event can define time point, a time duration or both.

[11] A timer is a specialized type of clock used to control the sequence of events.

```
120 temporal
121   : LBrace
122       '"TYPE"' Colon '"TEMPORAL"'
123       ( Comma '"TIME"' Colon String )?
124       ( Comma '"DURATION"' Colon String )?
125     RBrace
126   ;
```

The DOM event expression starting at Line 128 defines the central concept for tracking user clickstreams, and, thus, for personalizing ARRIAs. Any time an adaptation rule relies on a DOM event, the Adaptation Engine registers an appropriate event handler to the web application's DOM. When a relevant event is fired the attached event handler catches the event and feeds it to the event detection component of the Adaptation Engine, which is explained in the next chapter. Thus every interaction with a web application can be tracked.

A DOM event has three attributes: `TYPE`, `SELECTOR` and `EVENT`. The `TYPE` attribute must have the value `DOM`. The selector attribute is an array of strings where each string is a Cascading Style Sheets (CSS) selector. CSS selectors are used to declare which of the DOM elements the event applies to. A description of CSS selectors is given by Bos et al. (2009). The event attribute holds the name of the event of interest, like, for instance, `mouseover` or `blur`. A comprehensive list of DOM events is given by Pixley (2000).

```
128 dom
129   : LBrace
130       '"TYPE"' Colon '"DOM"'
131       Comma '"SELECTOR"' Colon
132         LBracket String ( Comma String )* RBracket
133       Comma '"EVENT"' Colon String
134     RBrace
135   ;
```

Complex Events

In the CEP literature basically two event types are distinguished (cf. Section 2.6): simple events and complex events. Simple events are indivisible, whereas complex event are constituted by other events, either simple or complex. The method of composing complex events is defined by the event operators. Thus, for instance, in our walkthrough example (cf. Section 4.2) a complex event is raised when Mr. Schmidt hovers for the third time over a link in order to read its tooltip which presents a short description of the link's destination. The corresponding complex event specification makes use of the *Sequence* operator in order to define that

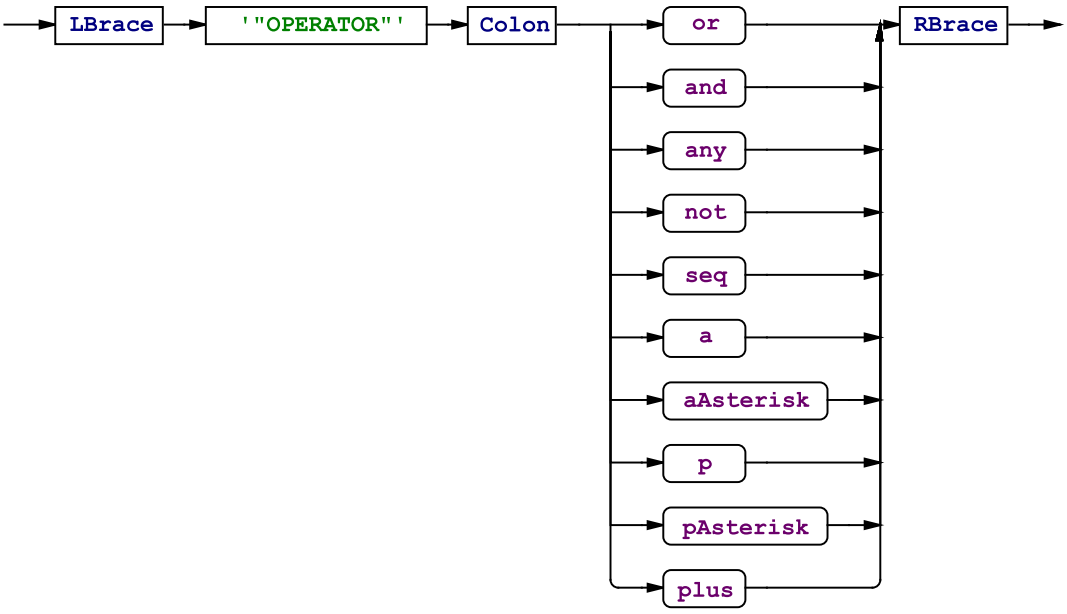

Figure 6.2: Syntax diagram for the complex event rule showing the supported event operators.

when for the third time in succession a tooltip is displayed, an event shall be fired.

Lines 137 to 143 show the EBNF for complex events. Complex events combine named event specifications from the events repository, simple events or complex event specifications by using a combination of the event operators listed in Figure 6.2. The event operators are the common operators from SnoopIB (Adaikkalavan and Chakravarthy, 2006). There are two kinds of operators: logical operators and temporal operators. The interval-based semantics for all event operators is defined by Galton and Augusto (2002).

Logical Operators. The logical operators are *OR*, *AND*, *ANY*, as well as *NOT*. The operators *OR* and *AND* are binary operators in the sense that they involve two operands. The *ANY* operator in a n-ary operator whereas the *NOT* operator is a ternary one.

OR. The *OR* (e_1, e_2) operator states that either of the two specified events must occur for the complex event to occur, for instance, either a *mouse click* or a *key stroke*.

AND. The *AND* (e_1, e_2) operator requests both events to occur before the complex event can be raised, for instance, a *mouse click* and a *key stroke*.

ANY. The *ANY* operator is a generalized form of the *AND* operator. It accepts an arbitrary list of events and a parameter m, which specifies the number of events that must be detected to match the *ANY* pattern. A complex event is detected by the *ANY (M, e_1, e_2, ...)* operator if M of the specified events $(e_1, e_2, \ldots)$ have been detected. Thus, for instance, the following complex event specification *ANY (1, mouse click, key stroke)* fires if a *mouse click* or a *key stroke* event has been detected.

NOT. The non-occurrence of an event in a dedicated time interval is detected by the `Not` operator. The *NOT (e_1, e_2, e_3)* operator raises a complex event if no e_2 event is detected in the time interval specified by e_1 and e_3. For instance, *NOT (start, mouse click, stop)* fires if no *mouse click* has been detected in the time interval delimited by the event occurrences of *start* and *stop*.

Temporal Operators. Additionally, we implemented the following temporal operators: *SEQ*, *A*, *A**, *P*, *P** as well as *PLUS*. The operator *Seq* is the sequence of two events in time. Operators *A* and *A** are ternary operators, detecting occurrences of an event type when they happen within an interval formed by the two other event types. *A** is a variant of *A*. It occurs only once at the end of the interval with all the collected constituents. *A* and *A** are aperiodic event operators because the detected constituents occur at irregular times. *P* and *P** are periodic event operators because of their metronomic characteristics. *P* and *P** are ternary operators as well, they also accept two events starting and ending an interval, but the third parameter is a time expression specifying the interval between the periodic occurrence of events during the given containing interval specified by the first and second event. The *P* operator may be used to collect event parameters for each periodic occurrence. *P** is the cumulative variant which occurs only once, containing all collected constituents. The *PLUS* operator accepts an event type and a time expression. The *PLUS* event occurs after the specified event type has occurred and the specified time has passed.

SEQ. The temporal operators start with the *SEQ (e_1, e_2)* operator. This operator enforces the strict sequence of the specified events, for instance, a *mouse click* followed by a *key stroke*. The constituent events are not allowed to overlap if they are complex themselves and are detected over an interval of time.

A. The *A (e_1, e_2, e_3)* operator signals an aperiodic event each time e_2 is detected within the time interval formed by the other two events. Thus, for instance, the complex event *A (start, mouse click, stop)* is detected each time the mouse is clicked within the time interval defined by the *start* and *stop* event.

A*. The the A^* *(e1, e2, e3)* operator is the cumulative version of the former event operator and is triggered at the end of the interval specified by e_1 and e_3 accumulating all event occurrences (if any) of the event description e_2. Thus, for instance, the complex event A^* *(start, mouse click, stop)* is detected only once at the end of the given interval even if multiple *mouse click* events are detected.

P. The *P (e1, TI, e3)* operator is a periodic event operator which is triggered regularly after the time interval TI. The time interval TI is given in milliseconds. An P event occurs every TI time-steps after an occurrence of e_1, so long as e_3 does not occur. The P operator does not describe the periodic recurrence of some detectable event, but only that a certain period of time has elapsed since a given detectable event (Galton and Augusto, 2002). Thus, for instance, the event description *P (start, 1000, stop)* raises an event each second after the *start* event has occurred and the *stop* event has not occurred. The P operator can be used to force periodic timer events.

P*. The cumulative version of the former event, the P^* *(e1, TI, e3)* operator, is detected at the end of e_3 and accumulates all event occurrences. Thus, the event description P^* *(start, 1000, stop)* raises only one complex event at the end of the time interval delimited by the *start* and *stop* events. The complex event carries the times the timer has expired.

PLUS. The last temporal operator is the *PLUS (e1, TI)* operator. The *PLUS* operator issues a complex event at TI time after the detection of e_1. According to the event description *PLUS (mouse click, 1000)* an *PLUS* event is fired one second after the occurrence of a *mouse click* event.

6.4.2 The Condition Part

In the ECA rule paradigm the condition part is a logical test carried out on objects of the working memory. For an action to get fired, the condition part of an ECA rule has to be satisfied over the entire period of the event detection. This conforms to an interval-based semantics. The condition part of an ECA supports the conjunction of arbitrary conditions. The condition part is started by the `IF` keyword followed by a list of conjunctive conditions.

```
206 conditionPart
207   : '"IF"' Colon conditions
208   ;
209
210 conditions
211   :  LBracket ( condition ( Comma condition )* )? RBracket
212   ;
```

A single atomic condition consists either of a name referencing a predefined condition from the Conditions Repository (cf. Section 6.3), a JavaScript function returning a boolean value or an condition declaratively comparing JavaScript objects and their attributes from the working memory (cf. Section 2.5). When using a JavaScript function, arbitrary code can be executed. This maximizes the possibilities of an application programmer and enables, for instance, access to the DOM or to remote applications by using the asynchronous communication facilities of Ajax (cf. Requirement 5 in Section 6.1).

```
214 condition
215   : id
216     | LBrace
217       ( script | declExpr )
218     RBrace
219   ;
```

A declarative condition consists of at least one of three parts: a relation, a variable or reference to an object, and, finally, a value to compare with. A relation is either one of the predefined relations shown in Line 231, or it is an attribute of an JavaScript object. The JavaScript object might be a member of the working memory but this is not required. It can be an arbitrary JavaScript object having global scope.

The second part consists either of an individual already existing object referred by name or a variable, a placeholder for objects which fulfill the condition. A variable declaration has to start with a question mark. In order to be able to express this special notation of variables we modified the original `String` lexer rules (cf. Lines 363 to 364).

The third part of a declarative condition expresses the value to compare with. A value can again be a variable or it can be a name referencing an object, a name denoting a class or a concrete typed value. The Adaptation Engine recognizes which kind of value is meant by the rule author. Thus, for instance, when using the `instanceOf` (cf. Line 231) relation the value must be a class name.

In order to illustrate what a concrete declarative condition looks like, we consider the following example. The current individual user has visited a page for requesting a marriage certificate. This is modeled in the user model as $hasVisited(?user, ?marriageForm)$. According to the association rules described in Section 5.5 the user might be interested also in requesting a birth certificate. This can be expressed by the following rule, where $?marriageForm$ has the type *MarriageCertificate* and $?birthForm$ has the type *BirthCertificate*:

$$hasVisited(?user, ?marriageForm) \rightarrow isInterested(?user, ?birthForm)$$

The condition part of an adaptation rule is an array containing the two atomic conditions. The `PREDICATE` keyword specifies the property of interest

while SUBJECT and OBJECT specify the variables. The antecedent, that is, body of a rule, has the following ARL syntax:

```
03       [ { "PREDICATE":"hasVisited",
04           "SUBJECT"  :"?user",
05           "OBJECT"   :"?marriageForm" }
06       ]
```

In addition to the three basic parts described above, a declarative condition can expose further optional elements. Thus, a JavaScript function can be given to perform computations on the objects before they are compared (cf. Line 224 and Line 226 in the following listing). Furthermore, in the case of comparing an object property with a data type value, appropriate comparison operators are provided (cf. Lines 227, 240 and 241).

```
221 declExpr
222   : '"PREDICATE"' Colon ( predefinedRelation | String )
223     Comma '"SUBJECT"' Colon ( Variable | id )
224     ( Comma script )?
225     Comma '"OBJECT"' Colon ( Variable | value )
226     ( Comma script )?
227     ( Comma '"OPERATOR"' Colon relationalOperator )?
228   ;
```

In ARL three relations are predefined: instanceOf, sameAs and differentFrom. When using the instanceOf relation in a condition, the Adaptation Engine checks weather or not the first operand is an instance of the class referenced by the second operand. A condition atom using the sameAs relation holds if the first object specified by SUBJECT is the same object as the one specified by OBJECT. An atom {"PREDICATE":"differentFrom","SUBJECT":"x","OBJECT":"y"} holds if x and y are interpreted as different objects. This is in line with the Semantic Web Rule Language (SWRL) (Horrocks et al., 2004).

```
230 predefinedRelation
231   : '"instanceOf"' | '"sameAs"' | '"differentFrom"'
232   ;
```

The SUBJECT attribute as well as an OBJECT attribute can be followed by the definition of a JavaScript function to be executed before the actual comparison takes place. The function takes the specified objects as input parameters, performs any user-defined computation on the inputs, and, finally returns the result.

```
234 script
235   : '"SCRIPT"' Colon
236       LBracket ( String ( Comma String )* )? RBracket
237   ;
```

The relational operators are listed in Line 240 and Line 241. In Line 240 the following relational operators are defined from left to right: equal ==, unequal ! =, greater than or equal >=, greater than >, less than or equal <=, and, finally, less than <. Their semantics correspond to that of JavaScript's relational operators. The operators in Line 241 expose a different semantics as no type-conversion is performed on the operands before the comparison is made. Thus, for instance `( 3 == "3" )` is evaluated to `true` by an JavaScript engine while `( 3 === "3" )` is evaluated to `false` as the string `"3"` is not converted to a number beforehand.

```
239 relationalOperator
240   : '"=="' | '"!="' | '">="' | '">"' | '"<="' | '"<"'
241     | '"==="' | '"!=="'
242   ;
```

6.4.3 The Action Part

The action part defines the actions to be taken when the condition part holds and the triggering event has been detected. Multiple actions can be specified. The `DO` keyword starts the action part.

Actions, for instance, directly manipulate the UI of an ARRIA, change its internal state or raise additional events. The action part might contain one or more JavaScript code blocks to gain a maximum degree of versatility for the rule author. Alternatively, ARL offers to trigger events as well as to manipulate the working memory.

```
244 actionPart
245   : '"DO"' Colon actions
246   ;
247
248 actions
249   : LBracket ( action ( Comma action )* )? RBracket
250   ;
```

Actions already defined in the rules repository are referenced in the rule head. The rule head is the action part of an ECA rule, which is also sometimes called the consequent. An action can be one of the following supported actions depicted in Figure 6.3: script, fire event, assert, modify or retract. With the `SCRIPT` instruc-

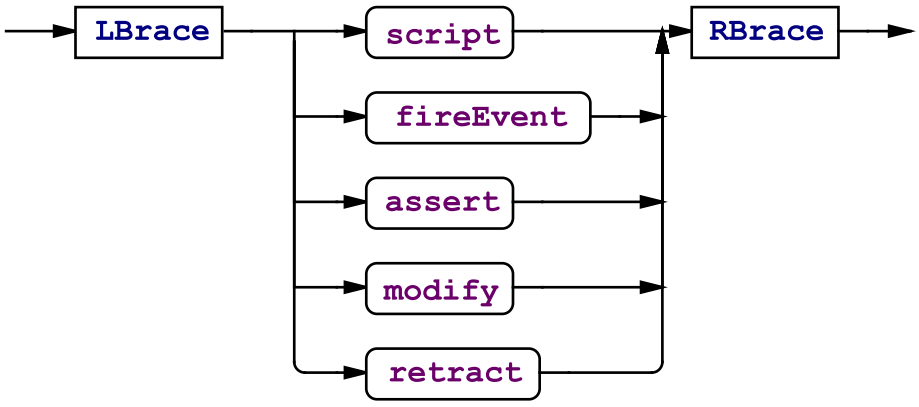

Figure 6.3: Syntax diagram for the action rule showing the supported actions.

tion arbitrary JavaScript code can be executed for a convenient manipulation of the web application.

Within an action the triggering event is reachable via the global `event` variable, just like within conditions. Thus, rule authors may create applications that do calculations on the parameters of the consumed events of the detected complex event triggering this particular action.

```
251 action
252   : id | actionObject
253   ;
254
255 actionObject
256   : LBrace
257       ( script | fireEvent | assert | modify | retract )
258     RBrace
259   ;
```

The `TRIGGER` command can fire an arbitrary event. The parameters of the event can be set by using the `PARAMETERS` key word. Parameters define the attributes of an object by providing key value pairs. The key is the name of the parameter which is set to the value. After an event is issued, it is directly fed back into the Adaptation Engine for further event detection. This allows a rule to trigger other rules.

```
262 fireEvent
263   : '"TRIGGER"' Colon String
264     Comma '"PARAMETERS"' Colon keyValueParams
265   ;
```

Apart from the imperative approach using JavaScript, ARL supports a declarative approach to alter the system state similar to that offered by traditional production systems like OPS5 (Brownston et al., 1985) or CLIPS (Culbert et al., 1993). Thus the working memory can be manipulated by adding and deleting, as well as by modifying business objects.

The last three alternative attributes of an action object `ASSERT`, `MODIFY` and `RETRACT` enable the rule author to manipulate the working memory declaratively. An object can be asserted to the working memory, it can be modified, and, if no longer needed it can be retracted from the working memory.

The `ASSERT` command asserts an object named `id` to the working memory. If the `id` is an empty string the object is assigned to the working memory without a name. The type of the Working Memory Element (WME) to be created is specified by the `CLASS` directive. Optionally, an valid JSON object can be given after the `OBJECT` keyword. Additionally, a JavaScript function can be optionally specified initializing further attributes of the newly created object by, for instance, retrieving data from an application server.

```
267 assert
268   : '"ASSERT"' Colon id
269     Comma '"CLASS"'  Colon String
270     ( Comma '"OBJECT"' Colon jsonObject )?
271     ( Comma script )?
272   ;
```

A WME gets modified by the `MODIFY` command. The modify rule is shown in Figure 6.4. The value of the `MODIFY` attribute is either a variable, an object name or a complex declarative expression. In the case of using a variable or an object id an additional JavaScript function can be given. The JavaScript function is called either on the object or on every object matching the variable declaration during run-time. But, WMEs can also be modified via a complex declarative expression which has been already detailed above. An example demonstrating the usage of the declarative expression for modifying WMEs is given at the end of the section.

```
274 modify
275   : '"MODIFY"' Colon
276       ( ( ( Variable | id ) Comma script )
277         | ( LBrace declExpr RBrace ) )
278   ;
```

The `RETRACT` attribute of a atomic rule deletes the specified objects from the working memory. After asserting, modifying or erasing objects the pattern matching graph is invalidated and has to be recomputed.

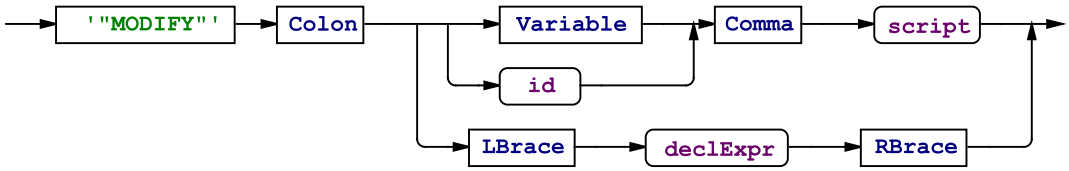

Figure 6.4: Syntax diagram for the modify rule.

```
380 retract
381   : '"RETRACT"' Colon ( Variable | id )
382   ;
```

In order to illustrate the use of a complex declarative expression in order to modify WMEs we come back to the above example. The action part specifies that all `?user` objects meeting the constraints of the condition part are related to all objects `?birthForm` via the `isInterested` property. The following listing shows the complete ARL rule file:

```
01 { "RULES_REPOSITORY" :
02   [ { "IF" :
03       [ { "PREDICATE":"hasVisited",
04           "SUBJECT"  :"?user",
05           "OBJECT"   :"?marriageForm" }
06       ],
07       "DO" :
08       [ { "MODIFY":
09           { "PREDICATE":"isInterested",
10             "SUBJECT"  :"?user",
11             "OBJECT"   :"?birthForm" }
12         }
13       ]
14     }
15   ]
16 }
```

6.5 Construction Wizard Example

In this section an ARL adaptation rule from the e-Government use case (cf. Section 3.1) is given as an example. The exemplary adaptation rule is part of the *Construction Wizard* addressing the most pressing problem of e-Government portals: finding the right form. The *Construction Wizard* directly guides the user to the right form. This kind of adaptation is called direct user guidance and is one adaptation technology as described in Section 2.3.

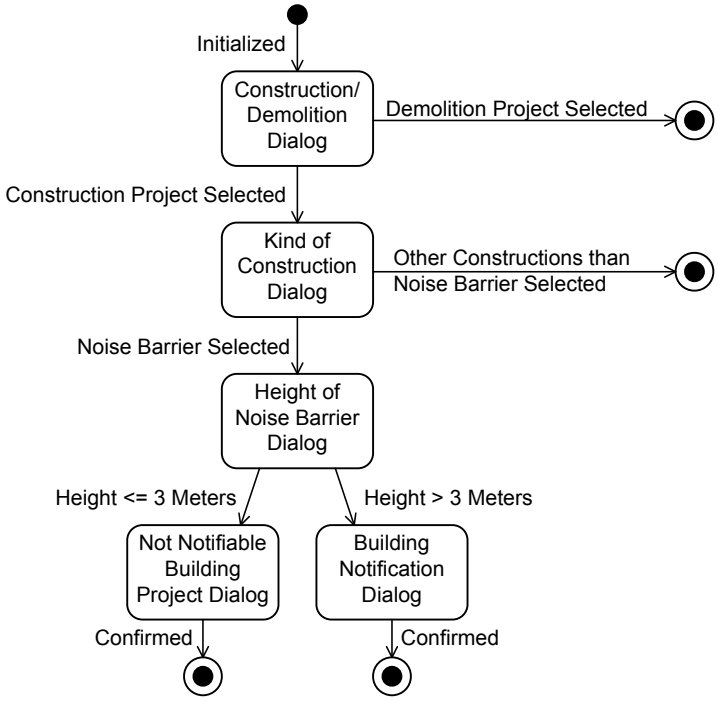

Figure 6.5: Finite state machine of the *Construction Wizard* for the second building rule.

The *Construction Wizard* is based on a question/answer mechanism as known from expert systems in order to assist the user in finding the right form. Actually, the wizard can be seen as a finite state machine where the states represent the choices and the transitions the state changes. The states and transitions are defined in the *Construction Wizard* rules file. The rules file comprises the three building permission scenarios from Subsection 3.1.1. At run time the rules of the rules file guide the user, depending on his/her previous answers, to the right form according to his/her planned building project.

In Figure 6.5 the states and transitions of the third building law rule (cf. Subsection 3.1.1) are depicted, exemplarily for the building rules of the other two building permission process scenarios. After initializing the *Construction Wizard*, a dialog is displayed asking the user whether he/she wants to construct or demolish a building. When the user chooses the construction option the wizard transits to the next stage and displays a dialog asking for the kind of building the user wants to built. As the user plans a to build a noise barrier he/she selects the appropriate item. After that the *Construction Wizards* responds with a dialog asking for the height of the noise barrier. If the user enters a height less than or equal to three meters, he/she gets notified by the wizard that for building a noise barrier with a height less than or equal to three meters no web form has to be submitted. This state of the *Construction Wizard* is depicted in Figure 6.6. Finally, in order to end the wizard, this notification has to be acknowledged by the user.

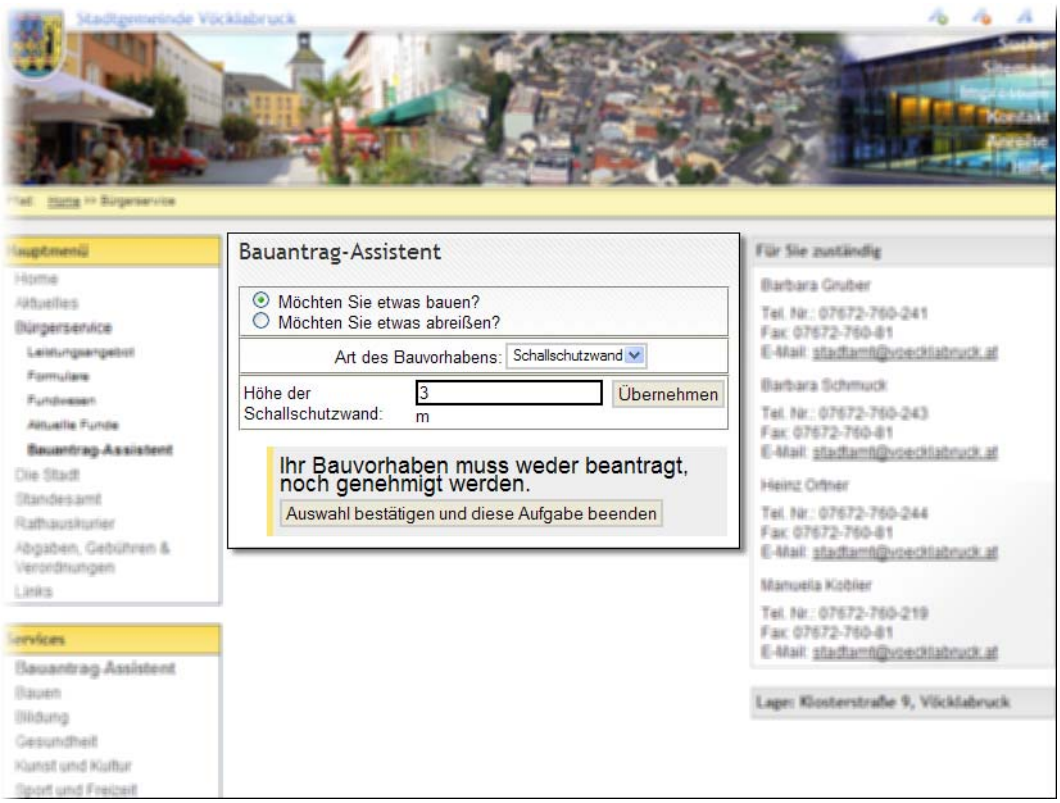

Figure 6.6: The *Construction Wizard* in action. The background is blurred in order to bring attention to the wizard in the middle of the figure.

The following fragment from the rules file details one rule exemplarily. The rule encodes the second transition from the state of waiting for the input specifying the height of a noise barrier (cf. Figure 6.6). The rule gets fired when the user provides a value for the height of the planned noise barrier that is greater than three meters. The action part of the rule instructs the ARRIA to display an additional section advising the user not to apply for a building permit or to submit a building notification.

The rules file starts with a `PRELUDE` section. The `PRELUDE` section contains some key value pairs providing metadata like, for instance, the name of the rule set, its version, its authors, or copyright information. The key value pairs can be arbitrarily defined by the rule author. In our example all repositories except the `RULES_REPOSITORY` are left blank.

```
01 { "PRELUDE" :
02   { "Rule Set Name" : "Construction Wizard"
03   },
04   "EVENTS_REPOSITORY"     : {},
05   "CONDITIONS_REPOSITORY" : {},
06   "ACTIONS_REPOSITORY"    : {},
07   "WIDGETS_REPOSITORY"    : {},
08   "RULES_REPOSITORY"      :
09   [ { "PRELUDE" : { "Name" : "NoiseBarrierRule" },
```

In this example the `RULES_REPOSITORY` consists only of a single rule, the Building Law Rule 3 from Subsection 3.1.1. This rule provides direct user guidance by displaying a hint that the building notification form has to be filled in if the height of a noise barrier is greater than 3 meters. The rule is named `NoiseBarrierRule` in its `PRELUDE` section.

The `NoiseBarrierRule` adaptation rule is an ECA rule with a dedicated event, condition and action part. We designed one triggering event for all adaptation rules of the *Construction Wizard*. This uniform trigger event is originally saved in the `Events_Repository`. In this example for the sake of clarity the trigger event is defined in the event part of the `NoiseBarrierRule`.

The idea of one uniform trigger event for all rules it that it can be defined once and used everywhere. Therefore, an complex event is specified that fires for any input in any HTML element. The event specification starts in Line 10. The complex trigger event is raised when any of the following three simple DOM events are detected: `blur`, `change` or `click`. These events are fired on different HTML elements, but all three events signal a new user input. The `blur` event is fired on special HTML input elements like single-row or multi-row input fields encoded in HTML as `<input type="text">` or `<textarea>`, respectively. The `change` event is raised on `<select>` elements like list or combo boxes when the user selects a new item. The `click` event signals that a radio button with the HTML signature `<input type="radio">` or check box with the HTML signature `<input type="checkbox">` has been clicked by the user.

The logical `ANY` operator is used in order to specify the trigger event. In Line 11 the kind of operator is set to `ANY`, and the number of events that have to be detected before the complex trigger event is raised is set to one by using the `M` parameter. This means, that whenever one of the constituent events is detected, the complex event is also fired. The constituent events are the three `DOM` events described above. Each of the three `DOM` events describes different events and event sources within the *Construction Wizard*. The constituent events are listed after the `CONSTITUENT_EVENTS` parameter in the following order: `blur` event, `change` event, and, finally, `click` event.

The first simple event specification is expressed from Line 14 to Line 20. The Adaptation Engine assigns an event handler (cf. Section 2.6) for the `blur` event

to all elements matching the CSS selectors in Line 16 and Line 17. This means, that whenever one of the specified elements loses its focus the corresponding `blur` (cf. Line 19) is caught and fed into the Event Engine, which is a component of the Adaptation Engine (cf. Chapter 7). The event handlers are assigned to the selected elements when the rule file is compiled and the event detection graph is built. The `SELECTOR` attribute of the first constituent event is made up of two CSS selectors. The first selector defined in Line 16 selects all single row input fields of the *Construction Wizard*, while the second selector in Line 17 selects all multi-row input fields for processing their focus lost events.

The second constituent specification of the complex `ANY` event occupies Lines 21–26. It defines the `change` event for all kinds of list boxes the *Construction Wizard* might feature. The `change` event is also an DOM event, and is signaled when the selection of, e.g, a list box has changed.

Finally, the third specification of a simple constituent DOM event spans from Line 27 to Line 33. It describes a mouse click on either a check box or a radio button.

```
10        "ON" :
11        { "OPERATOR" : "ANY",
12          "M"        : 1,
13          "CONSTITUENT_EVENTS" :
14          [ { "TYPE"     : "DOM",
15              "SELECTOR" :
16              [ ".ConstructionWizard input[type=text]",
17                ".ConstructionWizard textarea"
18              ],
19              "EVENT"    : "blur"
20            },
21            { "TYPE"     : "DOM",
22              "SELECTOR" :
23              [ ".ConstructionWizard select"
24              ],
25              "EVENT"    : "change"
26            },
27            { "TYPE"     : "DOM",
28              "SELECTOR" :
29              [ ".ConstructionWizard input[type=checkbox]",
30                ".ConstructionWizard input[type=radio]"
31              ],
32              "EVENT"    : "click"
33            }
34          ]
35        },
```

The condition part of the example rule evaluates the state of the *Construction Wizard*. Only if the *Construction Wizard* is in the state of *Height of Noise Barrier Dialog* (cf. Figure 6.6), and a value greater than three meters is filled in, will the action fire. The state of the *Construction Wizard* is persisted in the object `conWiz`.

```
36       "IF" :
37       [ { "PREDICATE" : "state",
38           "SUBJECT"   : "conWiz",
39           "OBJECT"    : "HeightOfNoiseBarrierDialog",
40           "OPERATOR"  : "=="
41         },
42         { "PREDICATE" : "value",
43           "SUBJECT"   : "conWiz",
44           "OBJECT"    : "3",
45           "OPERATOR"  : ">"
46         }
47       ],
```

Finally, the action part of the adaptation rule is defined. The action part describes the actions to be taken after an input for the noise barrier's height of greater than three meters is provided. The action part defines two actions. The first action changes the state of the *Construction Wizard* to the new *Building Notification Dialog* state, by setting the `state` property of the `conWiz` object to the value `BuildingNotificationDialog`. The second action, listed in Line 55, calls a JavaScript function which displays a link to the building notification dialog.

```
48       "DO" :
49       [ { "MODIFY" :
50           { "PREDICATE" : "state",
51             "SUBJECT"   : "conWiz",
52             "OBJECT"    : "BuildingNotificationDialog"
53           }
54         },
55         { "SCRIPT" : [ "showBuildingNotificationDialog();" ]
56         }
57       ]
58     }
59   ]
60 }
```

6.6 Related Work

We compare several reaction rule languages and position our approach with respect to them. The approaches under review are: the ECA-Web language suggested by Daniel et al. (2007), the event processing language for the web presented by May et al. (2005), the rule-based event processing language for XML events proposed by Bry and Eckert (2007a), and, finally, Reaction RuleML elaborated by Paschke et al. (2007).

The ECA-Web language suggested by Daniel et al. (2007) is an enhanced XML-based event condition action language for the specification of active rules, conceived to manage adaptiveness in web applications. Our ARL is different to that approach as we, as stated in the name, rely on JSON as exchange and execution format. Moreover, we incorporated an event algebra for specifying complex events based on Snoop. Besides that, the whole adaptation approach is quite different as we support punctual adaptation directly on the client compared to the server-side adaptation and rule execution approach of ECA-Web.

Another event processing language for the web is presented by May et al. (2005). It is likewise an ECA rule-based approach, but with pluggable language dialects for each of the E, C and A parts of a rule. An ontology of the compositional approach is presented. The question of connecting event sources is addressed in this work, but requires a degree of cooperation of nodes on the web which is currently not practical. For example, a possible source of events are changes to XML data. However, such events are only created if changes are monitored, e.g. with the help of an active XML database. As a workaround, so-called *ECA services* are proposed which provide active notifications from passive nodes. In contrast to the ARRIA approach their solution requires polling/querying and as such is strictly not event-driven. Our solution actively publishes events when they occur and as such is fully event-driven. In a federated setup of the mentioned related work, our solution could possibly be used as a source of events.

Bry and Eckert (2007a) describe a rule-based event processing language for XML events focusing on aspects of data extraction, event composition, temporal relationships and event accumulation. The approach is based on logic programming. Unfortunately, a few drawbacks are inherited from this which prevented their use in ARRIAs. The most striking fact is that events (simple and complex) are detected and reacted to in a query-driven fashion. This means that event patterns are only fulfilled when the query engine asks for the patterns. There is no data-driven way of fulfilling patterns as each event arrives. This behavior is based on the fact that logic programming systems such as Prolog (Demoen, 2005) operate in a backward-chaining way, fulfilling queries only when they are posed. There is no built-in notion of continuous queries. This means that the approach by Bry and Eckert (2007b) as well as other logic based approaches such as Paschke et al. (2007) are not truly event-driven, because events are not handled

when they occur but are stored until a query is posed. In comparison to our work the approaches based on logic programming are pure server-side realizations.

Paschke et al. (2007) proposes Reaction RuleML as an XML-serialized rule language incorporating different kinds of productions and complex event messaging. As Reaction RuleML is also intended as a rule interchange format, it is very broad and verbose. ARRIAs's Adaptation Rule Language is conceptually simpler than Reaction RuleML and uses a different serialization format that ensures efficient execution on the client-side.

6.7 Summary

In this section we specified ARL, a new language for programming adaptive web applications. We detailed the context-free grammar step by step by illustrating each part of ARL: the prelude, the repositories, and, finally, the adaptation rules themselves with their event, condition and action parts. This new declarative language is tailored to the needs of being executed directly on the client in order to provide seamless adaptations to user interactions or other external events. To the best of our knowledge this is the first ECA rule language relying on JSON as the data exchange format.

Chapter 7

The Adaptation Engine

In this chapter we detail the core component of the ARRIA Run-Time Framework: the Adaptation Engine (cf. Chapter 4). The Adaptation Engine is responsible for tracking the user, building the user model, and, based on the evaluation of the adaptation rules, which were described in the previous chapter, adapting the dynamic UI of the web application.

The ARRIA Adaptation Engine is our approach to meet Research Challenge 4 (cf. Section 1.3). The challenge was to design and implement an efficient and scalable client-side adaptation engine which is capable of executing adaptation rules encoded in ARL. In order to meet this challenge we designed the Adaptation Engine based on an efficient algorithm which evaluates adaptations expressed in ARL (cf. Chapter 6) on the client-side. To our knowledge, ARRIAs are the first systems where production engines are used in web applications on the client-side for computing personalizations in adaptive web applications. One major achievement of ARRIAs is a new way of extending the Rete algorithm with CEP capabilities. Furthermore, we tailored the production algorithm as well as the CEP algorithm, to the requirements imposed by the JavaScript-based, client-side Adaptation Engine.

We presented the concepts of the ARRIA Adaptation Engine at the *11th International Conference of Business Information Systems* (Schmidt and Stojanovic, 2008), at the *1st International Workshop on Complex Event Processing for the Future Internet collocated with the Future Internet Symposium* (Schmidt, Stühmer, and Stojanovic, 2008d) and at the *2009 AAAI Spring Symposium on Intelligent Event Processing* (Schmidt, Stühmer, and Stojanovic, 2009b).

The chapter is structured as follows. First, in Section 7.1 we give the requirements for the design of an client-side Adaptation Engine, and we explain, on the conceptual level, the architecture of the Adaptation Engine by using SAP's standardized TAM (Gröne, 2008a,b). After that, in Section 7.2 we give insight into the design level architecture and our novel algorithm of combining event processing with production processing. Subsequently, we explain in Section 7.3 how the internal data structures of the Adaptation Engine are constructed. Afterwards,

in Section 7.4 we describe how we analyzed the performance of the internal data structures of the Adaptation Engine. In Section 7.5 we discuss related work, and, finally, summarize the whole chapter in Section 7.6.

7.1 Concept Level Architecture

The requirements for the design and implementation of the Adaptation Engine are derived from the client-side solution approach which is derived from the state of the art gap analysis and use case analysis (cf. Chapter 1 to Chapter 3). The requirements for the Adaptation Engine are:

1. The Adaptation Engine shall be pluggable into existing web applications.
2. The Adaptation Engine shall be capable of executing declarative adaptation rules written in ARL.
3. The Adaptation Engine shall provide reactivity by efficiently processing user interactions and events from other event sources.
4. The Adaptation Engine shall provide timely adaptivity by efficently processing productions.

How we meet these requirements is depicted in Figure 7.1 on a conceptual level. ARRIAs have a clear component-based and modular architecture. In general, ARRIAs consist of the domain specific Web Application and of the universal Adaptation Engine. The Adaptation Engine is plugged into the Web Application. The Web Application might be a plain web site or a sophisticated RIA. Nevertheless, the Web Application has to expose some kind of dynamic, programmable UI like DHTML[1] in order to be adapted by the Adaptation Engine. Thus, the run-time architecture of ARRIAs embraces the Web Application and the Adaptation Engine. Treating the original Web Application and the Adaption Engine separately enables us to universally plug the ARRIA into any legacy web application, as demanded by Requirement 1.

To meet Requirement 1 we chose Ajax and JavaScript as RIA technologies upon which to base the implementation of the Adaptation Engine. This frees us from installing a proprietary RIA plug-in as required by other RIA technologies like, for instance, Silverlight (cf. Section 2.1). By directly executing JavaScript code the Adaptation Engine can manipulate the UI of the application on time by manipulating the DOM.

An ARRIA is adapted by the Adaptation Engine. The tracking of user interactions, the building of the user model, the evaluation of the adaptation rules,

[1] DHTML is used to animate interactive web sites and exposes compared to static web sites enhanced functionality leveraging sophisticated UI effects.

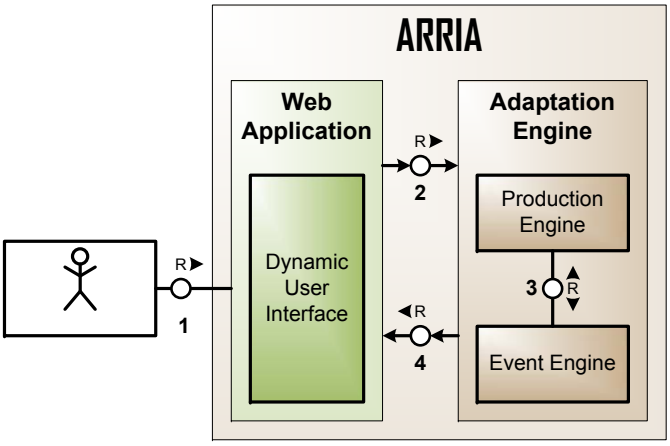

Figure 7.1: Conceptual level architecture of ARRIAs.

and, finally, the adaptation of the dynamic UI of the Web Application is the responsibility of the Adaptation Engine. In order to perform this task it executes ARL adaptation rules, according to Requirement 2. The Adaptation Engine conceptually consists of two active components[2]: the Production Engine and the Event Engine.

7.1.1 The Production Engine

The Production Engine checks whether an ARRIA requires any user-centric adaptations. The adaptation need is directly inferred from the objects stored in the working memory of the Production Engine (cf. Section 2.5). The objects of the working memory are mainly related to the user profile, but, actually, they can be arbitrary objects of the underlying implementation language, JavaScript in our case. The logic for processing these objects is encoded in ARL adaptation rules. Also, the actions which have to be taken, after a successful rule evaluation of both the event and the condition part, are declaratively defined by ARL adaptation rules. To efficiently process productions, Requirement 4, we implemented an object-oriented Rete algorithm in JavaScript. The Rete algorithm was first introduced by Forgy (1982). Rete has been chosen because of its outstanding performance characteristics, as described in Subsection 2.5.1.

The Rete algorithm used in our prototypical implementation of the ARRIA Run-Time Framework remedies the reevaluation of the entire knowledge base by introducing time-efficient pattern matching algorithms, at the expense of space. The algorithm saves the results of intermediate condition evaluation steps. Thus, whenever the working memory changes, the condition evaluation need not be fully

[2]Active components are capable of doing a certain action. In contrast, passive components are components upon which active components can act. Passive components are usually stores containing data of any kind.

recomputed. This is accomplished by dividing the conditions into a hierarchical network of nodes, each doing a single comparison, filter operation, join, etc. Each node has a memory which stores the objects that fulfill the constraints of this node. When an object is changed, the network does not need to be completely recomputed, rather only the changed object is re-fed into, or removed from the network.

The Rete Network $RN = (RV, RE)$ is a DAG (cf. Subsection 2.5.1) where RV are vertices representing object lists and RE the edges between them. Also the RN consists of four node types: Root Node, Alpha Node, Beta Node, and, finally, Terminal Node.

Root Node. The Root Node is the common entry point for all objects from the working memory (cf. Subsection 2.5.1). There is only one Root Node in the whole Rete Network. Objects are fed into the Rete Network implemented by the Production Engine, and they are subsequently distributed to an adjacent Alpha Node.

Alpha Node. An Alpha Node selects individual objects based on simple conditional tests matching object attributes against constant values. Alpha nodes are one-input nodes accepting only a single object at a time. Alpha Nodes are subclassed by *Type Nodes* and *Selection Nodes*. The immediate child nodes of the Root Node are Type Nodes. Type Nodes test the types, that is, the classes of a given object. Objects not matching any Type Node are not considered by the Rete Network. The second Alpha Node subclass is the Selection Node. Selection Nodes compare one or more attributes of the same object. If an object is successfully matched against the condition represented by an Alpha Node it is passed to the next node. This might be again an Alpha Node or a Beta Node. Objects successfully matched against all conditions of a branch are stored in the Alpha Memory (cf. Subsection 2.5.1).

Beta Node. A Beta Node performs tests on different objects by joining them. Beta Nodes are two-input nodes, which store the results of the join operation as a set of object lists in their associated memories called Beta Memories. There are two specialized Beta Nodes implemented in the ARRIA Production Engine: *Join Nodes* and *Adapter Nodes*. Join Nodes have two-inputs, a *left* and a *right* input. The left input is the Beta Memory from an antecedent Beta Node and the right input is the Alpha Memory from an Alpha Node branch. The Join Nodes performs joins between object lists from the Beta Memory and individual objects from the Alpha Memory. The second subclass of the Beta Node type is the Adapter Node. Adapter Nodes are used in the Rete Network when there is no left input from antecedent Beta Nodes, since they are the first Beta Nodes of the Beta Network. Adapter Nodes enable the treatment of Alpha Memories as

the left input of a Beta Node. The output of a Beta Node is either sent again to an adjacent Beta Node or to an Terminal Node, when the end of a branch is reached.

Terminal Node. Terminal Nodes represent a complete match of a single production. For each object list that arrives at a terminal node, an action instance will be fired.

7.1.2 The Event Engine

The Event Engine tracks the user according to the logic encoded by the ARL adaptation rules. We implemented all the event operators of ARL (cf. Chapter 6). Internally, the user actions are tracked and a user model describing the working context of the user is derived. The Event Engine is able to detect meaningful events, where *meaningful* means events that might indicate a situation in which the user needs support or guidance, e.g., in a *lost in hyperspace* situation (cf. Section 2.3).

In order to implement the client-side Event Engine we had to choose from two different alternative approaches: forward-chaining or backward-chaining event detection. We decided upon an active forward-chaining and event-driven algorithm, since backward-chaining and query-driven algorithms restrict the capacity to act to the intervals at which queries are issued. The advantage of forward-chaining evaluation is that for each change of state affecting a condition, the partial match is saved until it can be further extended to eventually form a complete match. Complete matches are reported immediately when they occur. Since events happen asynchronously and are generally not predictable by nature, forward-chaining algorithms actively push new events into an appropriate data structure that reactively detects complex events. After making the decision in favor of a forward-chaining algorithm, we now had to choose from different, alternative event-driven algorithms (cf. Section 2.6).

From the available approaches we chose and modified a graph-based event detection algorithm based on the work of Chakravarthy (1997) and Adaikkalavan and Chakravarthy (2006) as a basis for efficient event detection in order to meet Requirement 3. The forward-chaining event detection algorithms differ in their detection approach. They use either automata (Gehani et al., 1992a, 1993), Petri-nets (Jensen, 1992; Gatziu and Dittrich, 1994) or a graph-based approach (Chakravarthy et al., 1994). The reason for choosing a graph-based approach over the other detection methods is that the graph-based approach allows the detection of overlapping complex events while still offering a clear strategy for the consumption of the constituent events.

The Event Engine implements a modified Event Detection Graph as introduced by SnoopIB (Adaikkalavan and Chakravarthy, 2006). The Event Detection Graph is a DAG with nested complex events being parents of their less deeply

nested sub-events, down to the leaf nodes which are simple events. Detected events are propagated upwards in the network, starting with simple events which are fed into the graph at the simple event nodes. The propagation ends at top nodes which have no further parents.

Event nodes may have more than one parent. This occurs when an event expression is used in several places in event detection patterns. The reused expression is then manifested only once in the event graph but outgoing edges are linked to all nodes where the expression is reused. All parent nodes are informed of detected events. The sharing of common subtrees by more than one parent saves space and time compared to detecting the same sub-events multiple times.

The Event Detection Graph $EDG = (EV, EE)$ is a DAG, where EV are the event vertices and EE the edges between them. The EDG consists of four node types: Event Bus Node, Simple Event Node, Complex Event Node, and, finally, Conclusive Node.

Event Bus Node. The Event Bus Node is the entry point, the root node of the Event Detection Graph. The Event Bus Node dispatches the event to its corresponding Simple Event Nodes. If there are no Simple Event Nodes of the expected simple event type the event is discarded.

Simple Event Node. A Simple Event Node is a one-input node specifying an event type. It receives only events which are of the specified type. A Simple Event Node propagates the events of its type to its adjacent Complex Event Nodes.

Complex Event Node. A Complex Event Node is an n-ary input node corresponding to an ARL event operator like, for instance, *SEQ* (cf. Section 6.4 for the complete list of event operators defined by ARL). The Complex Event Node holds all events matching the operator's semantics. Complex Event Nodes can be succeeded by other Complex Event Nodes or by Conclusive Nodes.

Conclusive Node. A Conclusive Node represents a complete set of events matching the complex event specification of an ARL rule. Each event set arriving at a Conclusive Node triggers a new action instance.

7.1.3 Walkthrough of the ARRIA Adaptation Cycle

While a user interacts with a web browser, the browser produces events, which are called DOM events, in response to these interactions. In ARRIAs the user is tracked by the Event Engine by means of the DOM events (cf. step 2 in Figure 7.1) in order to recognize opportunities to adapt the WUI to the current user context. Triggered by a meaningful event detected by the Event Engine or a state change of the Working Memory the Production Engine evaluates the

adaptation rules taking into account the state of the entire application (cf. step 3 in Figure 7.1). In case at least one adaptation rule is applicable, the WUI is adapted according to the inferred user interests (cf. step 4 in Figure 7.1). As long as the user continues browsing, he/she is tracked, and user interface adaptation occurs as necessary. This procedure lasts until the user decides to end the session by closing the ARRIA.

7.2 Design Level Architecture

The design level run-time architecture of the Adaptation Engine is depicted in Figure 7.2. It refines the conceptual run-time architecture of ARRIAs (cf. Figure 7.1), details the system building blocks and shows the communication and data flow between the several components. We explain the design level architecture with the help of the Construction Wizard example given in Section 6.5. The design level architecture of the Adaptation Engine exposes the following active components: Graph Builder, Event Bus, Event Sources, Hybrid Inference Engine, Agenda, and, finally, the Dynamic User Interface/DOM. Additionally, the following passive components are depicted in Figure 7.2: Rules File, Fact Base, Working Memory and the Hybrid Discrimination Network with its subnetworks Rete Network and Event Detection Graph. In the following subsections the different active and passive components are detailed. Furthermore, the interactions between these components are described.

7.2.1 Graph Builder, Rules File and Fact Base

The Graph Builder converts the ARL adaptation rules stored in the Rules Files into the internal data-structures of the *Hybrid Inference Engine* by dissecting the adaptation rules. Furthermore, the Graph Builder loads the objects from the Fact Base into the Working Memory.

Rules File

The ARL adaptation rules are stored in the Rules File. The structure of such an rule file is defined by ARL (cf. Chapter 6). In fact the Adaptation Engine can handle an arbitrary number of Rules Files and Fact Bases. In Section 6.5 the exemplary rule file for the Construction Wizard is given.

Fact Base

The objects contained in a Fact Base specify the initial state of the application. Such an object is, for instance, the `conWiz` object in the Construction Wizard example. The state of the `conWiz` object and hence of the Construction Wizard is initially set to the state *Construction/Demolition*. The objects of the Fact Base

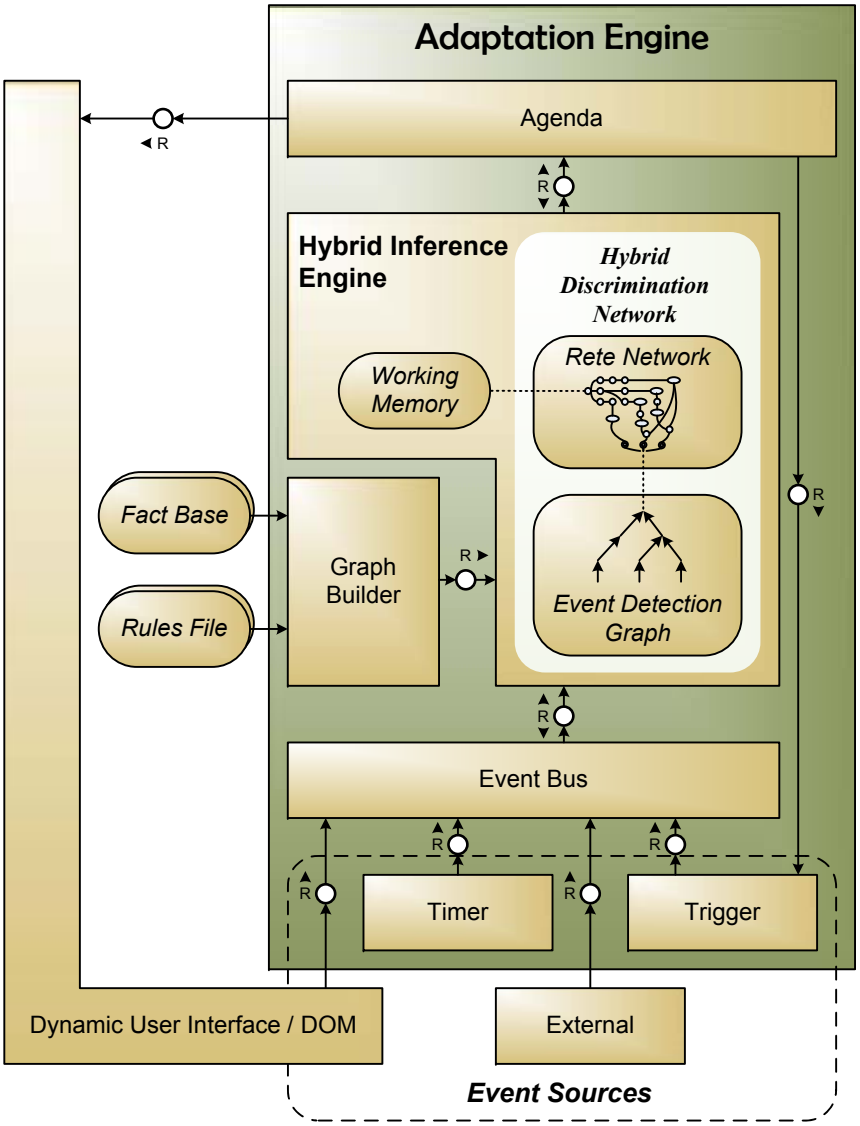

Figure 7.2: Design level run-time architecture, client-side.

are JavaScript objects, since JavaScript objects can be directly manipulated at run-time by the Adaptation Engine which, in turn, is also written in JavaScript.

The Fact Base might also contain web site annotations if there are any. As detailed in Section 5.2 annotations describe the topic of web objects. They are stored in an OWL ontology called the *Knowledge Base* which is constructed in the Modeling Cycle. After the materialization[3] of the Knowledge Base, its concepts, relations and individuals can be stored in the Fact Base as JavaScript objects.

[3]Materialization is the evaluation of inference rules in advance. Again, the results of this process are stored in the ontology. In other words, materialization is the explicit and in advance extraction of knowledge which is implicitly modeled in an ontology.

This step is necessary, as ARRIAs do not reason about OWL/XML ontologies on the client. If there are future use cases which require direct client-side reasoning the Production Engine and its Rete algorithm can also be used as an inference engine.

Graph Builder

The Graph Builder constructs both graphs of the Hybrid Discrimination Network: the Rete Network and the Event Detection Graph. For the event part, the event operators are turned into nodes of the Event Detection Graph, and the conditions are transformed into the object-oriented Rete Network. The objects of the Fact Base are transformed into WMEs. How both graphs of the Hybrid Discrimination Network are constructed is detailed next.

Building the Rete Network. The Rete Network $RN = (RN, RE)$ is a subgraph of the Hybrid Discrimination Network HDN with $RN \subseteq HDN$. In Section 2.5 we already explained how the Rete algorithm works and in Section 7.1 we described the different nodes types the Rete Network is constructed of. Now we focus on how ARL rules are converted into the graph structure of the Rete Network. The construction of the Rete Network from ARL rules is depicted in Algorithm 2. First, the Alpha Network is constructed.

Each condition found in an ARL rule checking the attributes of a single class is converted into a series of linked consecutive Alpha Nodes (cf. Subsection 7.1.1). These Alpha Nodes perform checks on the class type of objects or test attributes against constant values. After the single object checks are completely represented in the network, a memory is added to each Alpha Node. These memories are called Alpha Memories.

Beta Nodes are constructed whenever the condition involves two objects or object variables. Whenever a new Beta Node is the first node in given branch an Adapter Node is created, which accepts two Alpha Memories as input. If the Beta Node is not the first node in a branch a Join Node is created which accepts as left input a Beta Memory and as right input an Alpha Memory. Furthermore, the newly created Beta Node is linked to its predecessors.

Building the Event Detection Graph The Event Detection Graph $EDG = (EV, EE)$ is a subgraph of the Hybrid Discrimination Network HDN with $EDG \subseteq HDN$. The construction of the Event Detection Graph is illustrated in Algorithm 3.

We apply a modified event detection graph based on the work presented by (Adaikkalavan and Chakravarthy, 2006). Such an event detection graph is a DAG, and is constructed as follows: All simple events are represented as Simple Event Nodes at the bottom of the graph. There is one simple event node per simple event type. Complex event expressions are at the top of the graph. They are

Algorithm 2: Construction of the Rete Network.

```
allocate Root Node
for all rules files do
  for all rules do
    for all conditions do
      if number of objects == 1 then {Alpha Node}
        if type testing condition then
          allocate Type Node
        end if
        if constant testing condition then
          allocate Selection Node
        end if
      end if
      if number of objects == 2 then {Beta Node}
        if first Beta Node in a branch then
          allocate Adapter Node
        else
          allocate Join Node
        end if
      end if
    end for
    allocate Terminal Node
  end for
end for
```

represented by Complex Event Nodes. Descending the Event Detection Graph the event expressions becomes less complex culminating in the indivisible Simple Event Nodes at the bottom. Common sub-expressions needed by more than one ARL rule are shared between the nodes. Simple events coming from the Event Sources are fed into the event detection graph at the bottom. As they propagate through the connected nodes, the events are either discarded, queued or propagated further, if they match an event expression.

7.2.2 Hybrid Inference Engine

The Event Engine and Production Engine, which are separated at the conceptual level, are combined into a single engine at the design level. We call it the *Hybrid Inference Engine*. We combined standard event detection and pattern matching algorithms into an efficient, object-orientated algorithm implemented in JavaScript. Internally, we combined the Rete Network and the Event Detection Graph into a single *Hybrid Discrimination Network* which ensures the adequate performance of adaptations in ARRIAs.

Algorithm 3: Construction of the Event Detection Graph.

```
allocate Event Bus Node
for all rules files do
  for all rules do
    for all event descriptions do
      if SimpleEvent then
        allocate Simple Event Node
      else
        allocate Complex Event Node
      end if
    end for
    allocate Conclusive Node
  end for
end for
```

Hybrid Discrimination Network

The Hybrid Discrimination Network is a newly researched and newly developed data structure which combines production processing and event processing. Formally, the Hybrid Discrimination Network HDN is a DAG and defined as $HDN = (V, E)$. V are the vertices, the nodes of the Hybrid Discrimination Network and E the edges between them. The Hybrid Discrimination Network is depicted in Figure 7.3.

Although, the Rete Network and the Event Detection Graph seem to be similar at first glance, we decided not to merge them but to keep them separated by only combining them at their edges. Both, the Rete Network and the Event Processing Graph are DAGs, they are used for incremental pattern matching, in a bottom-up fashion, both are state saving algorithms which do not recalculate previous matches, and which reuse partial matches shared by more than one rule.

But, the Event Detection Graph and the Rete Network process semantically different information. Events in the Event Detection Graph are transient and are deleted automatically after they are delivered to all consumers. Facts in the Rete Network, on the other hand, must persist until they are explicitly deleted. Events are treated as immutable, whereas facts can be changed. Another analogy from Bry and Eckert (2006) compares facts to written text. Text may be altered or deleted, it does not expire by itself and it is available for anyone to retrieve it. Events, on the other hand, are like the spoken word. They are available only to the people who listen at the right time, and are immutable.

In order to distinguish non-temporal facts from temporal events we implemented the Hybrid Discrimination Network as two interconnected graphs in which facts and events are processed separately. We combined the graphs of the Rete Network and the Event Detection Network at their terminal nodes by introducing

a new node type common to both graphs: *Liaison Node*. Figure 7.3 depicts a sample graph where the Rete Network is joined with the Event Detection Network through the newly introduced Liaison Nodes in the middle of the figure. The Rete Network, depicted at the top, is responsible for the evaluation of the condition parts of ARL rules, and for finding all objects matching the rule conditions. The several node types of the Rete Network were already described in Section 7.1. The Event Detection Graph, at the bottom, is responsible for detecting events specified in the event part of ARL rules. The different node types of the Event Detection Graph are also described in Section 7.1. The Event Detection Graph detects all simple or composed events matching the predefined event patterns. Both the Rete Network and the Event Detection Graph propagate their results to the Liaison Nodes. Finally, the rule actions are triggered by the Liaison Nodes which replace the standard Terminal Nodes of the Rete Network as well as the Conclusive Nodes of the Event Detection Graph.

Liaison Nodes. We introduced the Liaison Node type in order to connect the Event Detection Graph with the Rete Network as depicted in Figure 7.3. In general, the Liaison Node has two inputs: a *top* input from the Rete Network and a *bottom* input from the Event Detection Graph. This general kind of Liaison Node corresponds to a typical ARL rule where both, the event part and the condition part are defined. For instance, the left Liaison Node and middle Liaison Node in Figure Figure 7.3 correspond to such an ECA rule (cf. Section 2.6 for a detailed analysis of ECA, EA and CA rules.). The right Liaison Node corresponds to an EA rule where no condition part is defined. Since an EA ARL rule has no condition part the corresponding Hybrid Discrimination Network also has no Rete Network. The third kind of Liaison Node corresponds to a CA rule and also has as, in the case of the EA rule, only one input. Here the input comes from the Rete Network in the form of a Beta Memory. There is no Event Detection Network created since no complex event expression is specified in an CA ARL rule. In all cases the Liaison Node fires its associated rule actions according to the ARL semantics. Thus, for an ECA rule to fire, the corresponding event must be detected, and during the complete detection interval of the event, all conditions of that rule have to be matched by WMEs. If the event specification and all conditions are fulfilled the ARL rule is instantiated and put onto the Agenda.

Conditional Event Processing. The Hybrid Discrimination Network possesses a new and outstanding feature which makes the whole rule evaluation algorithm very efficient: the activation and deactivation of the Event Detection Network based on the fulfillment of its conditions. In other words, event processing can be enabled and disabled on demand. This means that parts of the Event Detection Graph can be deactivated until all conditions of an ARL rule are satisfied. Additionally, this means that as long as the condition part of an ARL rule

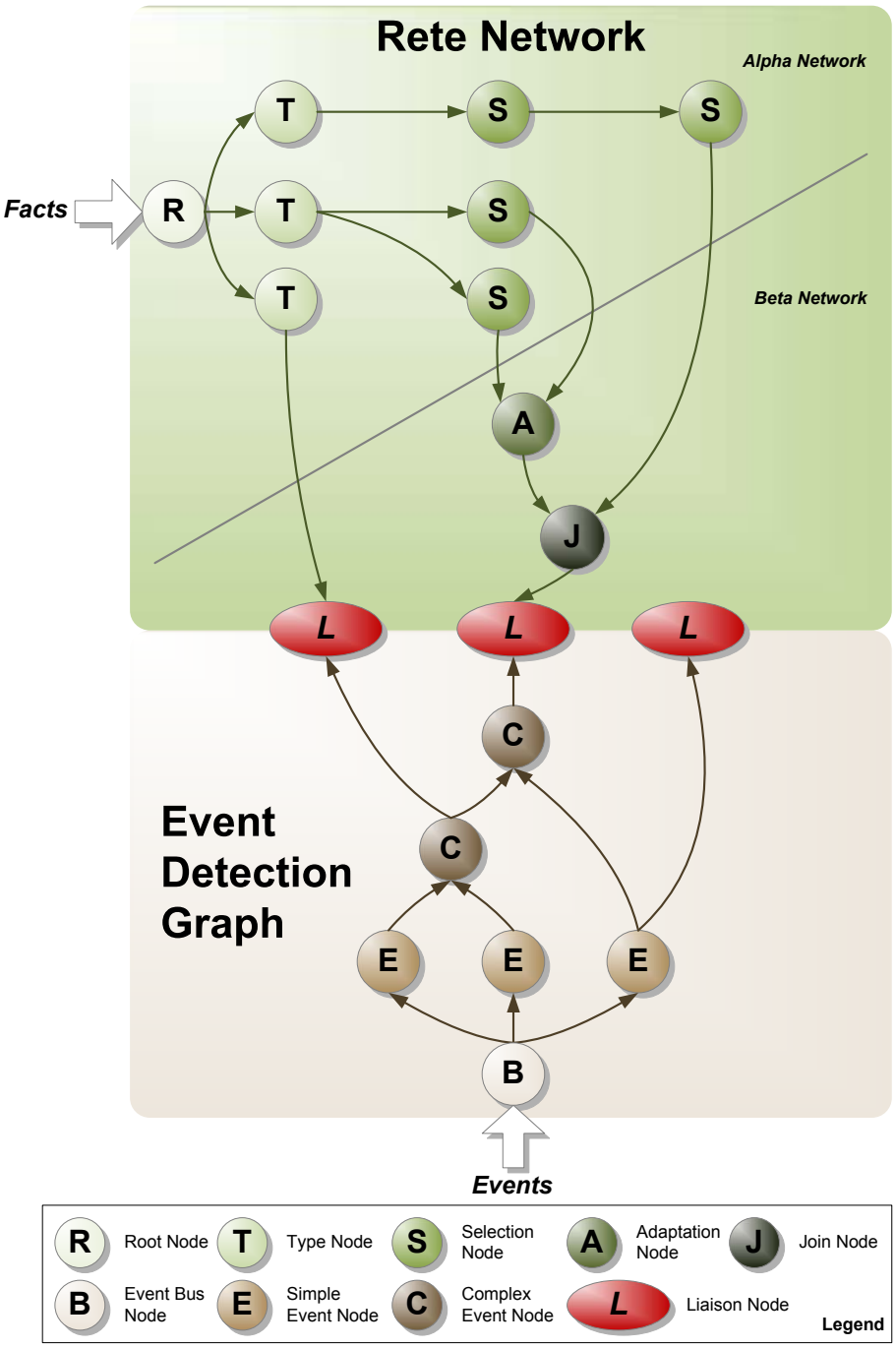

Figure 7.3: The Hybrid Discrimination Network.

does not hold, no event at all will be processed by the Event Detection Network for that individual rule. This feature boosts the time and space performance of the Hybrid Inference Engine. Computation time as well as memory consumption can be reduced during the evaluation of ARL rules. Unneeded event nodes

are disabled during the time the rule conditions are not matched by any objects of the Working Memory. Nodes are disabled recursively from top to bottom of the Event Detection Graph. The first node that is possibly disabled is the top Complex Event Node, and then all of its contributing child nodes are checked for deactivation. Care must be taken when encountering a shared node which has further consumers. Such a node is only disabled when all consumers are disabled as well.

7.2.3 Event Sources and Event Bus

The Event Sources depicted at the bottom in Figure 7.2 list the supported event sources. The Dynamic User Interface/DOM event source issues mouse events, keyboard events, HTML events[4] and mutation events[5]. The Timer event source periodically issues timer events triggered by the internal clock. External event sources might deliver arbitrary domain-specific events from any applications by using server-push technologies like for instance Comet[6] or client-pull technologies like Really Simple Syndication (RSS) feeds[7]. The Trigger event source is actually the Adaptation Engine itself, as it can trigger arbitrary events encoded in the action part of the Adaptation Rules.

Of manifold importance for ARRIAs is the Dynamic User Interface/DOM event source, as emitter of recordable events comprising the user's clickstream. In a web browser HTML pages are internally represented as DOM. Whenever a user interacts with the web page, the DOM fires appropriate events which can be caught by event handlers. In order to catch events an event handler has to first be registered for specific event types. Via ARL the application programmer has to explicitly specify which kinds of events shall be tracked. At run-time each recognized event is fed into the Hybrid Inference Engine. Here the relationships are resolved between the DOM events, the user interface elements and their annotations stored for instance in a Knowledge Base as discussed in Chapter 5. In the Construction Wizard example the DOM events `blur`, `change` and `click` are exploited in order to signal user input for different HTML elements. These three simple events are combined by the `ANY` event operator which fires whenever one of the DOM events is detected.

Events from the different Event Sources are caught by the Event Bus. For DOM events the Event Bus has the role of a user tracking component (cf. Sub-

[4]HTML events signal client-server interactions or changes to the browser window.

[5]Mutation events signal a change to the underlying DOM structure.

[6]Russell (2006) defines Comet as technology using long-lived HTTP connections to reduce the latency with which messages are passed to the server. Comet applications do not poll the server occasionally but instead the server has an open line of communication with which it can push data to the client.

[7]The RSS Advisory Board (2009) defines RSS as a web content syndication format. It is used to publish frequently updated content and has to be occasionally polled by a client application.

section 4.1.3), since DOM events are mainly caused by user interactions. The Event Bus receives all events, adds a timestamp to them and feeds them into the Event Detection Graph by invoking the Hybrid Inference Engine.

7.2.4 Agenda

The Agenda component, which is depicted above in Figure 7.2, determines the order in which the matching ARL rule instances may be fired. We implemented the Agenda as a queue using the FIFO principle. As defined by ARL the Agenda has to be able to execute a JavaScript function, to fire events and to manipulate WMEs (cf. Appendix A Line 258).

For every JavaScript function that is specified in the action part of an ARL rule, a new function[8] is created at runtime. The set of events and WMEs matching the rule instance are passed as input parameters to the function. Thus, within the JavaScript function the constituent events as well as the matching WMEs can be accessed and used for computations. Via JavaScript functions the Dynamic User Interface of ARRIAs can be directly adapted. The Adaptation Engine adapts the Dynamic User Interface whenever an action involving changes to the WUI is put onto the Agenda by the Hybrid Inference Engine. In the Construction Wizard example the `showBuildingNotificationDialog` JavaScript function is called when the event part and the condition part of the noise barrier rule are matched. This function changes the WUI so that a link to the building notification form is displayed.

In the action part of an ARL rule a new event can also be triggered. The Agenda sends this new event directly to the Trigger component. The Trigger component in turn forwards the new event to the Event Bus. The ability to issue new events in the action part of an ARL rule enables the triggering of other rules as the result of a rule evaluation.

Finally, the Working Memory can be manipulated within the action part of ARL rules. The Agenda directly manipulates WMEs by invoking one of the three predefined functions of an object which resides in the working memory: Assert, Modify or Retract. Assert creates a new WME, Modify changes an attribute of an already existing WME and Retract removes a WME from the Working Memory. The assertion and retraction of a WME invalidates the Rete Network and leads to its rebuild. For instance, in the Construction Wizard example taken from Section 6.5 the state of the `conWiz` WME is changed in the action part of the rule. The state of the `conWiz` WME is set to `BuildingNotificationDialog` as depicted in Lines 49–54.

[8] Functions are first-class citizens in the JavaScript language and can be dynamically created and passed along.

7.3 Construction Wizard Example

In this section we again pick up the Construction Wizard example from Section 6.5. With the help of the Construction Wizard example we explain how the Hybrid Discrimination Network is constructed by the Graph Builder in order to match events and facts for adapting the WUI of an ARRIA. The Hybrid Discrimination Network resulting from transforming the ARL rule of the Construction Wizard example (cf. Section 6.5) is depicted in Figure 7.4. The Graph Builder interprets the rule and constructs the Rete Network for the condition part and the Event Detection Graph for the event part.

The Graph Builder starts with the dissection of the event part of the rule. First, the entry point common for all simple events is created. The entry point common for all simple events is labeled with `B`. Afterwards, three Simple Event Nodes are created, one for each simple event defined in the Construction Wizard example in Lines 19, 25 and 32. The simple DOM events which can be detected by the Event Detection Graph are: `blur`, `change` and `click`. After constructing the Simple Event Nodes the Complex Event Node `ANY` is created representing the `ANY` event operator. The `ANY` node is a three-input node, where the three inputs are the simple events defined before. Finally, a new Liaison Node `L` is created and connected to the `ANY` node. Thus, whenever one of the specified simple events is detected, it is passed to the `ANY` node and the `ANY` node forwards it to the Liaison Node `L`.

After dissecting the event part of the Construction Wizard example rule and successfully creating the Event Detection Graph, the Graph Builder analyzes the condition part in order to create the Rete Network. First, the Graph Builder creates the Root Node as common entry point for all WMEs. Thereafter, a Type Node is created to check the object type of the WME forwarded to it. In the Construction Wizard the Type Node checks objects for the type `ConstructionWizard`. After constructing the Type Node two subsequent Selection Nodes are created according to the conditions specified in Lines 37–47 of the Construction Wizard example from Section 6.5. The first condition checks the state attribute of the Construction Wizard object. The condition is satisfied, if the Construction Wizard object is in the state `HeightOfNoiseBarrierDialog`. The second condition tests whether the user entered a height of the Noise Barrier greater than three meters. Since there are no inter-object tests defined in the Construction Wizard rule example, the Beta Network is left out. As a last step the last Selection Node is connected to the Liaison Node `L` which has been already created during the construction of the Event Detection Network.

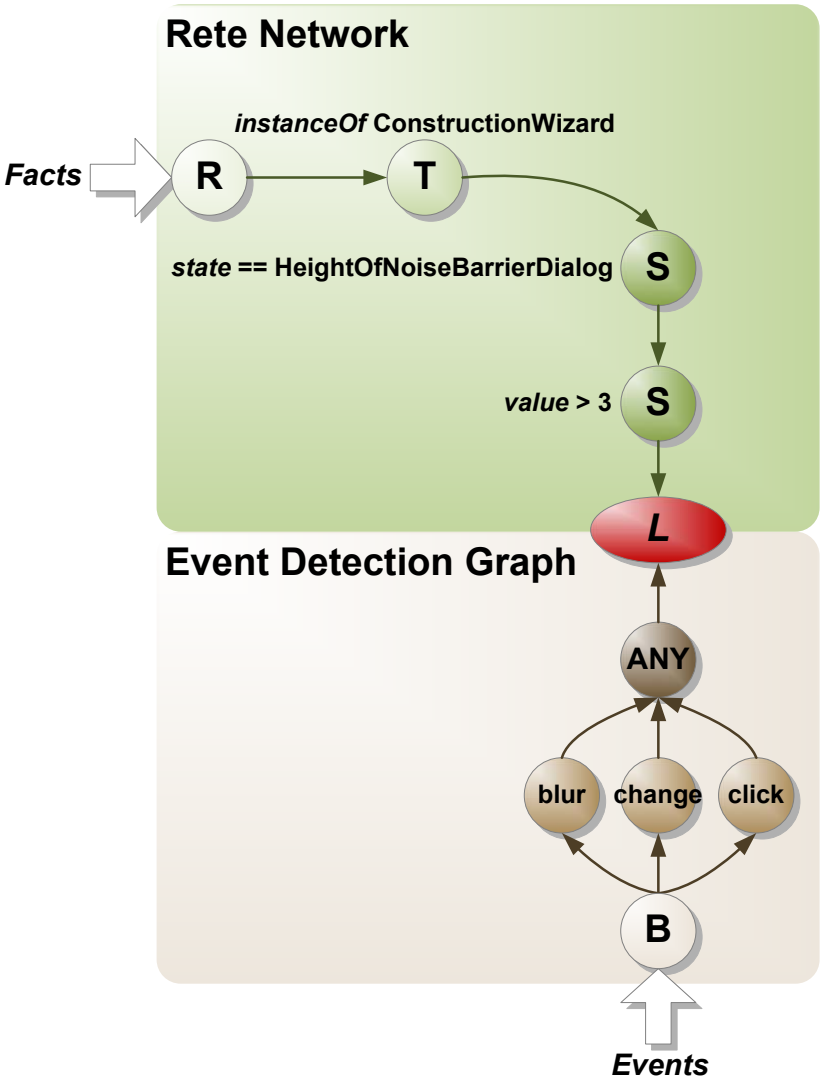

Figure 7.4: Hybrid Discrimination Network of the Construction Wizard example.

7.4 Performance Analysis

In order to show that the Adaptation Engine is competitive with the traditional approach of server-side adaptation computation, we conducted a performance analysis. The results of the performance analysis are meant to provide evidence that the Adaptation Engine is able to provide adaptive and responsive user interfaces for RIAs.

The performance analysis was designed according to the systematic approach to performance evaluation proposed by Jain (1991, pp. 22-25). We slightly adapted this methodology in order to meet our needs.

7.4.1 Goals of the Performance Analysis

Goal 1 (Main Goal). *The purpose of the performance analysis is the assessment of the Adaptation Engine in order to judge its applicability for building adaptive and responsive user interfaces for RIAs.*

As the Adaptation Engine consists of two distinct engines namely the Event Engine and the Production Engine we further divide Goal 1 into two more specific subgoals.

Goal 2 (Subgoal). *Performance assessment of the Event Engine.*

Goal 3 (Subgoal). *Performance assessment of the Production Engine.*

7.4.2 Metrics

To benchmark the Event Engine of ARRIAs we started out with the BEAST benchmark (Geppert et al., 1998). BEAST is a benchmark which measures the performance of CEP applications. It measures the processor load by increasing event frequency.

We also benchmark the performance of the rule engine. As metric for the rule engine we decided to measure the processor load caused by the pattern matching functions with increasing numbers of WME modifications.

7.4.3 Design of the Experiments

As the objective of the performance analysis is to measure the maximal throughput of events in the Event Engine and of changes to working memory elements in the Rete Engine we set up two experiments each measuring the processor load of a test machine depending, first, on events per second, and, second, on working memory modifications per second.

The experiments were executed on a test machine powered by an Intel Core Quad processor. Since JavaScript execution is inherently single-threaded it profits only from one processor core. Having spare cores for other tasks combined with a generally low operating system load provides results which are uninfluenced by other running tasks. We chose the JavaScript engine of Mozilla Firefox 3.0.3 for Windows in conjunction with the Firebug[9] profiler. The browser was installed freshly with no extra plug-ins.

Benchmarking the Event Engine

The complex event expressions which were tested are listed below, using event operators from the SnoopIB algebra (cf. Chapter 6), such as sequence `SEQ`, negation

[9] `http://getfirebug.com`

NOT and disjunction OR. The complex event expressions are part of the BEAST benchmark.

$$\texttt{SEQ}(\text{EvED-061}, \text{EvED-062}) \tag{7.1}$$

$$\texttt{NOT}(\text{ED07\_TX}, \text{EvED-07}, \text{ED07\_TX}) \tag{7.2}$$

$$\texttt{SEQ}(\text{EvED-091}, \texttt{SEQ}(\texttt{OR}(\text{EvED-092}, \text{EvED-093}), \text{EvED-094})) \tag{7.3}$$

The complex event expression 7.1 represents a sequence of two events. The subsequent event expression 7.2 represents the non-occurrence of the event EvED-07 in the interval of two other specified events. Finally, the complex event expression 7.3 represents the sequence of one event followed by a disjunction and followed by another event.

The tested rules contain empty rule actions, so only the processor load for the Event Engine is measured. We ran each test for 30 seconds at various frequencies of simple events per second. The simple events in the mentioned patterns were entered into the detection system in a round robin manner. We then measured the load percentage caused by the detection system matching the incoming events and producing complex events.

Production Engine

We took a similar approach for rule conditions to measure the performance of the implementation of the Production Engine. Three condition expressions were added to the Rete network with the same nesting depths as the event expressions. The working memory was initially filled with 1000 WMEs. Then we started modifying WMEs at different frequencies, round robin. Again, we measured the processor load caused by the pattern matching functions.

7.4.4 Analysis, Presentation and Interpretation of Data

The results of the performance tests for the Event Engine are shown in Figure 7.5. The chart shows that our event detector can handle a maximum of about 64 events per second. After that the JavaScript engine is used up to capacity and further incoming events are queued up. Figure 7.6 shows the benchmark results of the Production Engine. The Production Engine is capable of processing about 32 modifications per second without being overloaded.

Based on these results we judge that the Adaptation Engine meets goal three: applicability for building adaptive and responsive user interfaces. As ARRIAs aim at the user-centric adaptation of RIAs in response to the tracked user interactions, and since events from the human user are not occurring at millisecond rates, the Adaptation Engine is fast enough to handle events on time. Although, expensive rule actions may lessen these results, upcoming new browsers promise

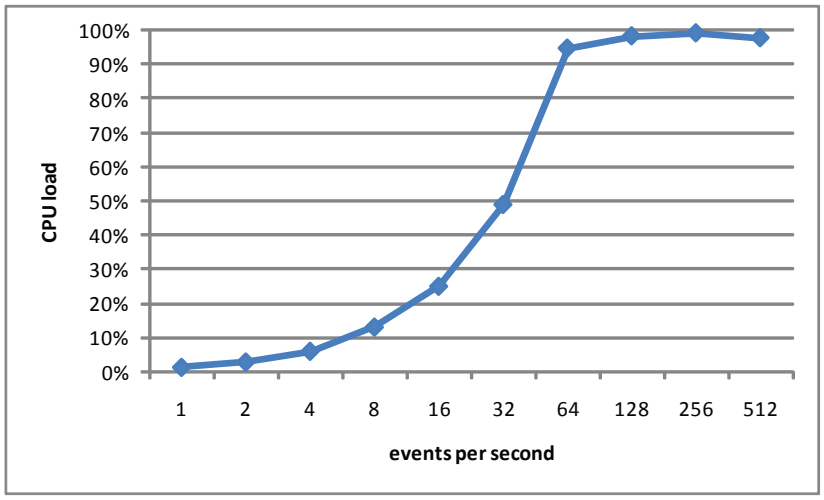

Figure 7.5: Processor load by increasing event frequency.

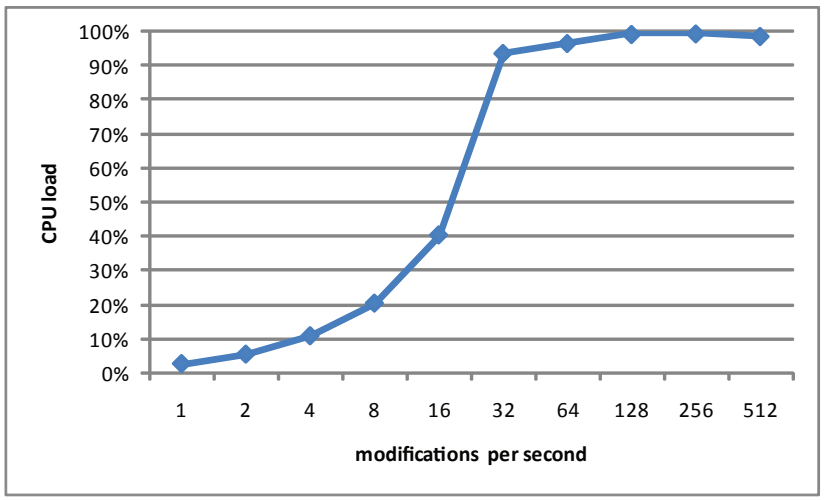

Figure 7.6: Processor load by increasing object modifications.

a significant increase in general JavaScript performance due to newer compiling techniques. Currently, we expect the results to be sufficient for most web applications including those of our use cases analyzed in Chapter 3.

The performance of our Hybrid Discrimination Network introduced in Subsection 7.2.2 varies depending upon a number of factors. If no event nodes are shared, the performance gains in space and time are a function of the event frequency and the probability of unsatisfied conditions in the rule base. In order to properly calculate the needed event detection sub-graphs of an ECA rule, all participating simple and complex event nodes, and all fragment nodes in between must be taken into account. However, determining the average frequency of event occurrences might not always be practicable or accurate, e.g. if a source emits events aperiodically. Additionally, if event detection nodes are shared by

several consumers, assessing the gained performance increase becomes more difficult. Event detection sub-graphs can only be disabled, if all conditions of all subscribed consumers do not hold.

7.5 Related Work

Maloof and Kochut (1993) extend Rete with the temporal operators `before`, `during` and `after` to allow reasoning about relative time representations. These operators may be used to compare temporal facts already stored in the working memory. Events are seen as having a duration, therefore an interval-based semantics is assumed.

Berstel (2002) also proposes to extend the Rete algorithm with the relative operators `before` and `after`. Events are handled as special facts which are asserted through a special API into the working memory of the Rete algorithm. He proposes a time-point-based semantics for their events. Furthermore, he devises a garbage collection mechanism to expunge unneeded events when they are no longer needed.

Walzer et al. (2007, 2008) extend the Rete algorithm with the thirteen temporal relations proposed by Allen (1983)[10]. They completely cover all possible relationships between intervals. Accordingly, Walzer et al. (2008) use an interval-based semantics for events and introduce temporal operators for each of the thirteen relationships. The operators can be quantified, e.g. specifying how much two intervals overlap in the beginning and in the end. Ranges for these quantifiers are also permitted. Walzer et al. (2008) also implement a garbage collection mechanism, that takes into account constraints and time-based windows of the join nodes as well as time stamps and maximum lifetimes of the events.

In the three approaches of Maloof and Kochut (1993), (Berstel, 2002) and Walzer et al. (2008) the semantics of an event that is visible in the working memory is not clearly defined. An event represented in the working memory solely means that the event has not been consumed by all of its subscribers. As also other features of WMEs are not applicable for events the purpose of adding events to working memories is questionable. For example the `modify` operator, the working memory provides to alter its WMEs, is not applicable as events are immutable. According to Neale (2008), Walzer et al. (2008) mark events as immutable facts in their implementation which is an extension of Drools[11] (Proctor, 2007). Moreover, the `retract` operator seems unreasonable for events, as well. An event either happened or it did not happen. All subscribers should consistently receive it. With ARRIAs we followed a completely different approach. Our new approach combines event processing and fact evaluation into a single Hybrid

[10] The thirteen temporal relations are: `equal`, `before`, `after`, `during`, `contains`, `overlaps`, `overlapped-by`, `meets`, `met-by`, `starts`, `started-by`, `finishes`, `finished-by`.

[11] `http://www.jboss.org/drools`

Discrimination Network while keeping events and facts separated. Events are not loaded into Rete's working memory, rather they are feed into a separate Event Detection Graph tailored to the transitory nature of events. The Event Detection Graph and the Rete Network are combined at their edges. This data structure allows us to consider the different semantics of events and facts.

In two of the above approaches of Maloof and Kochut (1993) and Walzer et al. (2008) events are treated as time intervals, whereas one approach (Berstel, 2002) treats them as time points. In the field of CEP this has been discussed for quite some time, e.g. in Galton and Augusto (2002), and the scientific consensus is that to receive expected and consistent results from temporal operators, events must be represented as intervals if they are detected over a period of time. We implemented an interval-based semantics for processing complex events in the ARRIAs approach.

Garbage collection is also treated differently by the three approaches discussed above. Events are streamed into an CEP application, are processed, and, finally, are discarded so that the application does not run out of memory. Events are transient representations of past incidents. These events may be used on a subscriber-basis by different consumers, i.e. rules in the Rete context. When no further consumer is interested in a particular event, it can be discarded. Thus, there is a fundamental difference between facts and events. Facts have a potentially indefinite lifetime. They are only removed if they are explicitly retracted. Events, on the other hand, have individual lifetimes which may be calculated from constraints imposed by their consumers. After their lifetimes are expired, they can be completely removed. In the ARRIA approach the garbage collection of events comes naturally. Events are automatically discarded from the queues of the event nodes when they are not needed anymore. ARRIAs implement no global repository for unused or partially unused events.

Carughi et al. (2007) propose an event-based architecture for RIAs very close to a streaming web architecture. In their paper RIAs have the ability to receive events and to react to those events. Events are observed on the client, however, complex events are not detected in the client. All simple events are propagated to the server for detection of patterns. Our solution goes one step further as we equip RIAs with the ability to construct complex events from simple ones. This architecture makes our solution more powerful, flexible and open for personalization. The main difference is that we moved the detection of complex events from the server to the client. This avoids latency and reduced locality for the processing of events, so the advantages of client-side event processing are preserved.

Garrigós et al. (2007) present in their web engineering paper the Adaptive Web Applications Creator (AWAC) prototype, a computer aided web engineering (CAWE) tool for the automatic generation of adaptive web applications based on the adaptive object oriented hypermedia (AOOH) methodology. They generate the whole web application from models, whereas we use the models only

to annotate already existing web applications. Furthermore, the authors define the personalization rules modeling language (PRML), an event condition action language tailored to the personalization needs of web applications. Our rule language follows a different approach, as it has to deal with complex events on the client-side. PRML does not support complex event processing and is not a general purpose event condition action language supporting more than personalization, in contrast to our ARL.

7.6 Summary

In this section we showed how the ARRIA Adaptation Engine executes rules encoded in ARL, which we introduced in the previous chapter. We introduced and detailed the newly researched algorithm of our client side Hybrid Inference Engine, which consists of the Production Engine and Event Engine. Furthermore, we showed how the Hybrid Discrimination Network which is the core of the Hybrid Inference Engine leverages client-side WUI adaptations. Finally, we evaluated the performance of the Adaptation Engine by metering the performance of the Event Engine and Production engine separately. We confirmed that our implementation fulfills the demands of a client-side personalization engine.

Part III

Demonstration and Evaluation

Chapter 8

Personalized E-Government

This chapter is the first out of three demonstrating the use of ARRIAs based on specific use cases. We demonstrate how ARRIAs enhance the user experience by adding client-side adaptivity and reactivity to the e-Government portal of Vöcklabruck. The architecture and the related work of this use case are already explained throughout the previous chapters. Therefore, we focus here on the evaluation of Vöcklabruck's ARRIA-enhanced e-Government portal. Together with the other two use cases of personalized web advertisement and personalized web search this use case targets at the fifth research contribution (cf. Section 1.4). We published the e-Government use case in the book *Semantic Technologies for E-Government: A European Perspective* (Schmidt, Stojanovic, Stojanovic, and Thomas, 2010a).

In this chapter we demonstrate and evaluate the ARRIA approach based on the personalized e-Government use case described in Section 3.1 and more concrete on the Construction Wizard example detailed in Section 6.5. The Section 8.1 provides an detailed explanation of how we realized the adaptive e-Government portal for the municipality of Vöcklabruck. We present how we use ARRIAs for solving the problem of finding the right form for the building permission process. We focus on demonstrating and evaluating the run-time aspects of our use case solution. In Section 8.2 a comparative usability test contrasts the old, non-adaptive version of Vöcklabruck's city portal with the new ARRIA-enabled adaptive version. By conducting this evaluation we observed and measured how well ARRIAs support a solution to the use case problem. The test results are also discussed here. Finally, in Section 8.3, the chapter is summarized.

8.1 The Adaptive City Portal of Vöcklabruck

Vöcklabruck runs a city portal providing e-Government services (cf. Subsection 3.1.1). Originally, the portal was a plain old, non-adaptive web application where all online-available administrative services were simply listed (cf. Fig-

Figure 8.1: Vöcklabruck's legacy web portal wrapped as ARRIA application with dynamically inserted Construction Wizard.

ure 3.1). In order to solve the problem of finding the right form as described in Section 3.1, and to enable average citizens to find the right web form for their building projects we decided to implement direct user guidance in the form of a Construction Wizard. The Construction Wizard is explained in detail in Section 6.5. We developed the wizard by using ARL and the Adaptation Engine. The whole implementation serves as a proof of concept for the ARRIA approach.

The operators of Vöcklabruck's e-Government portal constrained our ARRIA-enabled solution to be based on the legacy version of their portal. We were requested not to alter the original portal source code.

To meet this requirement, we engineered an ARRIA which wraps the unchanged e-Government web portal of Vöcklabruck as depicted in Figure 8.1. The operators of Vöcklabruck's e-Government portal had to change only two things: First, they had to rename the default HTML start page to a name different from `index.html`. Second, they had to copy the newly developed portal wrapper

page to the appropriate web server directory under the filename `index.html`. Thus, whenever Vöcklabruck's URL `http://www.voecklabruck.at` is requested the web browser first downloads the wrapper page. By using an inline frame (IFrame) referencing the renamed default start page, the unchanged portal is downloaded and displayed. As the wrapper itself has no UI the end user will not notice this indirection as he/she perceives the portal as unchanged. As the wrapper page also references the Adaptation Engine and the Construction Wizard, the web browser downloads them, actually before the original portal is loaded into the IFrame.

As we were constrained not to alter the source code we came up with a solution that alters the city portal after it has been loaded by the browser. To accomplish this we defined adaptation rules which fire after receiving the event that the content of the IFrame — the retained unchanged portal of Vöcklabruck — has been downloaded completely. These rules now change Vöcklabruck's portal on-the-fly, by, for instance, adding links or downloading additional JavaScript libraries. Thus, for instance, the encircled menu entry on the left in Figure 8.1 has been inserted dynamically, and links to the Construction Wizard.

8.2 Comparative Usability Test

We conducted a comparative usability evaluation as part of the User Centered Design Methodology (Norman and Draper, 1986) in order to verify the validness of our approach. We chose the city portal of Vöcklabruck (cf. Subsection 3.1.1) for this evaluation study.

8.2.1 Test Hypotheses

The next five hypotheses formulate our confidence about the advantage of adaptive over non-adaptive web sites. The hypotheses are directly derived from our first research challenge (cf. Section 1.3). The goal of the evaluation test is to gain significant test data that allow drawing conclusions about whether or not these hypotheses hold.

Main Test Hypothesis

Test Hypothesis 1. The usability of an adaptive city portal is greater than the usability of a non-adaptive portal.

Based on the problem analysis spanning all chapters of Part I of the thesis we argue for the implementation of a client-side and context-sensitive adaptation of user interfaces for web-based public services in order to increase user experience. The Test Hypothesis 1, the main test hypothesis, formulates that an adaptive e-Government portal is superior to a non-adaptive portal. In order to test this

hypothesis we conducted a comparative usability test based on the following sub-hypothesis.

Test Sub-Hypotheses

The sub-hypotheses put the main hypothesis in concrete terms by breaking it up into several measurable and more concrete hypotheses.

Test Sub-Hypothesis 2. Citizens are more likely to choose the correct form for their current working context in an adaptive environment than in an environment that does not adapt to the current user context.

Test Sub-Hypothesis 3. Citizens spend less time searching for the correct form in an adaptive than a non-adaptive portal.

Test Sub-Hypothesis 4. Citizens call search or help pages less often in an adaptive portal than in a non-adaptive portal.

Test Sub-Hypothesis 5. The perceived usability of an adaptive e-Government portal is higher than the perceived usability of a non-adaptive e-Government portal.

In the use case analysis of e-Government portals (cf. Section 3.1) we outlined the typical problem of e-Government portals: finding the right form. Test Sub-Hypothesis 2 states that an adaptive portal supports the citizen better in finding the right form than a non-adaptive portal. This also includes decision support for the question whether or not it is necessary to submit a form at all. Furthermore, we hypothesize in Test Sub-Hypothesis 3 that an adaptive portal not only supports citizens in finding the right form, but also shortens their search time. Another indicator for the Test Hypothesis 1 is hypothesized in Test Sub-Hypothesis 4. We assume that citizens do not call search or help pages so often than in an adaptive portal, since the user guidance is much stronger than in non-adaptive portals. The Test Sub-Hypothesis 5 aims at the perceived usability of adaptive and non-adaptive portals. We hypothesize that the perceived usability in the case of adaptive portals is much higher than the perceived usability in the non-adaptive case.

8.2.2 Test Planning

We planned the evaluation of our hypotheses by comparing two versions of the same web portal: a static version without, and an adaptive version with, ARRIA technology. In order to accomplish this, we needed a highly dynamic architecture that allows running two distinct versions of the same portal. The sources of the original portal should stay untouched while the adaptive technology should be added on the fly. This is crucial for low evaluation costs.

The evaluation portal has to choose randomly the version of the portal to be tested. Whenever a test user enters the evaluation portal the web browser 'throws a coin', and either loads the adaptive or the non-adaptive version of the city portal of Vöcklabruck. On average this ensures that both versions are tested equally.

In order to draw statistical conclusions we needed many test users. Our goal was to have at least 15 test users for each version. Taking this into account we decided to recruit the test users from colleagues and friends by approaching them by email.

To reach a high ratio of responses it is crucial for a usability test not to take too long. As we did not give any incentives to the test user we were dependent on their good will. As the good will of test persons shrinks the longer the test lasts, we planned a test that does not take longer than 10 minutes. Additionally we planned a time frame of two weeks for the whole test. This duration gives some freedom to the test users to choose the time point for doing the evaluation which fits best into their schedule.

With our usability test we want to evaluate the first three out of the following standard four usability goals: performance, accuracy, emotional response and recall. Performance is measured in time or number of steps required to complete predefined tasks. Accuracy measures the number of mistakes test users make while completing predefined tasks. The feelings of the test users about the tasks completed are recorded in a questionnaire as their emotional response. Recall[1] the fourth usability goal is not addressed by our test setup because it does not fit to the objectives of our solution.

We considered the validity of our usability test framework by asking a domain expert for the validity of our test cases. We presented our test task to a construction domain expert of the city of Vöcklabruck, and she confirmed that our test cases are valid, sound and address real world problems. She also confirmed that our selection of test users is representative.

The test tasks are the three cases of the building permission process described in Subsection 3.1.1. Table 8.1 summarizes the tasks and the expected results. Results are web pages containing the correct form corresponding to the given tasks. Which form is the correct one for which task is determined by the building law rules, which are described in Subsection 3.1.1.

Design

Figure 8.2 shows the evaluation wizard as a finite state machine. State transitions are preformed by the user either by clicking the navigation buttons *Next* or *Back*, or by clicking the *Task finished* button during the processing of a task. Usually the *Back* button resets the evaluation wizard to the previous state.

[1] Recall measures how much a test person remembers after a while.

Table 8.1: The test tasks and their solutions.

Task	Result
demolition of a double garage	building permit form
construction of a swimming pool	no form
construction of a noise barrier	building notification form

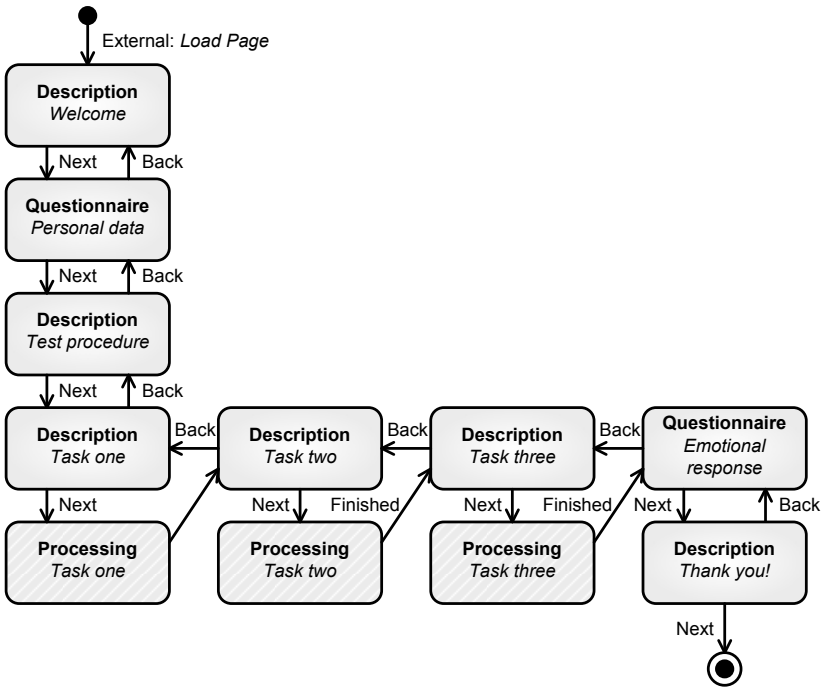

Figure 8.2: Finite state machine of the Evaluation Wizard.

After requesting the evaluation portal from a browser the wizard is loaded. A welcome page is shown to the test user. Figure 8.3 shows a screenshot of the welcome page. After pressing the Next button a questionnaire is loaded asking for the test user's personal data. Every piece of information is voluntary. We asked for: age, gender, mother tongue, intensity of using the Internet, and expertise in construction law. The age is interrogated in intervals from 0-25, 26-35, 36-45, 45-open ended. The options for the gender attribute are female and male, for the mother tongue attribute the options are German and others, for the intensity of using the Internet: seldom, often, intensive; and for the expert in construction law: yes or no. The personal data questionnaire is depicted in Figure 8.4.

The subsequent web page explains the evaluation procedure. A screenshot of it is shown in Figure 8.5 The next six states are conceptually the same. They all are paired with a state describing the task, and a subsequent processing state in which the user tries to accomplish the task described before. Here the quantitative measures are recorded.

Figure 8.3: Screenshot of the welcome page of the Evaluation Wizard.

As already explained in the motivating example, one problem of e-Government portals is finding the right form. Therefore we designed three tasks asking the user to find the right form for a given problem from the building law domain. The tasks are detailed in Subsection 3.1.1. Figure 8.6 shows the screenshot describing the first task: the demolition of a double garage. By clicking on the *Next* button the user starts the task solving process.

Figure 8.7 shows the starting point for processing every task: the home page of Vöcklabruck. While solving the task the user operates either on the static or on the adaptive version of the portal. Which version of the portal is evaluated is decided randomly right at startup of the evaluation wizard and will not change as long as the usability test lasts. Starting from the home page the user has to find the right form for the given task. We injected a colored section providing the user with a short task description in order to remind him/her of the task to solve. Furthermore, this section provides a button labeled with *Task finished* in order to end the task without choosing a form as required in the use case scenario two where a swimming pool has to be constructed and neither a building permit nor a building notification is needed for it.

As the user processes a task, we measure the time the user needs for finishing it. Additionally, we track the user behavior, which means we record the user's clickstream, and we store the chosen form. All these measures serve as quantitative benchmarks. The time spent for solving a task can be compared between the adaptive and non-adaptive version of the city portal. The recorded clickstream provides deep insights into the browsing behavior of the current user. The fre-

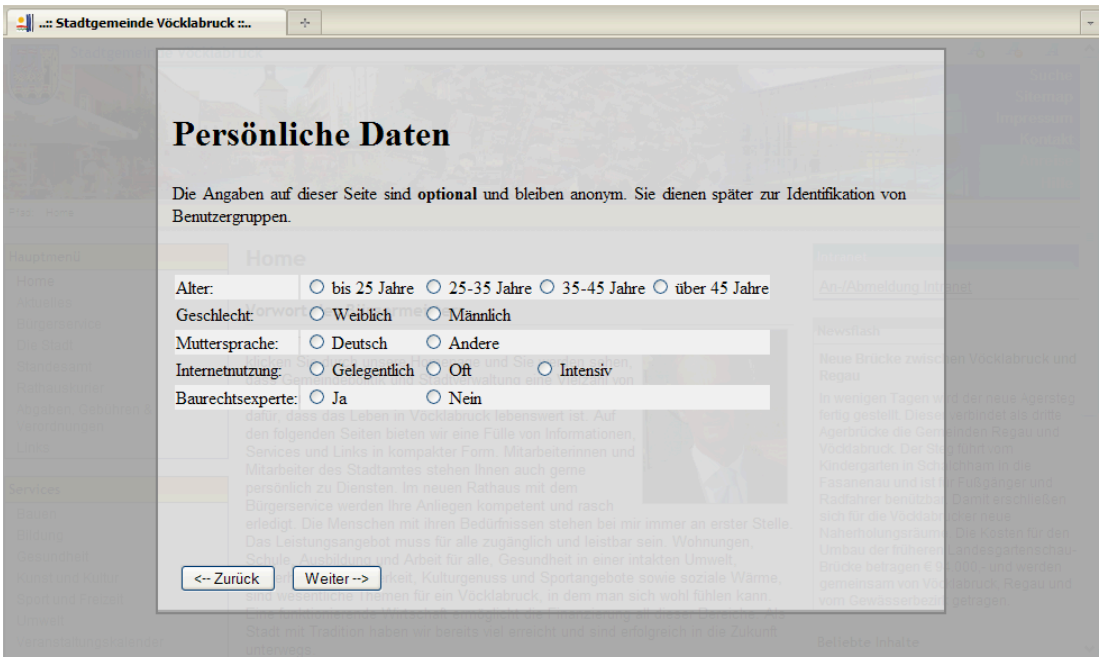

Figure 8.4: Screenshot of the personal data questionnaire.

quency of calling internal help or search pages can be evaluated. Finally we can compare the ratio of correctly chosen forms to all chosen forms for the adaptive and non-adaptive version of the portal.

The emotional response is captured by a questionnaire after solving the first three tasks. Figure 8.8 shows an empty screenshot of the questionnaire. In the questionnaire we ask the users for their subjective impressions of the city portal of Vöcklabruck. We ask the following questions:

1. How satisfied are you with the portal?
    - Users are asked for their overall impression of the usability of the portal. In order to specify their level of agreement to this statement a Likert scale (Likert, 1932) is applied.
2. Are the appropriate forms easy to find?
    - This more concrete questions aims at subjective feelings of how easy it is to find the right form for the set task. For this question also a Likert scale is applied.
3. Would you prefer the online portal or the citizens' advice bureau?
    - In this question we ask the user whether the usability of the city portal is superior to the citizens' advice bureau. Here a trivalent scale (Yes/No/No Answer) is applied.

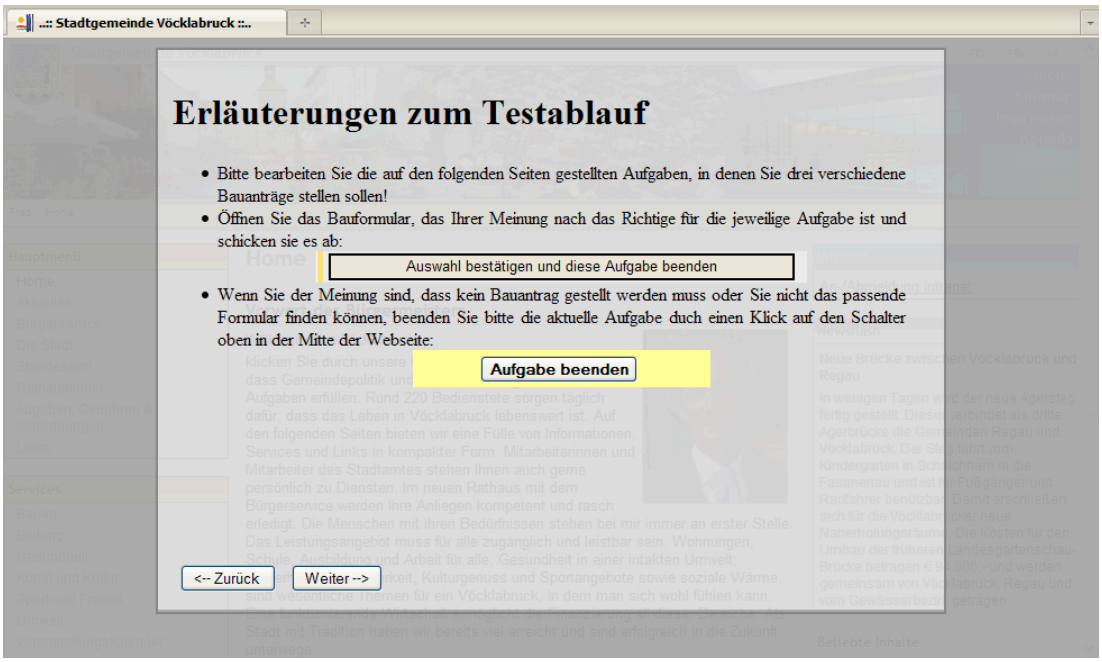

Figure 8.5: Screenshot explaining the evaluation procedure.

4. Comments

   - Here the user can enter free hand comments. User comments might give useful hints towards specific usability problems not foreseen in the design of the evaluation as well as of the original portal.

Finally, a farewell screen is displayed in the *Thank you!* state. The farewell screen expresses gratitude to all the test users who participated in the usability test.

8.2.3 Architecture and Implementation

The architecture follows the major design directive of not modifying the city portal of Vöcklabruck. Therefore, the evaluation test server relies for both test versions on the same original and unchanged city portal of Vöcklabruck. The test portal delivers the ARRIA consisting of User Tracking and UI Adaptation. The test portal also provides the evaluation wizard and construction wizard.

Both, the User Tracking and the UI Adaptation components are initialized with the rules and annotations retrieved form the server hosting the usability test. While initializing the adaptation components, the start page of the city portal of Vöcklabruck is loaded from the portal web server. After successfully initializing the adaptation components and after successfully loading the start page, the adaptation components decide what kind of portal will be tested by the current test user: the adaptive or the non-adaptive version. This decision is

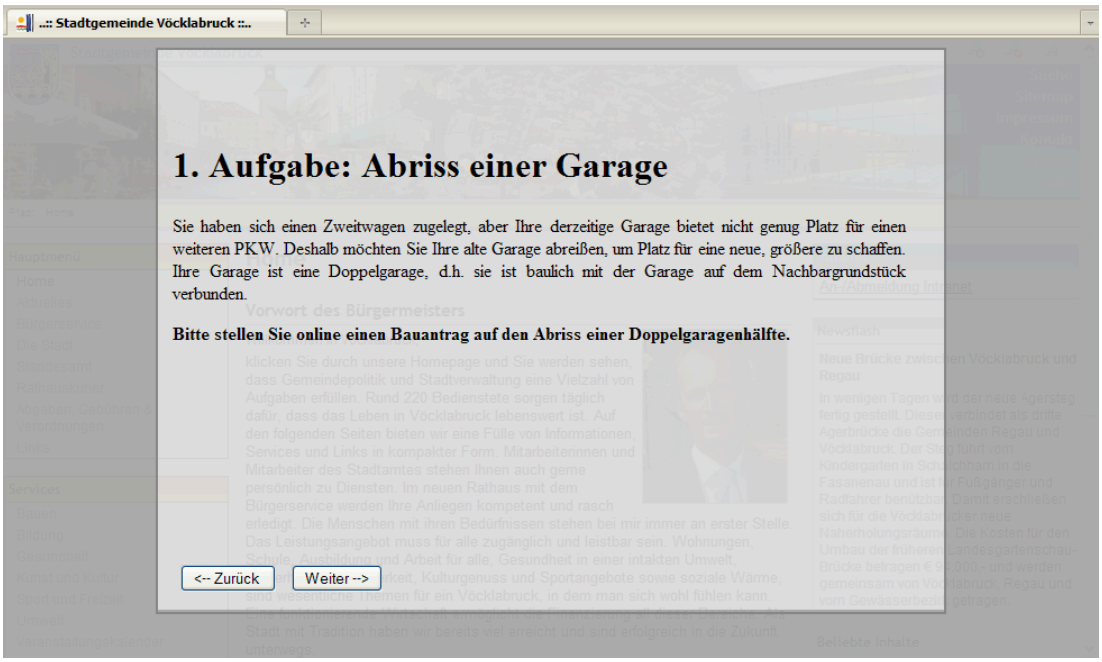

Figure 8.6: Screenshot describing the first task.

made randomly every time the evaluation portal is accessed. This ensures that on average the adaptive portal is evaluated as often as the static portal by the test users.

In the case of the adaptive version the original start page is altered. A link to the construction wizard is added dynamically. The static version of the portal is not altered at all, and, thus, will not have a construction wizard supporting the test user to fulfill the test tasks.

For both versions common events are registered according to the tracking rules. Additionally, listeners are implemented for dynamically adapting the original forms, menus, help and search pages. Then the evaluation wizard is started. The user model recorded as the user does the three tasks, as well as the questionnaire data are stored at the storage server in a file in JSON format.

The Adaptive Portal

In order to make the static e-Government portal adaptive, the Construction Wizard is dynamically injected (cf. Section 8.1). The Construction Wizard guides the user, based on a question/answer mechanism, to the right construction form. During run time the Construction Wizard evaluates adaptation rules which comprise the three building law rules (cf. Subsection 3.1.1). Depending on his/her previous answers the user is guided to the right form; what the right form is depends on the given task.

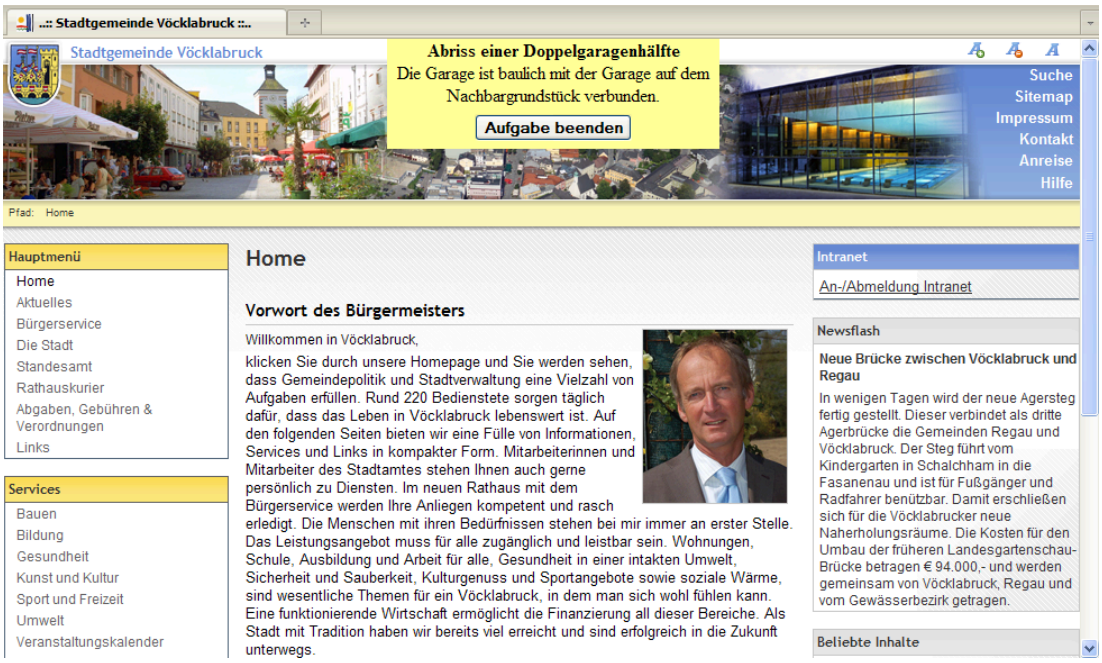

Figure 8.7: Screenshot of the static home page of Vöcklabruck with an ejected section repeating the task that has to be solved and a button for ending the task.

Test of the Evaluation Portal

Before going public we tested the usability test portal intensively. These checks ensured the compatibility of our software with all major browsers.

Additionally, we evaluated the usability test itself by means of a randomly chosen test user. During this evaluation it turned out that there was too much continuous text that could not be easily understood. We changed that in the final version of the evaluation wizard. Also, a completely new path to the construction forms was discovered. Furthermore, that evaluation impressively exhibited the *Where is the cow?* problem (Nielsen, 1993), as the test user followed always the same path to the constructions forms regardless of the presence of the construction Wizard. First, the non-adaptive version of the portal was evaluated. Afterwards, the test person tested the adaptive version. Surprisingly, the construction wizard was not used, because the test person went exactly the same path, she had already taken for the non-adaptive version.

8.2.4 Execution and Test Results

In the run-up to the comparative usability test we invited about 215 people by email to participate. We conducted the test for 16 days from the 6th October 2008 until the 21st October 2008. During that time we ensured, by daily checks, that the portal is always up and running. The test itself was performed by each

Figure 8.8: Screenshot of the questionnaire for capturing the emotional response of an test user.

test person autonomously. Forty-nine people out of 215 participated in the test. This is a quota of about 23%. The adaptive version of the portal was tested 33 times and the non-adaptive version 16 times. A complete list of the test results is given in Appendix B.

Forty-nine test participants out of 215 approached people is a good ratio of 23%. We are satisfied with this percentage as we did not provide any compensation for participation. The test participation was only based on the good will of the approached people.

The ratio of static test runs to adaptive test runs is 1:2. That is a little bit surprising, as intensive pre-release laboratory tests showed, that in the long run the average ratio is 1:1. One possible explanation for the observed imbalance is that some test users gave up working on the tasks in the static version of the portal. One reason could be the lack of the construction wizard, and, with that, the lack of user guidance. It is conceivable that they had serious problems finding the right form and therefore canceled the whole test procedure.

Demographics

Over 50% of the test users were between 26-35 years and 21% of the people between 36-45 years. The 21% of people under 26 are owed to recruiting also students for our comparative usability test. Only 4% were older than 46 years. Figure 8.9 depicts the aging structure of the test user population.

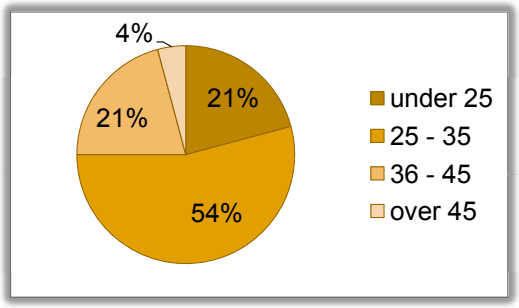

Figure 8.9: The age distribution of the test users.

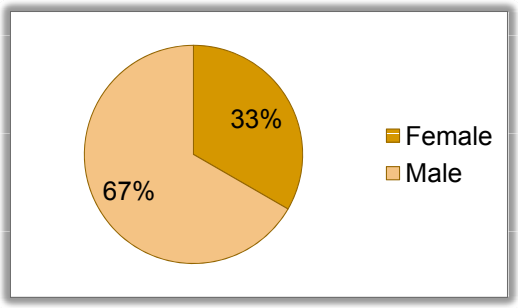

Figure 8.10: The gender distribution of the test users.

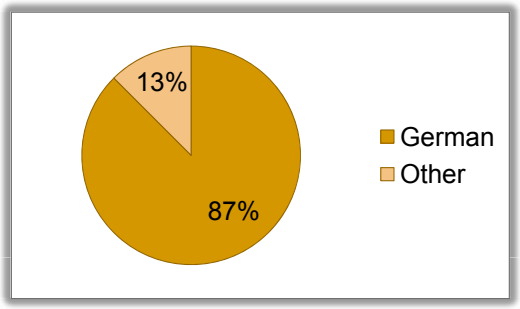

Figure 8.11: The German as mother tongue distribution of the test users.

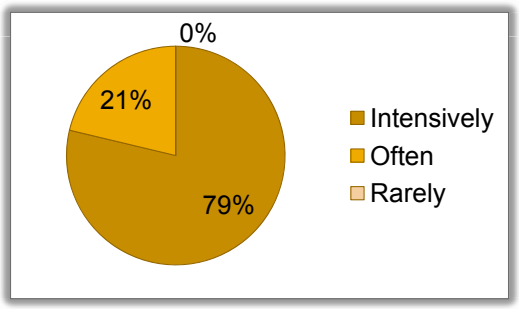

Figure 8.12: The familiarity with the Internet distribution of the test users.

One third of our test participants were women. This ratio is depicted in Figure 8.10. That corresponds to the ratio of women to men we addressed by our invitation emails.

Eighty-eight percent of the test persons were native speakers; only 12% did not speak German as a mother tongue. The ratio of German to non German speakers is depicted in Figure 8.11. This meets our expectations, as we mainly approached German speaking persons.

As depicted in Figure 8.12 the test users were very familiar with the Internet. Seventy-nine percent stated that they intensively use the Internet, 21% declared they often use the Internet. None of our test users rarely uses the Internet. That is not surprising as our usability test is web-based. Furthermore, we asked for participation via email and administrative and scientific staff work daily with the Internet.

Unfortunately, no expert in building law participated in our test. As the portal is designed for common people this is negligible, although it would have been nice to have their browsing behavior as a comparison value.

Table 8.2: Z-test for the correctness hypothesis.

null hypothesis	$H_0 = H_1$
alternative hypothesis	$H_0 < H_1$
level of significance	0.01
upper critical value	2.3263
correctly chosen forms in the static portal:	0
correctly chosen forms in the adaptive portal:	26
z-test statistic	**4.7757**
p-value	**8.95E-01**
Reject the null hypothesis!	

Quantitative Test Results

The statistical significance of our findings was assessed with hypothesis testing. The purpose of hypothesis testing is to test the viability of the null hypothesis in the light of experimental data. The null hypothesis is a hypothesis about a population parameter. It is typically a hypothesis of no difference (e.g. no difference between population means, proportions, variance etc.), although it can include also the direction of the effect. Depending on the data, the null hypothesis either will or will not be rejected as a viable possibility for a given level of significance. The alternative hypothesis on the other hand relates to the statement to be accepted if the null is rejected.

In order to decide whether the null hypothesis can be rejected for a given level of significance we chose the two-sample Z-test of proportion (Zou et al., 2003) for testing Sub-Hypothesis 2 and Sub-Hypothesis 4, and the two-sample T-test of mean for Sub-Hypothesis 3 since time is a cardinal variable. The p-value of the test has a central role in the decision regarding the rejection of the null hypothesis. A p-value is a measure of how much evidence we have against the null hypothesis (the smaller the p-value, the more evidence we have against it).

In the Test Sub-Hypothesis 2, we hypothesize that the ratio of rightly chosen to wrongly chosen forms is higher for the adaptive version of the portal than for the static version. The test users working with the static version of the portal were never able to choose the right form, not a single time. In contrast, working with the adaptive version leads in 79% of the cases to the selection of the right form. Zero percent of right forms for the static and about 80% of right forms for the adaptive portal deliver a clear message. The adaptive portal leads to a significantly higher percentage of correctly chosen forms. Evidence is given to the Sub-Hypothesis 2 by the Z-test results illustrated in Table 8.2. With regard to this measure the adaptive portal is clearly more effective and more efficient than the static one — Test Sub-Hypothesis 2 is proven.

Table 8.3: Two-sample T-test for the duration hypothesis.

		static group	*adaptive* group
null hypothesis	$\mu_{static} = \mu_{adaptive}$		
alternative hypothesis	$\mu_{static} > \mu_{adaptive}$		
upper critical value	2.4233		
level of significance	0.01		
degrees of freedom	40		
mean in ms		189912	121761.06
variance		15569312507	3558199088
t-test statistic	**2.6021**		
p-value	**0.006177**		
Reject the null hypothesis!			

Also the Test Sub-Hypothesis 3 could be proved. Test results reveal that the fastest test user, although not finding the right form, used the static version of the portal. He/she was three times faster than the fastest user of the adaptive portal. This could mean that experienced users familiar with building permissions might be faster using a static portal. On the other hand, the test results showed that on average it takes less time to find a form by using the adaptive portal. The two-sample T-test illustrated in Table 8.3 significantly revealed that test persons using the adaptive portal finish their tasks on average quicker than test persons using the static, non-adaptive version of the portal.

The frequency of using search or help pages is an indicator of the usability of a portal. We hypothesized in the Test Sub-Hypothesis 4 that users of the adaptive portal call search and help pages less frequently than users of the static version. The collected clickstream data revealed that a call to the search page is four times more likely in the static portal. Five times more users visited the help page in the static portal compared to the adaptive portal. As illustrated in Table 8.4 and Table 8.5 we could only partially prove the Test Sub-Hypothesis 4, since only the significance test results for accessing the help pages gave evidence to the Test Sub-Hypothesis 4. The test results for accessing the search pages gave no statistically significant evidence to the Test Sub-Hypothesis 4.

Qualitative Test Results

We hypothesized in Test Sub-Hypothesis 5, that the perceived usability of an adaptive e-Government portal is higher than the perceived usability of a non-adaptive e-Government portal. To prove this hypothesis we presented a questionnaire to each test user, asking three questions.

In the first question we asked the test users for their overall impression of the usability of the portal. The answers to this question are broken down in

Table 8.4: Z-test for the help hypothesis.

null hypothesis	$H_0 = H_1$
alternative hypothesis	$H_0 > H_1$
level of significance	0.01
lower critical value	-2.3263
access to help pages in the static portal:	7
access to help pages in the adaptive portal:	3
z-test statistic	**-4.1178**
p-value	**0.000019**
Reject the null hypothesis!	

Table 8.5: Z-test for the search hypothesis.

null hypothesis	$H_0 = H_1$
alternative hypothesis	$H_0 > H_1$
level of significance	0.01
lower critical value	-2.3263
searches in the static portal:	4
searches in the adaptive portal:	2
z-test statistic	**-1.8966**
p-value	**0.028942**
Do not reject the null hypothesis!	

Figure 8.13. Seventy-three percent of the test users who evaluated the adaptive portal were enthusiastic or very satisfied with the portal in contrast to 7% of the test users who evaluated the static portal. This is a strong vote for the use of adaptive portals in e-Government.

Another strong vote for adaptive portals is given by the answers to question two as depicted in Figure 8.14. Here we asked for the subjective feeling of how

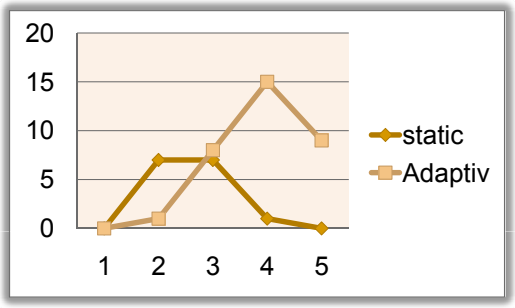

Figure 8.13: User satisfaction.

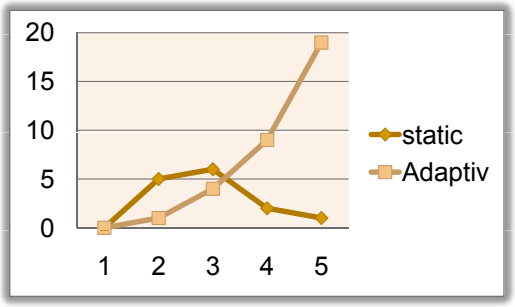

Figure 8.14: Easy to find forms.

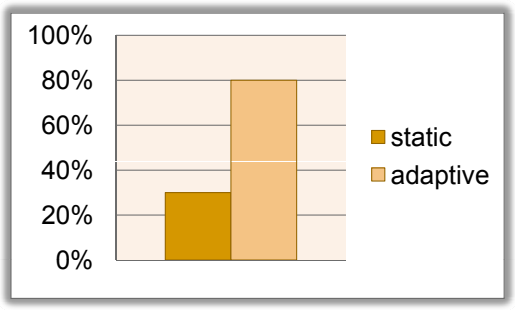

Figure 8.15: Preference to online portal.

easy it is to find the right form for the set task. Eighty-five percent of the test persons investigating the adaptive portal found it very easy or easy to get to the right form. Compared to that, only 21% users testing the static portal stated that it is very easy or at least easy to find the right form.

Additionally, we collected yet more evidence of the superiority of adaptive over static portals in the answers to the third question, in which we asked the test users whether the usability of the city portal is superior to the citizens' advice bureau. The answers draw a clear picture. Over 80% of the test users investigating the adaptive portal answered with *yes* whereas only about 30% of the users testing the static portal answered with *yes* (cf. Figure 8.15).

Finally, the comments given by the test users also supported the Test Hypothesis 1. All comments are listed in Appendix B. Mostly users who tested the static version of the portal wrote comments. The most frequent comment given by users testing the static portal was that they could not find an appropriate demolition form as requested in Task 1. This issue was not raised at all by users testing the adaptive portal, yet more evidence that the adaptive portal effectively solves one of the most pressing problems in e-Government portals (cf. Section 3.1), not finding the right form.

8.3 Summary

The evaluation of the comparative usability test, between an adaptive and a static version of the city portal of Vöcklabruck, gave evidence for our main Test Hypothesis 1, that the usability of an adaptive city portal is higher than the usability of a static portal. The qualitative as well as the quantitative test results impressively demonstrated the advantage of the adaptive over the static portal.

The quantitative test results showed that only the adaptive portal enables inexperienced users to choose the right form out of many. Additionally, on average, users working with the adaptive portal navigate faster to forms, and they also less frequently visit help and search pages.

The qualitative test results confirmed by subjectively perceived advantage of adaptive portals over static portals. The perceived usability of the adaptive portal in comparison to the static portal was more than twelve times higher, and more then twice as many prefer the online portal over the citizens' advice bureau. Four times more users rated the ease of finding the right form higher in the adaptive than in the static portal.

After performing the evaluation our conclusion for public administrations is: The ARRIA framework greatly enhances the usability of e-Government portals, increases the effectiveness and efficiency both of the citizens and the public service, and lowers costs by decreasing wrongly submitted forms.

Chapter 9

Personalized Web Advertisement

Current approaches, that can be used for realizing web advertisement, fail to generate very personalized ads for a current web user that is visiting a particular web content. They mainly try to develop a profile based on the content of that web page or on a long-term user's profile, without taking into account the user's current preferences. We argue that by discovering a user's interests from his/her current web behavior, we can improve the process of advertisement placement, especially the relevance of an ad for the user. This work has been presented at the 8^{th} *International Semantic Web Conference* (Stühmer et al., 2009b) and at the 8^{th} *International Conference on Ontologies, DataBases, and Applications of Semantics* (Stühmer et al., 2009a).

This second use case (cf. Section 3.2), out of three, serves as an additional demonstrator for showing how ARRIAs advance the state of the art of reactivity on the web and promote new ways of efficiently communicating web-based information. Generally speaking, this use case demonstrates how ARRIAs contribute to the realization of new, event-driven applications for the future web. In this use case the ARRIA technology stack, which is depicted in Figure 4.1, is not altered, except for the mode of annotating web pages with domain knowledge. So far, we stored annotations in a separate Knowledge Base. While this approach works well for web applications supplied by a single vendor, it does not fit the case of portals incorporating web objects from different content providers like, for instance, news portals. Therefore, we propose to include domain knowledge directly into the web site by using microformats instead of a monolithic Knowledge Base for personalized web advertisement. Additionally, instead of directly guiding the user we follow the adaptation approach of suggesting relevant links in this use case.

The chapter is structured as follows. The first section details the ARRIA-enabled personalized web advertisement architecture. The use case scenario from Section 3.2 is picked up and an exemplary web advertisement rule in ARL is presented. In Section 9.2 the results of a preliminary usability test are provided,

and in Section 9.3 related work is presented. Finally, in Section 9.4, our work is summarized.

9.1 Architecture

The ARRIA-based approach for personalized web advertisement is based on two main approaches: *contextual advertising* and *behavioral targeted advertising*. According to Kenny and Marshall (2000) contextual advertising is driven by the user's context usually in the form of keywords extracted from the web site content or related to the user's geographical location and other contextual factors. We enhance the state of the art by using semantics to address major drawbacks of today's contextual advertising. At the same time we also utilize behavioral targeted advertising based on the user's behavior, collected through the user's web browsing history. We address the drawbacks of behavioral targeted advertising by realizing short-term profiling (cf. Subsection 2.3.2) using the client-side reactivity inherent to ARRIAs. Contextual advertising and behavioral targeted advertising are discussed in more detail in Section 9.3. In the following subsections we first detail the design-time architecture, and afterwards the run-time architecture for personalized web advertisement based on ARRIAs.

9.1.1 Design-Time Framework

In accordance with the ARRIA Adaptation Framework users are profiled by first annotating web sites semantically, second, tracking the web usage behavior of a group of users directly in their clients, and, third, mining meaningful behavioral patterns from the tracked user profiles augmented with semantic annotations. Based on the behavioral patterns, client-side adaptation rules for ad placement can be designed by an ad service provider. Ad service providers like Google decide which ads to place based on context information.

Annotation of Web Applications

In order to better understand events from web clients and make sense of what happened we must semantically enrich the content of events when they are produced. A simple event in web clients is characterized by two dimensions; the type of event (e.g. click, mouseover) and the part of the web page, where the event occurred (e.g. a node of the DOM of the web document). This node is, however, just a syntactical artifact of the document as it is presented in a web browser. Adding this node, or parts of it, to the event body will not significantly add meaning to the event, and not ease the understanding of the event for the receiver of the event.

We therefore propose to add semantic information to the event which pertains to the actual topics that the web page is about. Semantic information

is added to web pages by their authors, like it is the case with The New York Times[1]. In order to add semantic information, the first step is to represent the content of a web page in a form useful for generating meaningful events. To do so without having to manually annotate every web document, we envision a mechanism, which ensures the relevance of the annotations. This can be done, e.g., by providing web forms as page templates, which for a given user's input, automatically adds the proper semantic relationships between the form fields. In this way all user generated content will be annotated. The web forms are created based on supported vocabularies for a particular web site. Our particular focus is on widely spread vocabularies such as Dublin Core[2], Creative Commons [3], FOAF[4], GeoRSS[5] and OpenCalais[6]. Regarding the format of structured data, RDFa (Adida et al., 2008), eRDF[7] and Microformats[8] are all good candidates for this purpose. They support semantics embedded within actual web page data, and allow reusable semantic markup inside of web pages. In our implementation we use *RDFa*, since in comparison to eRDF it is more favored by the W3C. Comparing it further to Microformats, RDFa is more flexible in mixing different existing vocabularies.

The ad space is a part of the web page which can be dynamically filled by an ad service provider in response to an event the client sends. In our approach the ad content is created based on the user's current intention, i.e., an event pattern being recognized in his/her behavior. In order to accomplish this, we need as much metadata as possible about the content of the web page. Therefore, we assume semantically enriched web content such that the context extraction is easier and more precise. Additionally, every page is split up into a number of Semantic Web Widgets (SWW). We introduce SWWs as self-contained components annotated with semantic data and displayed in a web page. SWWs contain semantic annotations giving a high-level description of the content, and providing the basic context of data contained in the widgets. For instance, on a news portal which would like to incorporate semantic advertising, one widget could be used for listing all news belonging to one subcategory, e.g., politics, another one for rain forest, etc. We use these annotations to discover user's interests when detecting complex events.

Extracting the context for web advertisement can also be handled in an environment where semantic annotations and relationships already exist. For in-

[1] `http://www.nytimes.com`
[2] `http://dublincore.org`
[3] `http://creativecommons.org`
[4] `http://foaf-project.org`
[5] `http://georss.org`
[6] `http://opencalais.com`
[7] `http://research.talis.com/2005/erdf`
[8] `http://microformats.org`

stance, a semantic media wiki[9] is a wiki that has an underlying model of the knowledge described in its pages (Völkel et al., 2006). This knowledge can be used to get the context of information a user is currently interested in, and, subsequently, to offer relevant advertisement.

Mining of Behavioral Patterns

In order to detect nontrivial situations of interest, simple events are combined into more complex events. To target specialized advertising we want to detect the behavior of a user on a web page. Statistical data from past users is required to define what is interesting behavior to an ad service provider. Such behavior is modeled as sequences of events observed from past usage of the web site. In order to form complex event expressions, these annotations are combined with a temporal model. Such expressions group the user's atomic actions into temporal contexts like, e.g., sequences of clicks. Determining sequences of interest is based on analyzing historical usage data statistically. By using data mining algorithms for click streams as described in Chapter 5, historical data is transformed into knowledge about unusual sequences of interaction such as clicks. Subsequently, the corresponding complex event expressions can be created.

Rules Design

We demonstrate the design of rules for personalized web advertisement by means of the use case scenario detailed in Section 3.2. A simple sequence along with its confidence might be *Politics* followed by *Rain Forest* with a low confidence of 2%. This means that only 2% of previous users have looked at a politics widget, followed by looking at a rain forest widget. This pattern in the user's behavior can be treated as unusual, i.e. his/her interest in Politics and Rain Forest are distinguished from the interests of others, so that this can be used for developing a very personalized ad. Such an ad will very likely attract the attention of the user, since it directly corresponds to his/her short-term profile.

The following listing shows the ARL representation of the example rule, which can be automatically created by analyzing histories of interesting behavior. The only requirement is knowledge that, e.g., states that only two percent of users look at a politics item followed by a rain forest item. The actual rule consists of an event part starting at Line `005` and an action part starting at Line `020`. The rule resembles an ECA rule where the condition is left blank, i.e. is always true.

The event part in this example describes a sequence of two sub-events. Both sub-events are of type `DOM` which causes the Adaptation Engine to add DOM event handlers to the web page. In this case each one listens for clicks on `div` elements in the DOM annotated with the concepts Politics and Rain Forest. The rule action is of type `TRIGGER` which means the rule raises another event. The event

[9] `http://semantic-mediawiki.org`

to be created is called `unusual` and carries a parameter containing a probability. This event can be subscribed to by further rules. Our example rule is a small part of the rule repository which detects, aggregates, and, finally, submits the user profile in the form of one or more complex events to the ad service provider.

```
001 { "PRELUDE" :
002   { "Rule"  : "Politics->RainForest=>2%"
003   },
004   "RULES_REPOSITORY" :
005   [ { "ON" :
006       { "OPERATOR"           : "SEQ",
007         "CONSTITUENT_EVENTS" :
008         [ { "TYPE"     : "DOM",
009             "SELECTOR" :
010             [ "div[property=dc:keywords][content~=politics]" ],
011             "EVENT"    : "click"
012           },
013           { "TYPE"     : "DOM",
014             "SELECTOR" :
015             [ "div[property=dc:keywords][content~=RainForest]" ],
016             "EVENT"    : "click"
017           }
018         ]
019       },
020       "DO" :
021       [ { "TRIGGER"    : "unusual",
022           "PARAMETERS" : { "probability" : 0.02 }
023         }
024       ]
025     }
026   ]
027 }
```

9.1.2 Run-Time Framework

The run-time architecture of the ARRIA approach for personalized web advertisement is depicted in Figure 9.1. The architecture is divided into client-side and server-side components by the protocol boundary. Below in Figure 9.1, on the server-side, several distributed servers hold the web content as well as the advertising content. The web content is annotated semantically relating it to the advertisements. The ad service provider analyzes user profiles to provide up-to-date and personalized advertisements. On the client-side a short-term model of

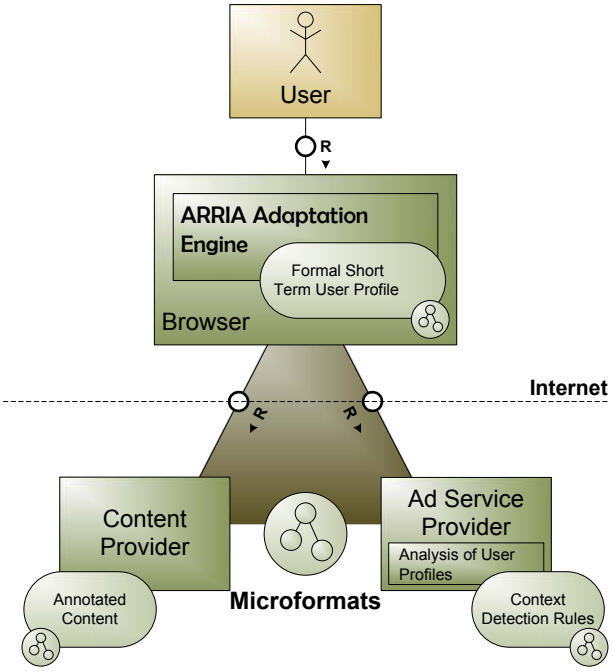

Figure 9.1: Run-time architecture of the ARRIA approach for personalized web advertisement.

the individual user is built at run-time by the ARRIA Adaptation Engine providing a temporal model of how the individual user currently interacts with the web content. The logic for the Adaptation Engine is supplied by ARL rules from a repository generated, as previously described, from past user activities.

At run-time the task of the ARRIA Adaptation Engine is to detect the user's short-term profile, and, subsequently, to deliver it to the ad service provider. After detecting a simple event which happened in the context of a SWW, we collect all semantic information in the web page about the subjects associated with the SWW. We currently achieve this by employing the client-side RDFa library ubiquity[10]. The acquisition of context is achieved in a two-phase process. In the first phase we collect all Resource Description Framework (RDF) triples annotating the SWW, and, subsequently, extract their subjects. This is done within the SWW close to where the event happened in the document to provide *accurate* context. To find valid subjects the first phase traverses the node where the event happened and its complete subtree. In the use case example this could include a news article about a politician and all the contained paragraphs. In the second phase all triples with the given subjects are collected from the entire document tree. In the use case example this includes all triples about the politician from the article as well as possible extra information scattered over the remaining parts of the web page. The gathered triples are appended as a bag to the event payload.

[10]The Ubiquity RDFa parser project; `http://ubiquity-rdfa.googlecode.com/`.

Even if the event itself becomes part of more complex events during the process of correlating and aggregating events, this basic data is retained as part of the simple event.

Once the complex events have been detected, higher-level rules fire. The consecutive actions of these rules transmit the completed short-term user profile to the ad service provider. The profile contains all participating events, so that the model can be evaluated by the ad service provider.

9.2 Evaluation

We provisionally evaluated the ARRIA-based personalized web advertisement approach by rating the retrieved keywords, which describe the user context, in terms of relevance to what test users had been doing within a test web site. The test site consists of four news articles. Each news article is wrapped by its own SWW and in turn each widget is annotated, using RDFa in conjunction with Dublin Core[11], with basic keywords pertaining to the article like, e.g., Politics or Rain Forest. Actually, the keywords are not taken from a controlled vocabulary or ontology. They are simple keywords enclosed by Dublin Core's *Subject* element.

For a user entering the test web site, each widget is partially concealed. The rationale for this is to solicit an action from the user to fully reveal the widget. Thereby the user expresses interest. This creates events which can then be processed by the ARRIA Adaptation Engine.

The initial evaluation of the ad quality was performed by five test users as follows:

1. We selected three different news domains: politics, culture, sports. Three different news domains were selected in order to prove the domain independence of the approach.

2. We selected five test users. Actually, the test users were Ph.D. students in the field of informatics.

3. The test users were instructed to browse the test web site and judge the relevance of generated ad-keywords in the case of

   (a) the keywords generated statistically from the web site (Google approach) and

   (b) keywords generated by using the event-driven approach described above.

   In order to ensure a fair comparison, the users did not know which list of ad-keywords was produced by which method.

[11] `http://dublincore.org`

User behavior is tracked and processed according to the rules. Subsequently, we asked the users to rate the gathered keywords in terms of relevance to what they had been doing on the news test site and to compare this with a list of keywords extracted statically from the overall page. The preliminary results are very encouraging: 85% of the time the keywords generated by our approach are described as *very relevant*, and 98% of the time as *relevant*. The results are very similarly distributed across all three domains. The traditional approach was rated as *very relevant* 65% of the time and as *relevant* ad-keywords 85% of the time.

In this test environment the ARRIA approach of providing relevant and very relevant personalized advertisements is slightly superior to the approach of generating keywords statistically from the content of a web site.

9.3 Related Work

In web advertising there are essentially two main approaches, contextual advertising and behavioral advertising. Contextual advertising (Kenny and Marshall, 2000) is driven by the user's context, represented usually in the form of keywords that are extracted from the web page content, or are related to the user's geographical location, time and other contextual factors. An ad service provider utilizes this metadata to deliver relevant ads. Similarly, a users' search words can also be used to deliver related advertisement in search engine result pages. However, contextual advertising, although exploited today by major advertising players (e.g., GoogleAdsense[12], Yahoo! Publisher Network[13], Microsoft adCenter[14], Ad-in-Motion[15] etc.), shows serious weaknesses. Very often the automatically detected context is wrong, and hence ads delivered within that context are irrelevant[16]. For instance, a banner ad offering a travel deal to Florida can possibly be seen side-by-side with a story of a tornado tearing through Florida. This is happening because the context was determined using purely keywords such as Florida or Shore without taking keyword semantics into account. While there are improvements in contextual advertising, e.g., the language-independent proximity pattern matching algorithm (Schonfeld, 2008), this approach still often leads companies to investments that waste their advertising budgets, and negatively impact brand promotion and sentiment. In contrast, our approach utilizes semantics to address these major drawbacks of today's contextual advertising. Semantics can be used to improve the analysis of the meaning of a web

[12] `http://google.com/adsense`

[13] `http://publisher.yahoo.com`

[14] `http://adcenter.microsoft.com`

[15] `http://ad-in-motion.com`

[16] Adam Ostrow, When Contextual Advertising Goes Horribly Wrong; `http://mashable.com/2008/06/19/contextual-advertising`.

page, and accordingly to ensure that the web page contains the most appropriate advertising.

Behavioral targeted advertising is based on the user's behavior, collected through the user's web browsing history. The behavior model for each user is established by a persistent cookie. For example, web sites for online shopping utilize cookies to record the user's past activities and thereby gain knowledge about the user, or a cluster of users. There are several reasons why behavioral targeted advertisement via cookies is not a definitive answer to all advertisement problems. After browsing and purchasing an item the user might not be interested in buying the same item again. Therefore, all ads, related to that purchased item, offered after purchasing it, may be annoying. Therefore, we propose to use short-term user profiling during the current user session, as displayed ads need to reflect current moods or transient user interests. Furthermore, there are problems with cookies. Computers are sometimes shared, and users see only ads prompted by other user's cookies.

9.4 Summary

We identified drawbacks in current web advertising approaches and demonstrated that ARRIAs foster the usage of *short-term user profiling* for realizing user-centric advertising. Assuming that the semantic annotations already exist, complex event detection enables ad service providers to deliver fine grained personalized advertising. The process uses statistical data captured from previous users. We implemented these patterns and provided a preliminary evaluation study.

Chapter 10

Personalized Web Search

Today's knowledge workers are confronted with an ever increasing information overload while searching for needed information in the web. Common search engines do not take into account the current working context of the user. But we consider context information as an effective means to implicitly narrow the information space of the web. We researched a new approach based on the ARRIA framework that increases the relevance of search results by considering the user's context. Our approach to personalized web search was presented at the *ACM Symposium on Applied Computing (SAC)* (Schmidt et al., 2009a).

This use case is the last, out of three, demonstrating the wide and cross-domain applicability of ARRIAs. Compared to the ARRIA solution for the other two use cases, the novelty of the approach presented here is that ontologies are replaced by keyword-based metadata, from social bookmarking sites, which are referred to as tags. According to the prominent bookmarking service *Delicious*[1], social bookmarking sites allow users to tag, save, manage and share web pages from a centralized source. The rationale to use metadata from social bookmarking sites is to cut down the costs for the development and maintenance of ontologies and to shift these costs to the web community annotating web sites with no costs involved for our personalized web search solution. By choosing this approach we took the loss of a controlled formal vocabulary, but gained an already large-scale annotated WWW. We assume that the tags given by a community of users to web sites represent a good summary of the content such sites.

Furthermore, the ARRIA approach to personalized web search is implemented as a browser extension which has to be explicitly installed by the user. This allows us to track user behavior across different web sites, and, thus, to track the complete browsing history of a user, which greatly facilitates the personalization of subsequently issued search queries.

[1] `http://delicious.com`

A third difference to the common ARRIA approach is the lifetime of the user model. While in the preceding use cases personalization is based on a short-term user model only, we additionally implemented a long-term user model.

The chapter is structured as follows. In Section 10.1 the architecture of our approach to personalized web search is detailed. The design-time aspects as well as the run-time aspects of our solution are discussed and illustrated with the help of a use case scenario. Section 10.2 presents the results of a preliminary qualitative evaluation of our ARRIA-based approach. In the next Section, Section 10.3, we compare our approach to related work. Finally, in Section 10.4 we summarize our solution to personalized web search.

10.1 Architecture

The architecture of the ARRIA Search Extension is depicted in Figure 10.1. The ARRIA approach to personalized web search is implemented as a browser extension in order to track the user's web browsing behavior across web sites. It monitors user actions, tracks what pages have been visited and extracts the metadata needed for the refinement of the search queries from external sources instead of the visited documents themselves. Social bookmarking sites are used as an external source and at the same time as a social filter with a collective vocabulary describing the content of the visited pages. The implementation of our approach is non-intrusive, as it is built upon already existing web search engines. The original non-personalized search results are displayed side by side with the personalized search results, so that the user can work as he/she is used to working, but at the same time can profit from the personalized search results. Our open architecture also facilitates the incorporation of different search engines, metadata providers, result pages and algorithms to select the terms for the proper refinement of search queries. The ARRIA Adaptation Framework is divided into the Design-Time Framework and Run-Time Framework (cf. Chapter 4), which are elaborated next.

10.1.1 Design-Time Framework

The four phases of the ARRIAs Design-Time Framework (cf. Chapter 5) are: provision of metadata, annotation of web applications with these metadata, discovery of behavioral patterns based on annotations, and, finally, the design of adaptation rules. The phase of metadata provisioning is, together with the phase of annotating web sites, already accomplished by the community process of social tagging. This is a process whereby many users collectively and collaboratively annotate pages with metadata in the form of keywords, which are called tags (Golder and Huberman, 2006). Furthermore, the phases of mining common behavioral patterns and designing adaptation rules are reduced. We do not mine

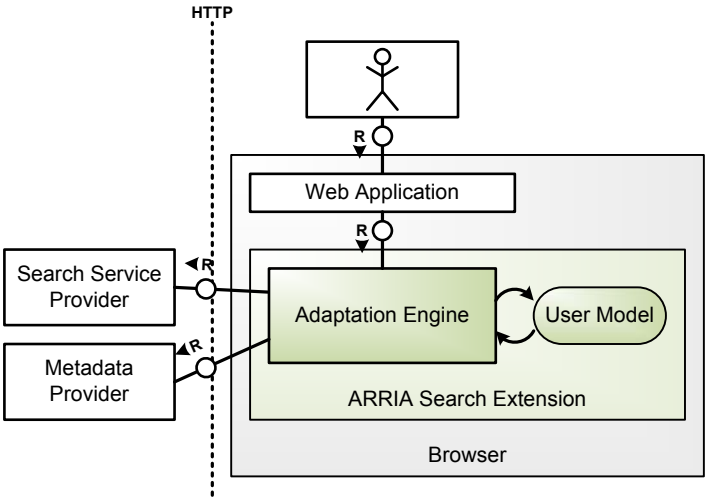

Figure 10.1: Architecture of the ARRIA Search Extension.

common behavioral patterns as user profiles are not shared. In our solution the user profiles are stored locally and exclusively on the client machine. Adaptation rules are used to instruct the Adaptation Engine to track the user and to start the personalization process of web search.

Our approach to personalized web search uses an aging user model. *Aging* means that recently visited web sites are ranked higher then web site visited some time ago. We made this design decision, as we wanted to emphasize the current user context, and, thus, the short-term user model without loosing track of the long-term user interest, the long-term user profile. Furthermore, web sites that are visited frequently are ranked higher than sites visited rarely. We assume that the frequency of web sites visits correlates with their relevance for the current user context. Finally, Internet users tag web sites differently, resulting in tag clouds with frequently used and rarely used tags. For example, the Wikipedia web site[2] is tagged with *encyclopedia* 5285 times and with *dictionary* 637 times. Thus, the frequent *encyclopedia* tag is rated higher than the less frequently used *dictionary* tag. Additionally, our user model also takes into account that web sites are bookmarked to different degrees, e.g., the home page of Wikipedia has been tagged about 35,000 times, whereas the home page of SAP[3] has been tagged less than 600 times. In order to compare web sites with different tagging degrees the individual tag counts are normalized.

The mathematical model for ranking the relevance of tags from a tag cloud for a web site is implemented as a metric. As the focus of this use case is the application of ARRIAs rather then the development of a sophisticated rating model for tag relevance we implemented only a simple straightforward metric.

[2] `http://wikipedia.org`
[3] `http://www.sap.com`

Our exemplary metric implementation combines two different datasets, the collection of visited web sites and the tags for each web site. The metric calculation starts by querying the local browsing history database for recently visited web sites, in our case for the recent 20 web sites. Duplicated sites are accumulated. So, we get a set of web sites in the format (a, b) where a stands for the web address and b for the accumulated number of visits. All web sites constitute the set: $\omega = \{w_i, \ldots, w_n\}$, where n represents the number of the most recently visited web sites. The first value is the last visited page and so on. When we combine these two sets, we get the set (1). The main idea is, that the metric rates recently visited web sites higher then older ones. To do so, the relevance of a web site is determined by a formula, and the weights are set (2). We use the rational functions $f(x) = \frac{1}{x}$ to give the last visited web site the highest rating. The second visited page has only half the weight and so on.

$$x = \{(a_i, b_i), \cdots, (a_n, b_n)\} \; ; \quad |x| = |\omega|,\, b_i \in \mathbb{N} \quad (1)$$

$$\gamma = \{g_i, \cdots, g_n\} \; ; \quad g_i = \frac{1}{i} * b_i,\, |\gamma| = |\omega|,\, g_i \in \mathbb{R}_+ \quad (2)$$

Every web site has a different number of tags. A tag has the format (f, α) where f is the tag name and α the number of users that assigned the tag to the web site. The set (3) contains all the tags assigned to a single web site. The value m is the number of all tags for that web site. For all web sites we get the set (4).

$$t = \{(f_j, \alpha_j), \cdots, (f_m, \alpha_m)\} \; ; \quad \alpha_j \in \mathbb{N} \quad (3)$$

$$T = \{t_i, \cdots, t_n\} \; ; \quad |T| = |\omega| \quad (4)$$

Internet users tag web sites differently, resulting in a tag cloud with frequently used and rarely used tags. For example, *wikipedia.org* has the frequently used tag, *encyclopedia*, which 4451 people have used, and *dictionary*, a relatively rarely used tag, which was only used by 492 people. With this result we can say *encyclopedia* should be rated higher than the other tag. But, we also have the problem, that web sites are bookmarked to different degrees, e.g., the highest count for *wikipedia.org*, 4451, may far exceed counts of tags for another page, because that page is not so important for other users. But, we have to compare these different web sites with their tags, so we normalize the values and get the set (5). For all web sites we get the set (6).

$$\vartheta = \{v_j, \cdots, v_m\} \; ; \quad v_j = \frac{\alpha_j * g_i}{max(t_i)},\, v_j \in \mathbb{R}_+ \quad (5)$$

$$\sigma = \{\vartheta_i, \cdots, \vartheta_n\} \; ; \quad |\sigma| = |\omega| \quad (6)$$

The aim of the metric is to get the most relevant tags. Set (6) contains all tags with names and weights. But we need a new set containing only unique tag names. Therefore we ranked all duplicated tags higher. In the end we have a set of key and value pairs, where the tag names are the keys and the calculated weights are the values.

This is only one example of a metric. It is also possible to configure our solution with another one. Thus, for instance, frequently visited pages may be rated higher than others, etc. The metric is further detailed in Schmidt et al. (2009a).

Our architecture enables the quick exchange of the metric even during the run-time of the ARRIA Search Extension. Furthermore, the web address of the web search provider and the locations of the bookmarking services providing the metadata can be configured via the ARRIA Search Extension UI.

10.1.2 Run-Time Framework

At run-time the ARRIA Search Extension basically performs two tasks: user tracking and web search personalization, which are detailed next.

User Tracking

Our approach to personalized web search tracks the user's web browsing behavior, stores visited pages and builds up a user model based on this information. As the user browses, the stored URLs of the visited pages are enhanced with tags from social bookmarking sites. This is done asynchronously by using Ajax to grab the corresponding tags for a web site. It is possible to configure the ARRIA Search Extension with more then one metadata provider, so that metadata from different social tagging sites can be grabbed. Thus, for example, it is possible to use Delicious together with Digg[4] at the same time.

The URLs are stored together with their corresponding tags and a timestamp in a database called User Model that can be queried via SQL. Thus, the user automatically provides context information by simply *surfing* through the WWW. The ARRIA Search Extension acts upon loading a new page, reloading a page or switching between pages displayed in different browser tabs. This logic is encoded in ARL.

Web Search Personalization

When the ARRIA Search Extension detects that the user has entered the pre-configured search page the personalization process is started. The personalization process is partitioned into four phases. In the first phase the original search query is intercepted. In the second phase this search query is enhanced by the most

[4] `http://digg.com`

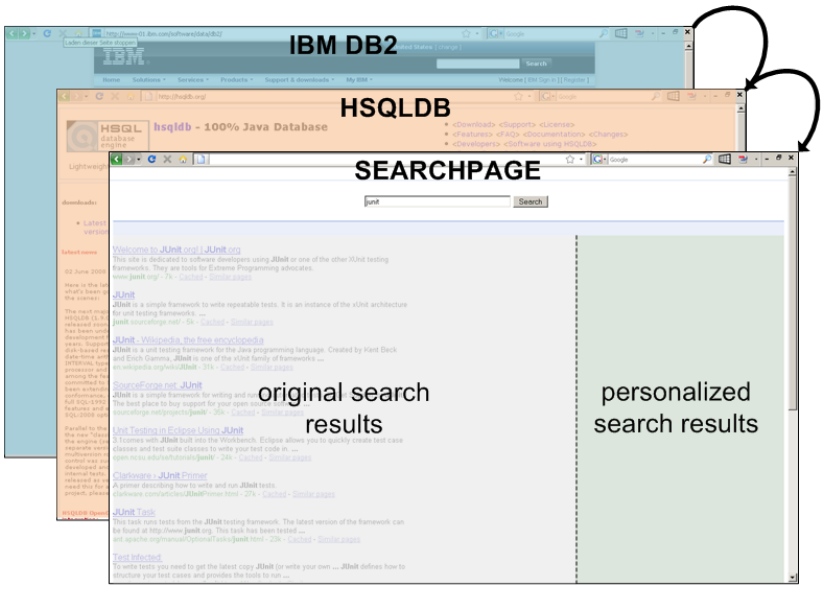

Figure 10.2: Abstract use case scenario.

relevant tags according to the user's browsing history. Which tags are most relevant is determined by a metric. Afterwards, the third phase starts. Now, both the original and the enhanced search query are issued in parallel to the pre-configured search engine, e.g., Google. Finally, in the fourth phase, the search engine's result page is altered in order to also display the personalized search results next to the original, unpersonalized search results.

10.1.3 Use Case Scenario

In order to illustrate the run-time behavior of the ARRIA Search Extension, we revisit the use case scenario from Section 3.3. In the scenario Mrs. Weber is looking for guidelines on how to test some database specific code snippet.

As illustrated in Figure 10.2, she already visited two web sites recently: `http://www.ibm.com/db2` and `http://www.hsqldb.org`. These web pages, illustrated in the background of the figure, represent the user model of Mrs. Weber. The user model comprises not only the URLs of the already visited web pages but also metadata describing the content of the pages. The metadata are asynchronously fetched from external social bookmarking services by the ARRIA system installed in advance as a browser extension. The visited pages are tagged on the social bookmarking page Delicious with *db2* and *database* for the first page and *database* and *java* for the second page.

As Mrs. Weber is looking for testing guidelines, she issues the following query in Google[5]: *test*. Now the ARRIA system uses the tag-enhanced browsing history

[5] `http://www.google.com`

of Mrs. Weber to refine her search query. In the moment, when Mrs. Weber enters the pre-configured search site and issues her search query the common search process familiar to her is started. The search query is issued unchanged and the result set is displayed without any adaptation. Behind the scenes the search query is recognized by the ARRIA browser-extension. The query is enhanced by the most prominent and relevant tags from the user model. The relevance of a tag is calculated by a metric, which is also pre-configured.

Now, the refined search query, which consists of the original keywords plus the most relevant tags from previously visited pages, is issued to the search page. The result page is filtered for the result links, and then the search page itself is manipulated in order to add the personalized search results. The personalized search page is divided into two parts, one part on the left for the original search results and another part on the right for the personalized context-sensitive results (cf. Figure 10.2).

The original search results returned by Google range from personality tests and web-based testing software to intelligence quotient tests, whereas the personalized search results, taking the user context into account, range from a DB2 test database generator over DB2 universal database to a thread on *java.net* about connecting to DB2 databases (cf. also Figure 10.3). The personalized results are tailored to the search history of the current Internet user and provide more specific results consistent with his/her working context.

10.2 Evaluation

We evaluated the ARRIA Search Extension by a qualitative user-driven usability test. In order to evaluate our approach to personalized web search we prototypically implemented our solution as Firefox extension[6]. The extension mechanism is a standardized way of adding new functionality to Firefox. We configured the ARRIA Search Extension with Google[7] as the web search engine. After installing the ARRIA Search Extension a new toolbar becomes visible on the Firefox UI. The toolbar provides access to reconfigure the parameters of our solution like changing the search engine, changing the metadata provider, changing the metric for calculating the relevance of tags, or even to switch off the solution entirely. Furthermore, the content of the User Model database can be deleted via the toolbar. Figure 10.3 shows the extension in action. On top there is the toolbar with the mentioned *on/off* button (1), the configuration option button (2) and a list of received and ranked tags (3) that will be used for the refinement of a search query. As described in the use case example, the user enters the search term in (4) and starts the search. Then the normal search results are displayed, while in the background an asynchronous search request with the extended search query is

[6] `https://developer.mozilla.org/en/Extensions`
[7] `http://www.google.com`

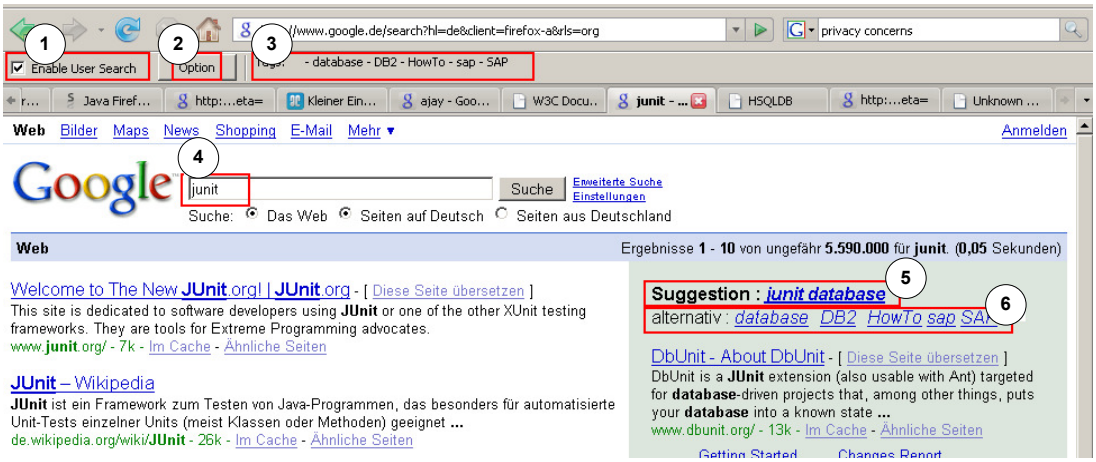

Figure 10.3: Concrete use case scenario.

issued, and than displayed side by side with the non-personalized search results. The refined query is displayed in (5,6).

We formulated the following usability hypotheses for our qualitative evaluation study:

Usability Hypothesis 1. *The ARRIA Search Extension facilitates the daily information retrieval tasks of information workers.*

Usability Hypothesis 2. *The ARRIA Search Extension integrates smoothly into the already existing search infrastructure and does not disturb users in their daily work.*

We designed a two-stage usability test. In a first stage, the behavior of the test users is tracked, and, in a second stage, a questionnaire is filled-in providing insights to the test users subjective opinions. The questionnaire uses a Likert scale (Likert, 1932) and is listed in Table C.1 in Appendix C.

Four informatics students evaluated the ARRIA Search Extension five days long. Nielsen (1989, 1990, 1993, 1994) and Krug (2005) argue that three to five persons discover about 85% of the usability problems. As the ARRIA Search Extension aims at all Internet users the selection of the test persons is not subject to any target group constraints. Actually, Nielsen argues that target groups do not really matter except for web sites designed only for special target groups.

After installation the ARRIA Search Extension was configured with Google as the Internet search provider, Delicious as the metadata provider and the metric described in (Schmidt et al., 2009a). At any time the ARRIA Search Extension could be turned off in order to enable untracked browsing. The following user actions constituting the context of the test users were tracked and saved in the User Model database by using SQLite[8]:

[8]SQLite is a software library implementing a SQL database engine; `http://www.sqlite.org`.

- Every page view of `http://www.google.com` or `http://www.google.de`
- Every click on a non-personalized native Google search result
- Every click on a personalized search result computed by the ARRIA Search Extension

We could track 192 search queries issued to Google in total. On average, in five percent of the cases the test users clicked on a personalized web search result computed by the ARRIA Search Extension rather than on results natively returned by Google. This means that in five percent the result list of our personalized web search approach outperforms the result list of Google. We regard this as a very good ratio as Google already delivers excellent search results.

Additionally, the user survey also casts a positive light on the ARRIA Search Extension. The tool was used by two test persons frequently or always. Three test persons, out of four, confirmed its usefulness. Also, the quality of the search results was judged as good by three test users. Every test participant would recommend the personal test enhancer browser add-on.

The tracked user actions and the end user survey indicate a tendency towards the acceptance of the Usability Hypothesis 1. We can state that the ARRIA Search Extension facilitates the daily information retrieval tasks of information workers.

The answers to the question regarding the presentation of the personalized search give support the Usability Hypothesis 2. Every subject held that the visualization of the personalized search results was not disturbing, but integrated, or even very well integrated.

10.3 Related Work

Personalized web search systems (Micarelli et al., 2007; Micarelli and Gasparetti, 2007) typically rely on short-term and long-term user models expressing the user's current information needs as inferred from previous user actions, browsed web sites or past search queries. Their aim is to improve search results compared to traditional unpersonalized IR techniques. The adaptation technologies that can be applied here are re-ranked search results or modified and refined search queries. Our ARRIA-based approach to web search personalization leverages both user model approaches and combines them to a medium-term user model.

There exist many different web-based systems which try to help the user to find what he/she is searching for. The most prominent example of a personalized web search approach on the server-side is Google's *Web History* (Google Inc., 2007). The search history of a logged-on user is tracked, including the viewed search results. Based on this web history Google personalizes the search results of new search queries. In contrast to ARRIAs which store personal data

like web histories only on the client, Google's server-side solution causes privacy concerns (Volokh, 2000; Pospisil, 2007). Another distinguishing feature is that our approach is web search provider agnostic, and, as such, is not committed to Google solely. Furthermore, our approach does not break the mental model of the individual user (cf. Section 2.3) as we visualize the personalized search results in addition to the original, unchanged results from the search service provider. In addition our approach is able to explain the personalization process by displaying the keywords used to refine the original search query. Our client-side approach is able to track any pages, not only those which are viewed as search results, and it does not require any user account. To compare our personalization algorithm with Google's is not easy as it is not completely transparent, how the web page history influences the search results.

Letizia was introduced by Lieberman (1995, 1997), as an autonomous interface agent that assists web browsing. It was one of the first approaches which used implicit user feedback for personalization. Letizia leverages a simple keyword-frequency IR measure to analyze the content of a page. These keywords are used to construct a long-term user profile. In order to get potentially interesting links, it performs a breath-first search of neighboring links. The ARRIA approach of personalizing web search takes a different approach for gaining metadata about the content of visited web pages by leveraging externally provided tag descriptions. Furthermore, we interweave our link suggestions into the original web application, thus providing an unobtrusive way of personalizing web applications.

10.4 Summary

In this chapter we demonstrated how ARRIAs can be applied to the use case of personalized web search. We presented the client-side ARRIA Search Extension which refines search queries by tags gathered from social bookmarking sites in order to personalize web searches. The ARRIA Search Extension is implemented as a Firefox extension using an aging user model. We showed how metadata, grabbed from social bookmaking sites, can be used to obtain personalized search results based on the browsing history of the individual user. The ARRIA Search Extension uses the collaborative process of social tagging in order to gather metadata for visited web sites. A first preliminary usability test indicated that our approach delivers valuable personalized search results and is easy to use.

Chapter 11

Conclusions

Rich Internet Applications (RIA) revolutionize the web. Responsive and rich graphical user interfaces (GUI) are no longer solely the domain of native desktop applications. Compared to traditional Adaptive Hypermedia System (AHS), RIAs offer more fine grained opportunities for tracking user interactions, user modeling, and, finally, user-centric personalization. The vision of web applications adapting themselves in an ad-hoc fashion and on time to the individual user's context based on short-term user profiles has come within range.

Until now research on Adaptive Hypermedia (AH) focused on solutions adapting web applications on the server after the user explicitly requests a new web page. Before sending the requested page to the user it is altered according to some user profile. The three major restrictions of server-side adaptation approaches are: The necessity of explicit page requests for applying user-centric adaptations, restricted clickstream tracking possibilities and limited adaptation opportunities.

Our research aimed at overcoming of these restrictions and answering the research question of how to provide ad-hoc adaptivity to RIAs, which exploits their rich GUIs, while preserving their responsiveness. We conducted our research according to the design science research methodology (Peffers et al., 2008).

We enhanced the state of the art in research on AH by researching the concepts of Adaptive and Reactive Rich Internet Applications (ARRIA). We developed the ARRIA Adaptation Engine which overcomes the restrictions of server-side adaptation approaches by following a client-side approach. The Adaptation Engine relies on two basic technologies contributing to the event-driven nature of RIAs: complex event processing (CEP) and production rule processing. We analyzed different approaches for detecting complex events. Based on the results of this analysis we decided to use a graph-based approach as proposed by Chakravarthy et al. (1994). As a pattern matching algorithm for implementing a client-side production rule system, we decided on the efficient Rete algorithm (Forgy, 1982). In the core of the ARRIA Adaptation Engine a newly developed algorithm is deployed that combines graph-based event processing with Rete networks. The new algorithm which is prototypically implemented by the Adaptation Engine

in JavaScript adds reactivity to production rule systems and thus ad-hoc adaptivity capabilities to RIAs. The ARRIA Adaptation Engine is capable of processing adaptations directly on the client on time. Furthermore, the client-side approach allows the tracking of the complete user clickstream even across application boundaries. Finally, the entire set of client-side Dynamic HTML (DHTML) animations and controls are accessible for manipulating the GUIs of web applications.

The event and rule processing capabilities of the ARRIA Adaptation Engine allow for the execution of declarative event condition action (ECA) rules. We designed Adaptation Rule Language (ARL), a new language for encoding adaptation rules declaratively, based on the ECA paradigm. ARL is a light weight declarative rule language for encoding user tracking, short-term user profiling and client-side ad-hoc adaptation. The outstanding feature of ARL is that adaptation rules can be encoded based on the chronological sequence of user actions enabled by the event algebra implemented in the event part of ECA rules. Moreover, ARL can be used as a general purpose rule language for encoding ECA rules that can be executed efficiently by web browsers. Because of ARL's JavaScript Object Notation (JSON) syntax web browsers can execute ARL rules very efficiently with only little parsing effort.

Furthermore, we developed a holistic framework covering not only run-time aspects of ARRIAs but also design-time aspects targeting the acquisition and the design of meaningful adaptation rules. We introduced a semantic web usage behavior mining approach which extended and adopted state of the art data mining algorithms to be used on semantic metadata. We showed how to annotate web applications with ontologies in order to gain semantically enhanced web server access log files which, in turn, provide the basis of our semantic data mining approach. Additionally, we demonstrated how microformats, embedded in an web application, or tags from, for instance, social bookmarking services provide an additional source of metadata for driving the personalization of web applications.

Finally, we demonstrated the wide applicability of ARRIAs by the examination of three use cases: personalized e-Government, personalized web advertisement and personalized web search. In the e-Government use case we demonstrated how to equip a formerly non-adaptive e-Government portal with ad-hoc adaptivity by using ARRIAs. We added adaptivity to the non-adaptive e-Government portal of the Austrian city of Vöcklabruck with only minor non-intrusive changes to the portal's original source code. In the subsequently conducted usability test we measured and compared the user experience of the old non-adaptive portal with the new ARRIA-enabled adaptive portal. The results of the usability test give evidence for our hypothesis that the usability of ARRIA-enabled web applications is higher than the usability of web applications offering no user guidance. The benefits of ARRIAs could also be shown by evaluating the second and third use case. The second use case applies ARRIAs to the do-

main of personalized web advertisement and the third use case applies ARRIAs to the domain of personalized web search. The last one is special in the sense that only parts of the ARRIA framework are applied, while at the same time the ARRIA approach is enhanced by cross application user tracking and by dynamic on-demand acquisition of non-formal metadata.

11.1 Ongoing and Future Work

In order to demonstrate the applicability of ARRIAs also for RIA technologies other than Asynchronous JavaScript and XML (Ajax) (cf. Section 2.1) we plan to implement the Run-Time Adaptation Framework of ARRIAs in Microsoft Silverlight. This is part of the OPEN project[1] which aims at the development of an environment providing people with the ability to continue to perform their tasks when they move around and change their interaction device. We investigate the use of ARRIAs in an emergency situation like fire or flooding. In this context we currently inspect different aspects of ARRIAs in order to provide context-sensitive services for the Performance Optimization Application (POA) Suite Foundation group of SAP's BusinessObjects Division.

Currently, we are also extending the personalized e-Government use case (cf. Section 3.1) by adaptive web surveys to measure the quality of portals and their online-services. The measurement of a portal's quality forms the basis of an iterative improvement process. We examine whether ARRIAs can be used for measuring user feedback concerning the quality of portals and their services. The envisioned system applies three axes of adaptation to a web survey: first based on feedback from users through questionnaires, second based on problems encountered by the user, and, finally, based on metadata of the pages visited by the user. The research results of this work will be published in Magoutas et al. (2010).

For the personalized web advertisement (cf. Section 3.2) we envision the realization of more complex event patterns to detect a user's behavior also on a higher level of abstraction. For instance, a user may look at particular items on a web page, then switch to another related item, and go back to the first one. In this particular situation, a user is likely comparing those two items. Other navigational patterns are conceivable.

As an extension of the personalized web search use case (cf. Section 3.3) we are investigating the applicability of ARRIAs to an enterprise search scenario. Consistent and efficient search in a company's ever increasing knowledge sources becomes increasingly important for today's enterprise information workers. However, the heterogeneity of such knowledge sources makes this task very difficult compared to conventional web search. One major difference is the diversity of

[1] Open Pervasive Environments for migratory iNteractive services (OPEN) is a VII Framework EU STREP project.

content sources and formats in enterprises as well as the need for personalized search based on the current context of the information worker. Preliminary results of our research have already been published in Schmidt et al. (2008c).

Further ongoing work relates to SAP's product portfolio addressing small and medium enterprises which is delivered as software as a service (SaaS) over the Internet. SaaS is a software application delivery model where a software vendor develops a web-native software application and hosts and operates the application for use by its customers over the Internet. SaaS adds new requirements like monitoring and metering, to the provision of software. To ensure the quality and availability of services, the hosted application instances must run reliably. In order to achieve this, all instances must be monitored. Further more, the system needs to be prophylactic, adapting to changing conditions, and capable of indicating forthcoming execution problems. The performance and availability of mission-critical applications are impacted by several interconnected factors (system resources, application architecture, behavior of application code, network infrastructure, usage, etc.). When problems occur, finding the bottlenecks in such a distributed system can be a complex, lengthy, and costly process. However, failure detection and maintenance can be improved by using a server-side implementation of ARRIA's Hybrid Inference Engine (cf. Section 7.2). Furthermore, the Hybrid Discrimination Network of the Hybrid Inference Engine provides a scalable solution, which not only correlates simple events to complex events, but also relates complex events to their triggered actions and the application contexts in which they were issued. First results have already been published in Anicic et al. (2008).

ARRIAs showed their high potential for providing personalized RIAs. Especially, ARRIA's declarative and reactive approach, and the extensive use of metadata for personalization makes it a good candidate for shaping future user-centric and adaptive web applications.

Part IV

Appendix

Appendix A

Grammar of the ARRIA Rule Language

```
001 /*------------------------------------------------------------------
002  *  Copyright 2008-2009 Kay-Uwe Schmidt, Roland Stühmer
003  *
004  *  Licensed under the Apache License, Version 2.0 (the "License");
005  *  you may not use this file except in compliance with the License.
006  *  You may obtain a copy of the License at
007  *
008  *     http://www.apache.org/licenses/LICENSE-2.0
009  *
010  *  Unless required by applicable law or agreed to in writing,
011  *  software distributed under the License is distributed on an
012  *  "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND,
013  *  either express or implied. See the License for the specific
014  *  language governing permissions and limitations under the License.
015  *------------------------------------------------------------------*/
016
017 grammar AdaptationRulesLanguage;
018
019 // parser rules
020 rulesFile
021   : LBrace
022       ( prelude Comma )?
023       ( eventsRepository Comma )?
024       ( conditionsRepository Comma )?
025       ( actionsRepository Comma )?
026       ( widgetsRepository Comma )?
027       rulesRepository
028     RBrace
029   ;
```

```
030
031 prelude
032   : '"PRELUDE"' Colon keyValueParams
033   ;
034
035 keyValueParams
036   : LBrace ( keyValueParam ( Comma keyValueParam )* )? RBrace
037   ;
038
039 keyValueParam
040   : id Colon value
041   ;
042
043 id
044   : String
045   ;
046
047 eventsRepository
048   : '"EVENTS_REPOSITORY"' Colon
049     LBrace
050       ( id Colon event
051       ( Comma id Colon event )* )?
052     RBrace
053   ;
054
055 conditionsRepository
056   : '"CONDITIONS_REPOSITORY"' Colon
057     LBrace
058       ( conditionSpecification
059       ( Comma conditionSpecification )* )?
060     RBrace
061   ;
062
063 conditionSpecification
064   : id Colon
065     LBracket condition ( Comma condition )* RBracket
066   ;
067
068 actionsRepository
069   : '"ACTIONS_REPOSITORY"' Colon
070     LBrace
071       ( actionSpecification
072       ( Comma actionSpecification )* )?
073     RBrace
074   ;
```

```
075
076 actionSpecification
077   : id Colon
078     LBracket action ( Comma action )* RBracket
079   ;
080
081 widgetsRepository
082   : '"WIDGETS_REPOSITORY"' Colon keyValueParams
083   ;
084
085 rulesRepository
086   : '"RULES_REPOSITORY"' Colon ruleSet
087   ;
088
089 ruleSet
090   : LBracket
091       ( adaptationRule ( Comma adaptationRule )* )?
092     RBracket
093   ;
094
095 adaptationRule
096   : LBrace
097       ( prelude Comma )?
098       ( eventPart Comma )?
099       ( conditionPart Comma )?
100       actionPart
101     RBrace
102   ;
103
104 eventPart
105   : '"ON"' Colon event
106   ;
107
108 event
109   : id | simpleEvent | complexEvent
110   ;
111
112 simpleEvent
113   : eventType | temporal | dom
114   ;
115
116 eventType
117   : LBrace '"TYPE"' Colon String RBrace
118   ;
119
```

```
120 temporal
121   : LBrace
122       '"TYPE"' Colon '"TEMPORAL"'
123       ( Comma '"TIME"' Colon String )?
124       ( Comma '"DURATION"' Colon String )?
125     RBrace
126   ;
127
128 dom
129   : LBrace
130       '"TYPE"' Colon '"DOM"'
131       Comma '"SELECTOR"' Colon
132         LBracket String ( Comma String )* RBracket
133       Comma '"EVENT"' Colon String
134     RBrace
135   ;
136
137 complexEvent
138   : LBrace
139       '"OPERATOR"' Colon
140         ( or | and | any | not
141           | seq | a | aAsterisk | p | pAsterisk | plus )
142     RBrace
143   ;
144
145 and
146   : '"AND"'
147     Comma '"CONSTITUENT_EVENTS"' Colon
148       LBracket event Comma event RBracket
149   ;
150
151 or
152   : '"OR"'
153     Comma '"CONSTITUENT_EVENTS"' Colon
154       LBracket event Comma event RBracket
155   ;
156
157 any
158   : '"ANY"'
159     Comma '"M"' Colon Integer
160     Comma '"CONSTITUENT_EVENTS"' Colon
161       LBracket event ( Comma event )* RBracket
162   ;
163
164 not
```

```
165   : '"NOT"'
166     Comma '"CONSTITUENT_EVENTS"' Colon
167       LBracket temporal Comma event Comma temporal RBracket
168   ;
169
170 seq
171   : '"SEQ"'
172     Comma '"CONSTITUENT_EVENTS"' Colon
173       LBracket event Comma event RBracket
174   ;
175
176 a
177   : '"A"'
178     Comma '"CONSTITUENT_EVENTS"' Colon
179       LBracket event Comma event Comma event  RBracket
180   ;
181
182 aAsterisk
183   : '"A*"'
184     Comma '"CONSTITUENT_EVENTS"' Colon
185       LBracket event Comma event Comma event  RBracket
186   ;
187
188 p
189   : '"P"'
190     Comma '"CONSTITUENT_EVENTS"' Colon
191       LBracket event Comma temporal Comma event RBracket
192   ;
193
194 pAsterisk
195   : '"P*"'
196     Comma '"CONSTITUENT_EVENTS"' Colon
197       LBracket event Comma temporal Comma event RBracket
198   ;
199
200 plus
201   : '"PLUS"'
202     Comma '"CONSTITUENT_EVENTS"' Colon
203       LBracket event Comma temporal RBracket
204   ;
205
206 conditionPart
207   : '"IF"' Colon conditions
208   ;
209
```

```
210 conditions
211   :  LBracket ( condition ( Comma condition )* )? RBracket
212   ;
213
214 condition
215   : id
216     | LBrace
217       ( script | declExpr )
218     RBrace
219   ;
220
221 declExpr
222   : '"PREDICATE"' Colon ( predefinedRelation | String )
223     Comma '"SUBJECT"' Colon ( Variable | id )
224     ( Comma script )?
225     Comma '"OBJECT"' Colon ( Variable | value )
226     ( Comma script )?
227     ( Comma '"OPERATOR"' Colon relationalOperator )?
228   ;
229
230 predefinedRelation
231   : '"instanceOf"' | '"sameAs"' | '"differentFrom"'
232   ;
233
234 script
235   : '"SCRIPT"' Colon
236       LBracket ( String ( Comma String )* )? RBracket
237   ;
238
239 relationalOperator
240   : '"=="' | '"!="' | '">="' | '">"' | '"<="' | '"<"'
241     | '"==="' | '"!=="'
242   ;
243
244 actionPart
245   : '"DO"' Colon actions
246   ;
247
248 actions
249   : LBracket ( action ( Comma action )* )? RBracket
250   ;
251
252 action
253   : id | actionObject
254   ;
```

```
255
256 actionObject
257   : LBrace
258       ( script | fireEvent | assert | modify | retract )
259     RBrace
260   ;
261
262 fireEvent
263   : '"TRIGGER"' Colon String
264     Comma '"PARAMETERS"' Colon keyValueParams
265   ;
266
267 assert
268   : '"ASSERT"' Colon id
269     Comma '"CLASS"'  Colon String
270     ( Comma '"OBJECT"' Colon jsonObject )?
271     ( Comma script )?
272   ;
273
274 modify
275   : '"MODIFY"' Colon
276       ( ( ( Variable | id ) Comma script )
277         | ( LBrace declExpr RBrace ) )
278   ;
279
280 retract
281   : '"RETRACT"' Colon ( Variable | id )
282   ;
283
284 /*-------------------------------------------------------------------
285  *  Copyright 2007 Taro L. Saito
286  *
287  *  Licensed under the Apache License, Version 2.0 (the "License");
288  *  you may not use this file except in compliance with the License.
289  *  You may obtain a copy of the License at
290  *
291  *     http://www.apache.org/licenses/LICENSE-2.0
292  *
293  *  Unless required by applicable law or agreed to in writing,
294  *  software distributed under the License is distributed on an
295  *  "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND,
296  *  either express or implied. See the License for the specific
297  *  language governing permissions and limitations under the License.
298  *-------------------------------------------------------------------*/
299
```

```
300 // parser rules
301 jsonObject
302   : object
303   ;
304
305 jsonArray
306   : array
307   ;
308
309
310 object
311   : LBrace ( objectElement ( Comma objectElement )* )? RBrace
312   ;
313
314 objectElement
315   : String Colon value
316   ;
317
318 array
319   : LBracket value (Comma value)* RBracket
320   ;
321
322 value
323   : String
324   | Integer
325   | Double
326   | object
327   | array
328   | TRUE
329   | FALSE
330   | NULL
331   ;
332
333 // lexer rules
334 Colon: ':';
335 Comma: ',';
336 LBrace: '{';
337 RBrace: '}';
338 LBracket: '[';
339 RBracket: ']';
340 fragment Dot: '.';
341 TRUE:  'true';
342 FALSE: 'false';
343 NULL: 'null';
344
```

```
345 fragment Digit: '0' .. '9';
346 fragment HexDigit: ( '0' .. '9' | 'A' .. 'F' | 'a' .. 'f' );
347 fragment UnicodeChar: ~( '"' | '\\' );
348 fragment StringChar :  UnicodeChar | EscapeSequence;
349
350 fragment EscapeSequence
351   : '\\' ( '\"' | '\\' | '/' | 'b' | 'f' | 'n' | 'r' | 't' | 'u'
352     HexDigit HexDigit HexDigit HexDigit )
353   ;
354
355 fragment Int: '-'? ( '0' | '1'..'9' Digit* );
356 fragment Frac: Dot Digit+;
357 fragment Exp: ( 'e' | 'E' ) ( '+' | '-' )? Digit+;
358
359 WhiteSpace
360   : ( ' ' | '\r' | '\t' | '\u000C' | '\n' ) { $channel=HIDDEN; }
361   ;
362
363 String: '"' ~('?') StringChar* '"';
364 Variable: '"' '?' StringChar* '"';
365
366 Integer: Int;
367 Double:  Int ( Frac Exp? | Exp );
```

Appendix B

Usability Test Results

The tables in this appendix list the results of the usability test. The usability test was conducted from the 6^{th} October 2008 until the 21^{st} October 2008. Forty-nine people out of 215 participated in this evaluation.

Each row in each table represents the implicit and explicit inputs of just one test user. The first column of every table is a unique identifier. The identifiers are consistent across all tables and reflect the sequence in which the test users started the evaluation. Gray columns represent the use of the static version of the city portal of Vöcklabruck. Non highlighted rows show user inputs based on the adaptive version of the city portal. ✓ as a column value means YES and ✗ means NO with regard to the table header. For instance in the German column of Table B.1 ✓ means that a test person's mother tongue is German whereas ✗ states that the test person's mother tongue is not German. *n/a* in a table's column stands for 'not applicable'.

The Table B.1 shows the demographical data together with the survey results. The table part with the demographical data is captioned with Personal Data. It shows shows the age, the gender, whether or not the test user's mother tongue is German, how intensive he/she uses the Internet and finally whether he/she is an expert in building law. The second part of the table captioned with Questionnaire shows the results of the conclusive survey. It shows the subjective perception of the usability of the test portal. The overall contentedness of the test user with the portal is shown in the first column, the second displays the answers given to the question how easy it is to find the form pages in the portal, and, finally, the third column states weather or not the test person prefers the portal to a mortar-and-brick office.

The Table B.2 lists the free text comments of the test users given during the survey. These comments belong conceptually to the Questionnaire table above but as only some users filled in the commend field we decided to put them into an extra table for the sake of clarity.

The Table B.3 provides an aggregated view on the quantitative measures of the usability evaluation. The Duration column presents the accumulated time

for processing all three test cases. The next column entitled with Correctness displays whether or not all test cases were solved correctly. The Wizard column shows whether or not the Construction wizard was used in all three test cases. The next two columns report about whether or not a search or help page was called during the test.

The Table B.4, Table B.5 and Table B.6 list the quantitative measures of all test persons for all three test cases in detail. The Time column shows the time needed to fulfill the corresponding task. The next column entitled with Answer shows which form has been selected. The name of the right form representing the solution of the corresponding task is displayed in the next column. The last column lists the recorded user behavior for this task as Clickstream.

Table B.1: Personal Data and Questionnaire

	Personal Data					**Questionnaire**		
No	Age	Gender	German	Internet Use	Expert	Contentedness	Ease of Finding Forms	Portal Preferred
1	25-35	Female	✓	Intensive	✗	Very satisfied	Very easy	✓
2	under 25	Female	✓	Often	✗	Enthusiastic	Very easy	✓
3	under 25	Female	✓	Often	✗	Satisfied	Hard	✗
4	25-35	Male	✓	Intensive	✗	Very satisfied	Easy	✓
5	under 25	Male	✓	Often	✗	Enthusiastic	Very easy	✓
6	36-45	Female	✓	Intensive	✗	Satisfied	Satisfactory	*n/a*
7	under 25	Female	✓	Often	✗	Enthusiastic	Easy	✓
8	25-35	Male	✓	Intensive	✗	Very satisfied	Very easy	✓
9	25-35	Female	✓	Often	✗	Unsatisfied	Hard	✗
10	under 25	Male	✓	Intensive	✗	Enthusiastic	Very easy	✓
11	25-35	Female	✓	Intensive	✗	Satisfied	Satisfactory	✓
12	36-45	Male	✓	Intensive	✗	Unsatisfied	Hard	✗
13	36-45	Male	✓	Intensive	✗	Very satisfied	Very easy	✓
14	25-35	Male	✓	Often	✗	Very satisfied	Very easy	✗
15	under 25	Male	✓	Intensive	✗	Very satisfied	Very easy	✓
16	under 25	Male	✓	Intensive	✗	Very satisfied	Very easy	✓
17	25-35	Male	✓	Intensive	✗	Unsatisfied	Satisfactory	✗
18	25-35	Male	✗	Intensive	✗	Unsatisfied	Satisfactory	✓
19	25-35	Male	✓	Intensive	✗	Very satisfied	Very easy	✓
20	25-35	Male	✓	Intensive	✗	Satisfied	Very easy	✓
21	25-35	Male	✓	Intensive	✗	Satisfied	Easy	✓
22	25-35	Female	✓	Intensive	✗	Very satisfied	Easy	✓
23	25-35	Male	✓	Often	✗	Satisfied	Satisfactory	✗
24	25-35	Male	✓	Intensive	✗	Satisfied	Satisfactory	✗

	Personal Data					**Questionnaire**		
No	Age	Gender	German	Internet Use	Expert	Contentedness	Ease of Finding Forms	Portal Preferred
25	under 25	Male	✓	Intensive	✗	Very satisfied	Very easy	✓
26	over 45	Male	✓	Intensive	✗	Satisfied	Easy	✗
27	25-35	Male	✓	Intensive	✗	Satisfied	Easy	✓
28	25-35	Female	✗	Intensive	✗	Unsatisfied	Hard	✗
29	36-45	Male	✓	Intensive	✗	*n/a*	*n/a*	✗
30	36-45	Female	✓	Intensive	✗	Unsatisfied	Hard	✗
31	25-35	Female	✓	Intensive	✗	Unsatisfied	Satisfactory	✗
32	under 25	Male	✓	Intensive	✗	Enthusiastic	Very easy	✓
33	25-35	Male	✓	Intensive	✗	Very satisfied	Very easy	✓
34	25-35	Female	✗	Intensive	✗	Enthusiastic	Very easy	✓
35	36-45	Male	✓	Often	✗	Satisfied	Easy	✓
36	36-45	Male	✓	Intensive	✗	Very satisfied	Very easy	✓
37	25-35	Male	✓	Intensive	✗	Satisfied	Easy	✓
38	under 25	Male	✓	*n/a*	✗	Enthusiastic	Easy	*n/a*
39	*n/a*	*n/a*	*n/a*	*n/a*	*n/a*	Very satisfied	*n/a*	*n/a*
40	25-35	Male	✓	Intensive	✗	Satisfied	Easy	✓
41	36-45	Male	✗	Intensive	✗	Satisfied	Satisfactory	✓
42	36-45	Female	✗	Intensive	✗	Very satisfied	Very easy	✓
43	over 45	Male	✓	Often	✗	Very satisfied	Very easy	✓
44	25-35	Male	✓	Intensive	✗	Satisfied	Satisfactory	✗
45	36-45	Male	✓	Intensive	✗	Unsatisfied	Hard	✗
46	25-35	Male	✓	Often	✗	Very satisfied	Easy	✓
47	25-35	Female	✗	Intensive	✗	Satisfied	Satisfactory	✓
48	25-35	Female	✓	Intensive	✗	Enthusiastic	Very easy	✓
49	25-35	Female	✓	Intensive	✗	Enthusiastic	Very easy	✓

Table B.2: [Test users comments]Test users comments.

No	Comment
6	Letzteres hängt von meiner Vorinformation ab.
11	Das Formular Abbruchanzeige war nicht verfügbar.
16	Vielleicht sollte man auf die Grenzwerte hingewiesen werden, falls man sich noch gar nicht sicher ist wie hoch/breit man genau bauen will.
17	Ich war irritiert, dass der Link auf das Formular mit der Abrissanzeige gefehlt hat.
19	Vorziehen, ja. Allerdings bleibt unklar, wie die wesentlich komplexeren Folgeprozesse der Bauantragsabwicklung online durchgeführt werden.
20	Ich bin überhaupt nicht sicher ob ich den 'passenden Bauantrag' gefunden habe. Eigentlich habe ich immer nur einen Bauantrag gesehen. Die Information für Bürger war ein PDF und das wollte ich nicht lesen, so dass ich nun wohl in Unkenntnis sterben werde ;) Mir war völlig unklar, wozu viel der in der Aufgabe genannten Informationen nützlich gewesen wären (z.B. 4m hoch?).
21	Die Auswahl der Arten von Bauvorhaben war angenehm kurz. Das erscheint mir wie ein unrealistischer Testfall. Wenn die gleiche Liste statt 5 mehr als 20 Einträge hätte, wäre es viel schwerer zu benutzen.
23	Wenn ich wüsste, ob ich die Aufgaben "richtig" gelöst habe, würde mir die Beantwortung obiger Fragen leichter fallen.
27	Der Bauantragsassistent hätte einen Link DIREKT auf der Startseite verdient.
29	Ich habe keine Ahnung, was ich da überhaupt gemacht habe!
30	Sorry, aber ich habe gar nicht gemerkt, ob es unterschiedliche Anträge gab, bzw. bei der Suche nach "Doppelgarage Abriss" über das gesamte Portal kam der Link zur Formularübersicht, aber innerhalb der Formulare keine überschrift mit "Abriss oder Doppelgarage". Somit wusste ich jetzt nicht welches Formular ich nehmen muss. Ich habe dann irgendeins ausgewählt, allerdings dann nicht gesehen, wie dies aussieht um entscheiden zu können, ob dies das richte war. Somit bin ich entweder die falsche Userin oder war einfach zu schnell.
31	Ich habe nur den Bauantrag-Assistenten verwendet. Bei der zweiten Aufgabe war ich mir unsicher u. wollte mehr Informationen finden. Diese konnte ich jedoch auf dem Portal nicht finden. Was mich bei dem Assistenten verunsichert hat, ist eine fehlende Begründung, z.B. Schwimmbad ist ohne Antrag zu bauen unabhängig der sonst verbreiteten "m^2 Regel", weil... oder Garage benötigt Abrissantrag, weil...
35	Ich war mir nie sicher, ob es der richtige Antrag ist. Da fehlt mir ein Beispieltext oder eine Hilfezeile. Aber wenn man diese Unsicherheit klären kann, ist das Portal vorzuziehen.
41	Ohne Bauassistent wäre es unmöglich gewesen.

No	Comment
44	Leider war kein Formular für eine Abbruchgenehmigung zu finden. Darüber hinaus wären kurze Informationen zu den Formularen auf der Formularhauptseite wünschenswert. Sonst muss erst das PDF gefunden werden.
46	So lange es sich um "straight foreward" Bauvorhaben (wie im Beispiel) handelt, ja... leider ist es nicht immer ganz so einfach, aber mal testen ob das System zu einem vernünftigen Ergebnis kommt kostet nicht so viel Zeit, dass man es nicht zumindest probieren könnte!

Table B.3: Summary

No	Duration	Correctness	Wizard	Serach	Help
1	01:10.032	✓	✓	✗	✗
2	00:52.265	✓	✓	✗	✗
3	00:49.604	✗	✗	✗	✗
4	02:36.934	✗	✗	✗	✗
5	01:22.392	✗	✗	✗	✗
6	00:14.758	✗	✗	✗	✗
7	02:22.761	✓	✓	✗	✗
8	01:30.503	✓	✓	✗	✗
9	02:12.473	✗	✗	✗	✓
10	02:16.705	✗	✓	✗	✗
11	06:48.287	✗	✗	✓	✓
12	01:27.790	✗	✗	✗	✗
13	01:50.437	✓	✓	✗	✗
14	01:01.032	✓	✓	✗	✗
15	02:21.012	✓	✓	✗	✗
16	02:21.012	✓	✓	✗	✗
17	05:52.074	✗	✗	✗	✓
18	06:00.647	✗	✗	✗	✗
19	03:44.705	✓	✗	✗	✓
20	01:42.483	✗	✗	✗	✓
21	03:03.522	✓	✓	✓	✗
22	00:55.123	✓	✓	✗	✗
23	04:38.011	✗	✗	✓	✗
24	03:20.868	✗	✗	✗	✓
25	02:11.735	✓	✓	✗	✗
26	01:44.474	✓	✓	✗	✗
27	01:36.831	✓	✗	✗	✗
28	02:03.719	✗	✗	✗	✗
29	03:28.048	✗	✗	✗	✓
30	05:47.561	✗	✗	✓	✓
31	03:23.049	✓	✓	✗	✗

No	Duration	Correctness	Wizard	Serach	Help
32	00:56.952	✓	✓	✗	✗
33	01:25.503	✓	✓	✗	✗
34	03:24.430	✓	✓	✓	✗
35	01:10.756	✗	✗	✗	✗
36	01:49.142	✓	✓	✗	✗
37	01:08.120	✗	✗	✗	✗
38	05:01.357	✗	✗	✗	✓
39	01:28.689	✗	✗	✗	✗
40	01:11.658	✓	✓	✗	✗
41	03:57.280	✓	✓	✗	✗
42	01:26.505	✓	✓	✗	✗
43	01:25.491	✓	✓	✗	✗
44	03:46.169	✗	✗	✗	✓
45	02:49.007	✗	✗	✓	✗
46	01:30.676	✓	✓	✗	✗
47	02:17.587	✗	✓	✗	✗
48	01:57.940	✓	✓	✗	✗
49	00:45.842	✓	✓	✗	✗

Table B.4: Task 1

No	Time	Answer	Clickstream
	min:sec.msec	Construction Permit	
1	00:22.360	Construction Permit	index.html, buergerservice.html?showWizard=1
2	00:13.810	Construction Permit	index.html, buergerservice.html?showWizard=1
3	00:23.968	No Official Approval	index.html, buergerservice.html, buergerservice/formulare.html
4	01:36.577	Construction Note	index.html, bauen.html, buergerservice.html?showWizard=1, bauen.html, bauen/gebaeudemanagement.html, bauen.html, bauen/baurecht.html, bauen/form_baubewilligung.html, bauen/form_bauanzeige.html
5	00:29.829	Construction Permit	index.html, bauen.html, bauen/baurecht.html, bauen/form_baubewilligung.html
6	00:08.549	No Official Approval	index.html
7	01:25.587	Construction Permit	index.html, abgaben-gebuehren-verordnungen.html, abgaben-gebuehren-verordnungen/verordnungen.html, buergerservice.html?showWizard=1
8	00:39.970	Construction Permit	index.html, buergerservice.html?showWizard=1
9	00:33.805	Construction Permit	index.html, bauen.html, bauen/baurecht.html, bauen/form_baubewilligung.html
10	01:05.727	Construction Permit	index.html, bauen.html, buergerservice.html?showWizard=1
11	01:28.236	No Official Approval	index.html, suche.html, buergerservice/formulare.html
12	01:02.029	Construction Note	index.html, bauen.html, bauen/baurecht.html, bauen/form_bauanzeige.html
13	00:49.386	Construction Permit	index.html, buergerservice.html?showWizard=1
14	00:24.813	Construction Permit	index.html, buergerservice.html?showWizard=1
15	00:56.864	Construction Permit	index.html, buergerservice.html?showWizard=1
16	00:56.864	Construction Permit	index.html, buergerservice.html?showWizard=1

No	Time	Answer	Clickstream
	min:sec.msec	Construction Permit	
17	02:38.664	No Official Approval	index.html, bauen.html, buergerservice.html, buergerservice/formulare.html, bauen/form_baubewilligung.html, bauen/form_bauanzeige.html, buergerservice/leistungsangebot.html, buergerservice/leistungsangebot.html, buergerservice/formulare.html, http://83.65.190.84/STADTVB-FIT/BO/index.jsp
18	03:57.412	Construction Permit	index.html, bauen.html, bauen/gebaeudemanagement.html, wohnen-soziales.html, bauen.html, bauen/grundangelegenheiten.html, bauen/baurecht.html, bauen/form_baubewilligung.html
19	02:46.253	Construction Permit	index.html, bauen.html, bauen/baurecht.html, component/docman/doc_download/132-baurecht-detailinformationen.pdf, bauen/form_baubewilligung.html
20	00:44.745	Construction Permit	index.html, bauen.html, bauen/baurecht.html, bauen/form_baubewilligung.html
21	00:48.440	Construction Permit	index.html, suche.html
22	00:14.843	Construction Permit	index.html
23	02:38.169	Construction Permit	index.html, bauen.html, bauen/grundangelegenheiten.html, buergerservice.html, buergerservice/formulare.html, bauen/form_baubewilligung.html
24	00:42.811	Construction Permit	index.html, bauen.html, buergerservice.html?showWizard=1
25	00:51.724	Construction Permit	index.html, bauen.html, buergerservice.html?showWizard=1
26	01:01.828	Construction Permit	index.html, buergerservice.html?showWizard=1
27	00:54.407	Construction Permit	index.html, buergerservice.html?showWizard=1, bauen.html, buergerservice/formulare.html, bauen/form_baubewilligung.html
28	01:24.610	Construction Permit	index.html, bauen.html, bauen/baurecht.html, bauen/form_baubewilligung.html

No	Time	Answer	Clickstream
	min:sec.msec	Construction Permit	
29	01:50.532	Construction Permit	index.html, bauen.html, bauen/baurecht.html, bauen/form_baubewilligung.html, component/docman/doc_download/132-baurecht-detailinformationen.pdf, bauen/form_baubewilligung.html
30	04:53.917	Construction Permit	index.html, bauen.html, bauen/baurecht.html, bauen/form_bauanzeige.html, bauen/form_baubewilligung.html, component/docman/doc_download/132-baurecht-detailinformationen.pdf, suche.html, bauen/form_bauanzeige.html, suche.html, bauen/form_baubewilligung.html
31	00:50.329	Construction Permit	index.html, buergerservice.html?showWizard=1
32	00:15.969	Construction Permit	index.html, buergerservice.html?showWizard=1
33	00:49.174	Construction Permit	index.html, buergerservice.html?showWizard=1
34	02:02.550	Construction Permit	index.html, bauen.html, suche.html
35	00:43.641	Construction Permit	index.html, buergerservice.html, buergerservice/formulare.html, bauen/form_baubewilligung.html
36	00:35.876	Construction Permit	index.html, buergerservice.html?showWizard=1
37	00:47.504	Construction Permit	index.html, bauen.html, bauen/baurecht.html, bauen/form_baubewilligung.html
38	04:04.170	Construction Permit	index.html, bauen.html, bauen/baurecht.html, bauen/form_baubewilligung.html
39	00:27.049	Construction Permit	index.html, bauen.html, bauen/baurecht.html, bauen/form_baubewilligung.html
40	00:39.465	Construction Permit	index.html, buergerservice.html?showWizard=1
41	00:57.484	Construction Permit	index.html, buergerservice.html?showWizard=1
42	00:30.658	Construction Permit	index.html, buergerservice.html?showWizard=1
43	00:39.193	Construction Permit	index.html, buergerservice.html?showWizard=1
44	01:28.104	No Official Approval	index.html, buergerservice.html, buergerservice/formulare.html, bauen.html

No	Time	Answer	Clickstream
	min:sec.msec	Construction Permit	
45	02:30.733	Construction Permit	index.html, bauen.html, bauen/gebaeudemanagement.html, buergerservice.html, buergerservice/formulare.html, bauen/form_baubewilligung.html, bauen/form_bauanzeige.html, bauen/form_bauanzeige.html, bauen/form_bauanzeige.html, buergerservice/leistungsangebot.html, suche.html, bauen/form_baubewilligung.html
46	00:38.563	Construction Permit	index.html, buergerservice.html?showWizard=1
47	01:10.122	Construction Permit	index.html, buergerservice.html?showWizard=1
48	00:59.128	Construction Permit	index.html, buergerservice.html?showWizard=1
49	00:14.209	Construction Permit	index.html, buergerservice.html?showWizard=1

Table B.5: Task 2

No	Time	Answer	Clickstream
	min:sec.msec	No Official Approval	
1	00:27.532	No Official Approval	index.html, buergerservice.html?showWizard=1
2	00:23.063	No Official Approval	index.html, buergerservice.html?showWizard=1
3	00:16.746	Construction Note	index.html, buergerservice.html, buergerservice/formulare.html, bauen/form_bauanzeige.html
4	00:58.093	Construction Note	index.html, bauen.html, bauen/tiefbau.html, bauen/tiefbau.html, bauen/baurecht.html, bauen/form_bauanzeige.html
5	00:28.454	Construction Note	index.html, bauen.html, bauen/baurecht.html, bauen/form_bauanzeige.html
6	00:03.635	No Official Approval	index.html
7	00:35.229	No Official Approval	index.html, buergerservice.html?showWizard=1
8	00:37.845	No Official Approval	index.html, bauen.html, buergerservice.html?showWizard=1
9	01:17.837	Construction Permit	index.html, bauen.html, bauen/tiefbau.html, bauen/baurecht.html, bauen/form_baubewilligung.html, component/docman/doc_download/132-baurecht-detailinformationen.pdf, bauen/form_baubewilligung.html
10	00:53.382	Construction Note	index.html, buergerservice.html?showWizard=1
11	02:25.377	No Official Approval	index.html, buergerservice.html, buergerservice/formulare.html, buergerservice/formulare.html, bauen/form_baubewilligung.html, bauen.html, bauen.html, bauen/baurecht.html, component/docman/doc_download/132-baurecht-detailinformationen.pdf
12	00:14.183	Construction Note	index.html, bauen.html, bauen/baurecht.html, bauen/form_bauanzeige.html
13	00:39.192	No Official Approval	index.html, bauen.html, buergerservice.html?showWizard=1
14	00:21.531	No Official Approval	index.html, buergerservice.html?showWizard=1
15	00:54.432	No Official Approval	index.html, buergerservice.html?showWizard=1

No	Time	Answer	Clickstream
	min:sec.msec	No Official Approval	
16	00:54.432	No Official Approval	index.html, buergerservice.html?showWizard=1
17	00:18.169	Construction Permit	index.html, buergerservice.html, buergerservice/formulare.html, bauen/form_bauanzeige.html, bauen/form_baubewilligung.html
18	00:52.966	Construction Permit	index.html, bauen.html, bauen/baurecht.html, bauen/form_baubewilligung.html, bauen/baurecht.html, bauen/form_bauanzeige.html, bauen/form_baubewilligung.html
19	00:40.762	No Official Approval	index.html, bauen.html, buergerservice.html?showWizard=1
20	00:29.050	Construction Permit	index.html, bauen.html, bauen/baurecht.html, bauen/form_baubewilligung.html
21	01:02.167	No Official Approval	index.html, suche.html
22	00:24.375	No Official Approval	index.html, buergerservice.html?showWizard=1, buergerservice.html?showWizard=1, END_2, START_1, index.html, buergerservice.html?showWizard=1
23	00:42.254	Construction Permit	index.html, buergerservice.html, buergerservice/formulare.html, bauen/form_baubewilligung.html
24	02:18.182	Construction Permit	index.html, bauen.html, bauen/baurecht.html, component/docman/doc_download/132-baurecht-detailinformationen.pdf, bauen/form_bauanzeige.html, component/docman/doc_download/132-baurecht-detailinformationen.pdf
25	00:51.786	No Official Approval	index.html, bauen.html, buergerservice.html?showWizard=1
26	00:27.487	No Official Approval	index.html, buergerservice.html?showWizard=1
27	00:24.138	No Official Approval	index.html, buergerservice.html?showWizard=1
28	00:21.625	Construction Permit	index.html, bauen.html, bauen/baurecht.html, bauen/form_baubewilligung.html
29	00:54.375	Construction Permit	index.html, bauen.html, bauen/baurecht.html, bauen/form_baubewilligung.html
30	00:27.834	Construction Permit	index.html, bauen.html, bauen/baurecht.html, bauen/form_baubewilligung.html

No	Time	Answer	Clickstream
	min:sec.msec	No Official Approval	
31	02:05.252	No Official Approval	index.html, buergerservice.html?showWizard=1, bauen.html, bauen/raumplanung.html, bauen/grundangelegenheiten.html, bauen/gestaltungsbeirat.html, buergerservice.html?showWizard=1
32	00:22.319	No Official Approval	index.html, buergerservice.html?showWizard=1
33	00:22.895	No Official Approval	index.html, buergerservice.html?showWizard=1
34	00:48.934	No Official Approval	index.html, suche.html, suche.html
35	00:16.825	Construction Note	index.html, buergerservice.html, buergerservice/formulare.html, bauen/form_baubewilligung.html, bauen/form_bauanzeige.html
36	00:41.938	No Official Approval	index.html, buergerservice.html?showWizard=1
37	00:16.383	Construction Note	index.html, bauen.html, bauen/baurecht.html, bauen/form_baubewilligung.html, bauen/form_bauanzeige.html
38	00:22.609	Construction Permit	index.html, bauen.html, bauen/tiefbau.html, bauen/baurecht.html, bauen/form_baubewilligung.html
39	00:14.657	Construction Permit	index.html, bauen.html, bauen/baurecht.html, bauen/form_baubewilligung.html
40	00:17.824	No Official Approval	index.html, buergerservice.html?showWizard=1
41	01:39.078	No Official Approval	index.html, bauen.html, bauen.html, buergerservice.html?showWizard=1
42	00:39.955	No Official Approval	index.html, bauen.html, buergerservice.html?showWizard=1
43	00:27.849	No Official Approval	index.html, buergerservice.html?showWizard=1
44	02:07.297	Construction Note	index.html, buergerservice.html, buergerservice/formulare.html, bauen/form_baubewilligung.html, bauen/form_bauanzeige.html, buergerservice/leistungsangebot.html, bauen.html, bauen/baurecht.html, component/docman/doc_download/132-baurecht-detailinformationen.pdf, buergerservice.html, buergerservice/formulare.html, bauen/form_bauanzeige.html
45	00:11.153	Construction Permit	index.html, buergerservice.html, buergerservice/formulare.html, bauen/form_baubewilligung.html

No	Time	Answer	Clickstream
	min:sec.msec	No Official Approval	
46	00:34.645	No Official Approval	`index.html`, `buergerservice.html?showWizard=1`
47	00:44.502	No Official Approval	`index.html`, `bauen.html`, `buergerservice.html?showWizard=1`
48	00:33.828	No Official Approval	`index.html`, `buergerservice.html?showWizard=1`
49	00:18.305	No Official Approval	`index.html`, `buergerservice.html?showWizard=1`

Table B.6: Task 3

No	Time	Answer	Clickstream
	min:sec.msec	Construction Note	
1	00:20.140	Construction Note	index.html, buergerservice.html?showWizard=1
2	00:15.392	Construction Note	index.html, buergerservice.html?showWizard=1
3	00:08.890	Construction Permit	index.html, buergerservice.html, buergerservice/formulare.html, bauen/form_baubewilligung.html
4	00:02.264	No Official Approval	index.html
5	00:24.109	Construction Note	index.html, bauen.html, bauen/baurecht.html, bauen/form_bauanzeige.html
6	00:02.574	No Official Approval	index.html
7	00:21.945	Construction Note	index.html, buergerservice.html?showWizard=1
8	00:12.688	Construction Note	index.html, buergerservice.html?showWizard=1
9	00:20.831	Construction Permit	index.html, bauen.html, bauen/baurecht.html, bauen/form_baubewilligung.html
10	00:17.596	Construction Note	index.html, buergerservice.html?showWizard=1
11	02:54.674	Construction Permit	index.html, bauen.html, bauen/baurecht.html, component/docman/doc_download/132-baurecht-detailinformationen.pdf, bauen/form_baubewilligung.html
12	00:11.578	Construction Permit	index.html, bauen.html, bauen/baurecht.html, bauen/form_baubewilligung.html
13	00:21.859	Construction Note	index.html, buergerservice.html?showWizard=1
14	00:14.688	Construction Note	index.html, buergerservice.html?showWizard=1
15	00:29.716	Construction Note	index.html, buergerservice.html?showWizard=1
16	00:29.716	Construction Note	index.html, buergerservice.html?showWizard=1

No	Time	Answer	Clickstream
	min:sec.msec	Construction Note	
17	02:55.241	Construction Permit	index.html, bauen.html, bauen/baurecht.html, component/docman/doc_download/132-baurecht-detailinformationen.pdf, component/docman/doc_download/132-baurecht-detailinformationen.pdf, component/docman/doc_download/132-baurecht-detailinformationen.pdf, bauen/form_baubewilligung.html
18	01:10.269	Construction Permit	index.html, bauen.html, bauen/firmenhinweistafeln-auf-oeffentlichem-gut.html, bauen/gebaeudemanagement.html, bauen/gestaltungsbeirat.html, bauen/raumplanung.html, bauen/tiefbau.html, bauen/baurecht.html, bauen/form_baubewilligung.html
19	00:17.690	Construction Note	index.html, buergerservice.html?showWizard=1
20	00:28.688	Construction Permit	index.html, bauen.html, bauen/baurecht.html, component/docman/doc_download/132-baurecht-detailinformationen.pdf, bauen/form_baubewilligung.html
21	01:12.915	Construction Note	index.html, suche.html, bauen/form_baubewilligung.html, bauen/form_baubewilligung.html, bauen/form_bauanzeige.html, bauen/form_bauanzeige.html, suche.html
22	00:15.905	Construction Note	index.html, buergerservice.html?showWizard=1, buergerservice.html?showWizard=1
23	01:17.588	Construction Permit	index.html, buergerservice.html, buergerservice/formulare.html, bauen/form_baubewilligung.html, bauen/form_bauanzeige.html, bauen/form_bauanzeige.html, END_3, START_3, index.html, suche.html, bauen/form_baubewilligung.html
24	00:19.875	Construction Note	index.html, buergerservice.html?showWizard=1
25	00:28.225	Construction Note	index.html, bauen.html, buergerservice.html?showWizard=1
26	00:15.159	Construction Note	index.html, buergerservice.html?showWizard=1
27	00:18.286	Construction Note	index.html, buergerservice.html?showWizard=1

No	Time	Answer	Clickstream
	min:sec.msec	Construction Note	
28	00:17.484	Construction Permit	index.html, bauen.html, bauen/baurecht.html, bauen/form_baubewilligung.html
29	00:43.141	Construction Permit	index.html, bauen.html, bauen/baurecht.html, bauen/form_baubewilligung.html
30	00:25.810	Construction Permit	index.html, bauen.html, bauen/baurecht.html, buergerservice.html, buergerservice/formulare.html, bauen/form_baubewilligung.html
31	00:27.468	Construction Note	index.html, buergerservice.html?showWizard=1
32	00:18.664	Construction Note	index.html, buergerservice.html?showWizard=1, bauen.html, buergerservice.html?showWizard=1
33	00:13.434	Construction Note	index.html, buergerservice.html?showWizard=1
34	00:32.946	Construction Note	index.html, suche.html
35	00:10.290	Construction Permit	index.html, buergerservice.html, buergerservice/formulare.html, bauen/form_baubewilligung.html
36	00:31.328	Construction Note	index.html, buergerservice.html?showWizard=1
37	00:04.233	No Official Approval	index.html
38	00:34.578	Construction Permit	index.html, bauen.html, bauen/baurecht.html, component/docman/doc_download/132-baurecht-detailinformationen.pdf, bauen/form_bauanzeige.html, bauen/form_bauanzeige.html, bauen/form_baubewilligung.html
39	00:46.983	No Official Approval	index.html, bauen.html, bauen/baurecht.html
40	00:14.369	Construction Note	index.html, buergerservice.html?showWizard=1
41	01:20.718	Construction Note	index.html, bauen.html, bauen/baurecht.html, buergerservice.html?showWizard=1
42	00:15.892	Construction Note	index.html, buergerservice.html?showWizard=1
43	00:18.449	Construction Note	index.html, buergerservice.html?showWizard=1
44	00:10.768	Construction Permit	index.html, buergerservice.html, buergerservice/formulare.html, bauen/form_baubewilligung.html

No	Time	Answer	Clickstream
	min:sec.msec	Construction Note	
45	00:07.121	Construction Permit	index.html, buergerservice.html, buergerservice/formulare.html, bauen/form_baubewilligung.html
46	00:17.468	Construction Note	index.html, buergerservice.html?showWizard=1
47	00:22.963	No Official Approval	index.html, buergerservice.html?showWizard=1
48	00:24.984	Construction Note	index.html, buergerservice.html?showWizard=1
49	00:13.328	Construction Note	index.html, buergerservice.html?showWizard=1

Appendix C

ARRIA Search Extension Usability Study

The following tables show the questionnaire and the test results of the user-driven evaluation study of the ARRIA Search Extension described in Chapter 10.

Table C.1: Questionnaire: questions and scales.

No	Question	Scale
1	Did you use the ARRIA Search Extension?	1–Never *to* 5–Always
2	Did you experience the ARRIA Search Extension as useful?	Yes, No, Sometimes
3	How would you judge the quality of search results of the ARRIA Search Extension?	1–Very Poor *to* 5–Very Good
4	Would you recommend the ARRIA Search Extension?	Yes, No
5	Did you have any reservations regarding privacy?	1–Strong Reservations *to* 5–No Reservations
6	What did you think about the presentation of the personalized search results?	1–Very Disturbing *to* 5–Very Well Integrated
7	Comments	Free text input for additional remarks regarding the usability of the ARRIA Search Extension.

Table C.2: Question 1.

Never			Always	
1	2	3	4	5
-	2	-	1	1

Table C.3: Question 2.

Yes	No	Sometimes
3	-	1

Table C.4: Question 3.

Very Poor			Very Good	
1	2	3	4	5
-	-	1	3	-

Table C.5: Question 4.

Yes	No
4	-

Table C.6: Question 5.

Strong			No	
1	2	3	4	5
1	1	-	1	1

Table C.7: Question 6.

Disturbing			Integrated	
1	2	3	4	5
-	-	-	2	2

Table C.8: Question 7.

Free hand comments
"My reservations about the privacy issue concern the fact that the evaluator might have an extensive knowledge about the websites I searched for, which I might not want in certain cases. But fortunately, in these cases I can disable the user search, so that is not really a big concern on my side. Also, it would be nice if there was some kind of learning mechanism in place, so that the quality of the search results increases over time. Over all, from my perspective, the extension is quite useful and has great potential if enhanced with the already mentioned learning mechanism. It is easy to use and the suggested search results are clearly presented next to the 'normal' google results, without distracting me."
"Sometimes the 'suggestions frame' in the page is longer than the actual Google results, which makes you scroll more to get to the next page. Privacy problem (maybe there is a way of keep the privacy of the private content entered by the user...)."

Table C.9: Recorded user behavior.

		Follow-up Actions		
User	Google Views	Personalized Result	Native Result	No Action
1	39	2	9	28
2	55	1	48	6
3	68	1	30	37
4	30	2	25	5

Bibliography

Adaikkalavan, R., and Chakravarthy, S. (2006). SnoopIB: Interval-Based Event Specification and Detection for Active Databases. *Data Knowl. Eng.*, *59*(1), 139–165.

Adida, B., Birbeck, M., McCarron, S., and Pemberton, S. (2008). RDFa in XHTML: Syntax and Processing. Online resource, `http://www.w3.org/TR/rdfa-syntax`. Created on October 14, 2008. Accessed on November 10, 2009.

Agrawal, R., and Srikant, R. (1994). Fast Algorithms for Mining Association Rules in Large Databases. In J. B. Bocca, M. Jarke, and C. Zaniolo (Eds.) *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12–15, 1994, Santiago de Chile, Chile*, (pp. 487–499). Morgan Kaufmann.

Aleman-Meza, B., Halaschek-Wiener, C., Arpinar, I. B., and Sheth, A. P. (2003). Context-Aware Semantic Association Ranking. In I. F. Cruz, V. Kashyap, S. Decker, and R. Eckstein (Eds.) *Proceedings of SWDB'03, The first International Workshop on Semantic Web and Databases, Co-located with VLDB 2003, Humboldt-Universität, Berlin, Germany, September 7–8, 2003*, (pp. 33–50).

Allaire, J. (2002). Macromedia Flash MX — A next-generation rich client. Tech. rep., Macromedia.

Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Commun. ACM*, *26*(11), 832–843.

Anicic, D., Sen, S., Stojanovic, N., Ma, J., and Schmidt, K.-U. (2008). Contextualised Event-Triggered Reactivity with Similarity Search. In D. Anicic, C. Brelage, O. Etzion, and N. Stojanovic (Eds.) *iCEP2008: 1st International Workshop on Complex Event Processing for the Future Internet collocated with the Future Internet Symposium (FIS2008)*, vol. Vol-412. CEUR Workshop Proceedings (CEUR-WS.org, ISSN 1613-0073).

Armstrong, R., Freitag, D., Joachims, T., and Mitchell, T. (1995). WebWatcher - A Learning Apprentice for the World Wide Web. In *AAI Spring Symposium*. AAAI Spring Symposium.

Balasubramanian, V. (1994). State of the Art Review on Hypermedia Issues and Applications. Online resource, `http://www.stack.nl/~boch/hyper/index.html`. Updated August 1999. Accessed on February 5, 2009.

Baraglia, R., and Silvestri, F. (2007). Dynamic Personalization of Web Sites without User Intervention. *Commun. ACM*, *50*(2), 63–67.

Barnickel, N., Flügge, M., and Schmidt, K.-U. (2006). Interoperability in eGovernment through Cross-Ontology Semantic Web Service Composition. In *Workshop Semantic Web for eGovernment, 3rd European Semantic Web Conference 2006, Budva, Montenegro*.

Bass, T. (2007). Mythbusters: Event Stream Processing versus Somplex Event Processing. In H.-A. Jacobsen, G. Mühl, and M. A. Jaeger (Eds.) *Proceedings of the 2007 Inaugural International Conference on Distributed Event-Based Systems, DEBS 2007, Toronto, Ontario, Canada, June 20–22, 2007*, vol. 233 of *ACM International Conference Proceeding Series*, (p. 1). ACM.

Batory, D. (1994). The LEAPS Algorithms. Tech. rep., University of Texas at Austin, Austin, TX, USA.

Bechhofeer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D., Patel-Schneider, P., and Stein, L. A. (2004). OWL — Web Ontology Language Reference. Recommendation, W3C.

Benatallah, B., Casati, F., Georgakopoulos, D., Bartolini, C., Sadiq, W., and Godart, C. (Eds.) (2007). *Web Information Systems Engineering - WISE 2007, 8th International Conference on Web Information Systems Engineering, Nancy, France, December 3-7, 2007, Proceedings*, vol. 4831 of *Lecture Notes in Computer Science*. Springer.

Bercic, B., and Vintar, M. (2004). Simple Life-Events Ontology in SU(M)O-KIF. In M. Wimmer (Ed.) *Knowledge Management in Electronic Government, 5th IFIP International Working Conference, KMGov 2004, Krems, Austria, May 17–19, 2004, Proceedings*, vol. 3035 of *Lecture Notes in Computer Science*, (pp. 128–135). Springer.

Berendt, B., Hotho, A., Mladenic, D., van Someren, M., Spiliopoulou, M., and Stumme, G. (2004). A Roadmap for Web Mining: From Web to Semantic Web. In *Web Mining: From Web to Semantic Web*, vol. 3209, (pp. 1–22). Heidelberg: Springer.

Berendt, B., Hotho, A., and Stumme, G. (2002). Towards Semantic Web Mining. In I. Horrocks, and J. A. Hendler (Eds.) *The Semantic Web — ISWC*

2002, First International Semantic Web Conference, Sardinia, Italy, June 9–12, 2002, Proceedings, vol. 2342 of *Lecture Notes in Computer Science*, (pp. 264–278). Springer.

Berstel, B. (2002). Extending the Rete Algorithm for Event Management. In *Proc. Ninth International Symposium on Temporal Representation and Reasoning TIME 2002*, (pp. 49–51). Washington, DC, USA: IEEE Computer Society.

Borst, W. N. (1997). *Construction of Engineering Ontologies for Knowledge Sharing and Reuse.*. Doctoral Thesis of the University of Tweenty: Enschede, The Netherlands.

Bos, B., Çelik, T., Hickson, I., and Lie, H. W. (2009). Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification. World Wide Web Consortium, W3C Candidate Recommendation.

Bradley, K., Rafter, R., and Smyth, B. (2000). Case-Based User Profiling for Content Personalisation. In Brusilovsky et al. (2000a), (pp. 62–72).

Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., and Yergeau, F. (2008). Extensible Markup Language (XML) 1.0 (Fifth Edition). World Wide Web Consortium, Recommendation REC-xml-20081126.

Broughton, V. (2006). *Essential Thesaurus Construction*. Neal-Schuman.

Broughton, V., and Slavic, A. (2007). Building a Faceted Classification for the Humanities: Principles and Procedures. *Journal of Documentation*, *63*(5), 727–754.

Brownston, L., Farrell, R., Kant, E., and Martin, N. (1985). *Programming Expert Systems in OPS5: An Introduction to Rule-Based Programming*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Brusilovsky, P. (1996). Methods and Techniques of Adaptive Hypermedia. *User Model. User-Adapt. Interact.*, *6*(2–3), 87–129.

Brusilovsky, P. (1997). Efficient Techniques for Adaptive Hypermedia. In C. K. Nicholas, and J. Mayfield (Eds.) *Intelligent Hypertext: Advanced Techniques for the World Wide Web*, vol. 1326 of *Lecture Notes in Computer Science*, (pp. 12–30). Springer.

Brusilovsky, P. (2007). Adaptive Navigation Support. In Brusilovsky et al. (2007), (pp. 263–290).

Brusilovsky, P., Kobsa, A., and Nejdl, W. (Eds.) (2007). *The Adaptive Web, Methods and Strategies of Web Personalization*, vol. 4321 of *Lecture Notes in Computer Science*. Springer.

Brusilovsky, P., and Millán, E. (2007). User Models for Adaptive Hypermedia and Adaptive Educational Systems. In *The Adaptive Web*, (pp. 3–53).

Brusilovsky, P., Stock, O., and Strapparava, C. (Eds.) (2000a). *Adaptive Hypermedia and Adaptive Web-Based Systems, International Conference, AH 2000, Trento, Italy, August 28-30, 2000, Proceedings*, vol. 1892 of *Lecture Notes in Computer Science*. Springer.

Brusilovsky, P., Stock, O., and Strapparava, C. (2000b). Preface. In Brusilovsky et al. (2000a), (p. v).

Bry, F., and Eckert, M. (2006). Twelve Theses on Reactive Rules for the Web. In T. Grust, H. Höpfner, A. Illarramendi, S. Jablonski, M. Mesiti, S. Müller, P.-L. Patranjan, K.-U. Sattler, M. Spiliopoulou, and J. Wijsen (Eds.) *Current Trends in Database Technology - EDBT 2006, EDBT 2006 Workshops PhD, DataX, IIDB, IIHA, ICSNW, QLQP, PIM, PaRMA, and Reactivity on the Web, Munich, Germany, March 26–31, 2006, Revised Selected Papers*, vol. 4254 of *Lecture Notes in Computer Science*, (pp. 842–854). Springer.

Bry, F., and Eckert, M. (2007a). Rule-Based Composite Event Queries: The Language XChange^{EQ} and Its Semantics. In M. Marchiori, J. Z. Pan, and C. de Sainte Marie (Eds.) *Web Reasoning and Rule Systems, First International Conference, RR 2007, Innsbruck , Austria, June 7–8, 2007, Proceedings*, vol. 4524 of *Lecture Notes in Computer Science*, (pp. 16–30). Springer.

Bry, F., and Eckert, M. (2007b). Towards Formal Foundations of Event Queries and Rules. In *Proceedings of the Second Int. Workshop on Event-Driven Architecture, Processing and Systems, Proceedings of 33rd International Conference on Very Large Data Bases, Vienna, Austria (23rd–27th September 2007)*.

Bry, F., and Patranjan, P.-L. (2005). Reactivity on the Web: Paradigms and Applications of the Language XChange. In H. Haddad, L. M. Liebrock, A. Omicini, and R. L. Wainwright (Eds.) *Proceedings of the 2005 ACM Symposium on Applied Computing (SAC), Santa Fe, New Mexico, USA, March 13–17, 2005*, (pp. 1645–1649). ACM.

Carughi, G. T., Comai, S., Bozzon, A., and Fraternali, P. (2007). Modeling Distributed Events in Data-Intensive Rich Internet Applications. In Benatallah et al. (2007), (pp. 593–602).

Casteleyn, S., Daniel, F., Dolog, P., Matera, M., Houben, G.-J., and Troyer, O. D. (Eds.) (2007). *Proceedings of the 2nd International Workshop on Adaptation*

and Evolution in Web Systems Engineering AEWSE'07, Como, Italy, July 19, 2007, vol. 267 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Center for Democracy and Technology, and InfoDev (2002). The E-Government Handbook For Developing Countries. The World Bank Group, Online resource, `http://www.cdt.org/egov/handbook/2002-11-14egovhandbook.pdf`. Created on November 14, 2002. Accessed on March 27, 2009.

Chakravarthy, S. (1997). SENTINEL: An Object-Oriented DBMS With Event-Based Rules. In J. Peckham (Ed.) *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13–15, 1997, Tucson, Arizona, USA*, (pp. 572–575). ACM Press.

Chakravarthy, S., Krishnaprasad, V., Anwar, E., and Kim, S. K. (1994). Composite Events for Active Databases: Semantics, Contexts and Detection. In J. B. Bocca, M. Jarke, and C. Zaniolo (Eds.) *20th International Conference on Very Large Data Bases, September 12–15, 1994, Santiago, Chile proceedings*, (pp. 606–617). Los Altos, CA 94022, USA: Morgan Kaufmann Publishers.

Chen, L., and Sycara, K. P. (1998). WebMate: A Personal Agent for Browsing and Searching. In *Agents*, (pp. 132–139).

Chen, M.-S., Park, J. S., and Yu, P. S. (1998). Efficient Data Mining for Path Traversal Patterns. *IEEE Trans. on Knowl. and Data Eng.*, *10*(2), 209–221.

Cimiano, P., and Völker, J. (2005). Text2Onto. In A. Montoyo, R. Muñoz, and E. Métais (Eds.) *Natural Language Processing and Information Systems, 10th International Conference on Applications of Natural Language to Information Systems, NLDB 2005, Alicante, Spain, June 15–17, 2005, Proceedings*, vol. 3513 of *Lecture Notes in Computer Science*, (pp. 227–238). Springer.

Conklin, J. (1987). Hypertext: An Introduction and Survey. *Computer*, *20*(9), 17–41.

Cooley, R., Mobasher, B., and Srivastava, J. (1997). Web Mining: Information and Pattern Discovery on the World Wide Web. In *ICTAI*, (pp. 558–567).

Cooley, R., Mobasher, B., and Srivastava, J. (1999). Data Preparation for Mining World Wide Web Browsing Patterns. *Knowledge and Information Systems*, *1*(1), 5–32.

Cooper, A. (1999). The Inmates are Running the Asylum. In U. Arend, E. Eberleh, and K. Pitschke (Eds.) *Software-Ergonomie '99, Design von Informationswelten, Gemeinsame Fachtagung des German Chapter of the ACM, der Gesellschaft für Informatik (GI) und der SAP AG, Walldorf, 8.–11. März 1999, Walldorf/Baden*, vol. 53 of *Berichte des German Chapter of the ACM*, (p. 17). Teubner.

Corcho, Ó., Fernández-López, M., and Gómez-Pérez, A. (2003). Methodologies, Tools and Languages for Building Ontologies: Where is their Meeting Point? *Data Knowl. Eng.*, *46*(1), 41–64.

Craig, J., Cooper, M., Pappas, L., Schwerdtfeger, R., and Seeman, L. (2009). Accessible Rich Internet Applications (WAI-ARIA) Version 1.0. World Wide Web Consortium, Working Draft WD-wai-aria-20090224. Online resource, `http://www.w3.org/TR/2009/WD-wai-aria-20090224`. Created on February 24, 2009. Accessed on November 10, 2009.

Cranor, L. F. (2004). 'I didn't buy it for myself': Privacy and Ecommerce Personalization. *Designing personalized user experiences in eCommerce*, (pp. 57–73).

Cristofor, D., Cristofor, L., and Simovici, D. A. (2000). Galois Connections and Data Mining. *J. UCS*, *6*(1), 60–73.

Crockford, D. (2006a). JSON: The Fat-Free Alternative to XML. In *Proceedings of XML 2006*.

Crockford, D. (2006b). RFC4627: JavaScript Object Notation. Tech. rep., IETF.

Culbert, C., Riley, G., and Donnell, B. (1993). CLIPS Reference Manual Volume 1, Basic Programming Guide, CLIPS Version 6.0. *Software Technology Branch, Lyndon B. Johnson Space Center, NASA*.

Dai, H., and Mobasher, B. (2002). Using Ontologies to Discover Domain-Level Web Usage Profiles. In *2nd Semantic Web Mining Workshop at ECML/PKDD-2002*.

Dai, H., and Mobasher, B. (2004). Integrating Semantic Knowledge with Web Usage Mining for Personalization. In A. Scime (Ed.) *Web Mining: Applications and Techniques*. Idea Group Publishing.

Daniel, F., Matera, M., Morandi, A., Mortari, M., and Pozzi, G. (2007). Active Rules for Runtime Adaptivity Management. In Casteleyn et al. (2007).

Davies, J., Studer, R., and Warren, P. (Eds.) (2006). *Semantic Web Technologies — trends and research in ontology-based systems*. John Wiley and Sons.

Dayal, U., Buchmann, A. P., and McCarthy, D. R. (1988). Rules are Objects too: A Knowledge Model for an Active, Object-Oriented Databasesystem. In *Lecture notes in computer science on Advances in object-oriented database systems*, (pp. 129–143). New York, NY, USA: Springer-Verlag New York, Inc.

Demoen, B. (2005). Programming in Prolog. Using the ISO Standard. by William F. Clocksin, Christopher S. Mellish, Springer-Verlag, 2003, ISBN 3-540-00678-8, xiii+299 pages. *Theory Pract. Log. Program.*, *5*(3), 391–395.

Díaz, R. P. (2003). A Faceted Approach to Building Ontologies. In *Proceedings of the 2003 IEEE International Conference on Information Reuse and Integration, IRI – 2003, October 27–29, 2003, Las Vegas, NV, USA*, (pp. 458–465). IEEE Systems, Man, and Cybernetics Society.

Diestel, R. (2005). *Graph Theory*, (third ed.). vol. 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Heidelberg.

Dolog, P., Henze, N., Nejdl, W., and Sintek, M. (2004). The Personal Reader: Personalizing and Enriching Learning Resources using Semantic Web Technologies. In *International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2004)*.

Driver, M., Valdes, R., and Phifer, G. (2005). Rich Internet Applications are the Next Evolution of the Web. Tech. rep., Gartner.

Drucker, P. (1959). *Landmarks of Tomorrow: A Report on the New 'Post-Modern' World*. New York, NY, USA: Harper.

Duc Thanh Tran, A. A., Philipp Cimiano (2006). Rules for an Ontology-based Approach to Adaptation. In *1st International Workshop on Semantic Media Adaptation and Personalization*. Athen, Greece.

Duhl, J. (2003). Rich Internet Applications. Tech. rep., IDC.

ECMA International (1999). Standard ECMA-262, ECMAScript Language Specification, 3rd edition. Online resource, `http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf`. Created December 1999. Accessed on May 5, 2009.

Edwards, D. M., and Hardman, L. (1999). Lost in hyperspace: cognitive mapping and navigation in a hypertext environment. *Hypertext: theory into practice*, (pp. 90–105).

Etzion, O. (2008). On Events and Data. Online resource, `http://epthinking.blogspot.com/2008/07/on-events-and-data.html`. Created on July 5, 2008. Accessed on November 10, 2009.

European Commission (2007). Putting citizens first. Online resource, `http://ec.europa.eu/information_society/tl/soccul/egov/index_en.htm`. Updated September 2007. Accessed on March 27, 2009.

Feldkamp, D., Hinkelmann, K., Thönssen, B., and Thomas, S. M. (2006). D5: As-Is-Analysis. Public FIT deliverable, Online resource, `http://www.fit-project.org/D5.pdf`. Created September 2006. Accessed on March 31, 2009.

Fellbaum, C. (1998). *WordNet: An Electronic Lexical Database*. Bradford Books.

Findlater, L., and McGrenere, J. (2004). A Comparison of Static, Adaptive and Adaptable Menus. In E. Dykstra-Erickson, and M. Tscheligi (Eds.) *Proceedings of the 2004 Conference on Human Factors in Computing Systems, CHI 2004, Vienna, Austria, April 24 – 29, 2004*, (pp. 89–96). ACM.

Flügge, M., and Schmidt, K.-U. (2004). Using Semantic Web Services for ad hoc Collaboration in Virtual Teams. In R. Tolksdorf, and R. Eckstein (Eds.) *Berliner XML Tage 2004, 11.–13. Oktober 2004 in Berlin*, (pp. 187–198). XML-Clearinghouse.

Flügge, M., and Schmidt, K.-U. (2005). Verständnisvolle Dienste — Web Services mit der OWL. *iX — Magazin für professionelle Informationstechnik*, *5*, 136–143.

Forgy, C. L. (1979). *On the Efficient Implementation of Production Systems.*. Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA, USA.

Forgy, C. L. (1982). Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem. *Artificial Intelligence*, *19*, 17–37.

Frasincar, F., and Houben, G.-J. (2002). Hypermedia Presentation Adaptation on the Semantic Web. In P. D. Bra, P. Brusilovsky, and R. Conejo (Eds.) *AH*, vol. 2347 of *Lecture Notes in Computer Science*, (pp. 133–142). Springer.

Fukuda, R., and Bubb, H. (2003). Eye Tracking Study on Web-Use: Comparison Between Younger and Elderly Users in Case of Search Task with Electronic Timetable Service. *Psychnology Journal*, *1*(3), 202–228.

Galton, A., and Augusto, J. C. (2002). Two Approaches to Event Definition. In *DEXA '02: Proceedings of the 13th International Conference on Database and Expert Systems Applications*, (pp. 547–556). London, UK: Springer-Verlag.

Garrett, J. J. (2005). Ajax: A New Approach to Web Applications. Online resource, `http://www.adaptivepath.com/publications/essays/archives/000385.php`. Created on February 18, 2005. Accessed on March 5, 2009.

Garrigós, I., Cruz, C., and Gómez, J. (2007). A Prototype Tool for the Automatic Generation of Adaptive Websites. In Casteleyn et al. (2007).

Gatziu, S., and Dittrich, K. R. (1994). Detecting Composite Events in Active Database Systems Using Petrinets. In *Proc. Fourth International Workshop on Active Database Systems Research Issues in Data Engineering*, (pp. 2–9).

Gauch, S., Speretta, M., Chandramouli, A., and Micarelli, A. (2007). User Profiles for Personalized Information Access. In Brusilovsky et al. (2007), (pp. 54–89).

Gehani, N. H., Jagadish, H. V., and Shmueli, O. (1992a). Composite Event Specification in Active Databases: Model & Implementation. In *VLDB '92: Proceedings of the 18th International Conference on Very Large Data Bases*, (pp. 327–338). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Gehani, N. H., Jagadish, H. V., and Shmueli, O. (1992b). Event Specification in an Active Object-Oriented Database. *SIGMOD Rec.*, *21*(2), 81–90.

Gehani, N. H., Jagadish, H. V., and Shmueli, O. (1993). COMPOSE: A System for Composite Specification and Detection. In *Advanced Database Systems*, (pp. 3–15). London, UK: Springer-Verlag.

Geppert, A., Berndtsson, M., Lieuwen, D., and Roncancio, C. (1998). Performance Evaluation of Object-Oriented Active Database Systems Using the BEAST Benchmark. *Theor. Pract. Object Syst.*, *4*(3), 135–149.

Golder, S. A., and Huberman, B. A. (2006). Usage Patterns of Collaborative Tagging Systems. *J. Information Science*, *32*(2), 198–208.

Google Inc. (2007). Google Web History. Online resource, `http://www.google.com/psearch`. Created April 2007. Accessed on March 5, 2009.

Goy, A., Ardissono, L., and Petrone, G. (2007). Personalization in E-Commerce Applications. In Brusilovsky et al. (2007), (pp. 485–520).

Grau, B. C., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P., and Sattler, U. (2008). OWL 2: The Next Step for OWL. *Web Semant.*, *6*(4), 309–322.

Gröne, B. (2008a). How to Communicate Architecture — Technical Architecture Modeling at SAP (part 1). Online resource, `https://www.sdn.sap.com/irj/scn/weblogs?blog=/pub/wlg/8201`. Created January 2008. Accessed on June 24, 2009.

Gröne, B. (2008b). How to Communicate Architecture — Technical Architecture Modeling at SAP (part 2). Online resource, `https://www.sdn.sap.com/irj/scn/weblogs?blog=/pub/wlg/8684`. Created February 2008. Accessed on June 24, 2009.

Gruber, T. R. (1993). A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, *5*(2), 199–220.

Han, J., and Fu, Y. (1995). Discovery of Multiple-Level Association Rules from Large Databases. In *Proc. of 1995 Int'l Conf. on Very Large Data Bases (VLDB'95), Zürich, Switzerland, September 1995*, (pp. 420–431).

Han, J., and Fu, Y. (1999). Mining Multiple-Level Association Rules in Large Databases. *Knowledge and Data Engineering*, *11*(5), 798–804.

Han, J., Pei, J., and Yin, Y. (2000). Mining Frequent Patterns without Candidate Generation. In *SIGMOD '00: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, (pp. 1–12). New York, NY, USA: ACM.

Hanson, E. N., and Hasan, M. S. (1993). Gator: An Optimized Discrimination Network for Active Database Rule Condition Testing. Tech. rep., TR-93-036, CIS Department, University of Florida.

Heidasch, R. (July/August 2007). Get ready for the next generation of SAP business applications based on the Enterprise Service-Oriented Architecture (Enterprise SOA). *SAP Professional Journal*, *9*, 103–128.

Hevner, A. R., March, S. T., Park, J., and Ram, S. (2004). Design Science in Information Systems Research. *MIS Quarterly*, *28*(1).

Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., and Dean, M. (2004). SWRL: A Semantic Web Rule Language Combining OWL and RuleML. Tech. rep., W3C Member submission 21 may 2004.

Jain, R. (1991). *The Art of Computer Systems Performance Analysis: techniques for experimental design, measurement, simulation, and modeling*. New York: Wiley.

Jameson, A. (2007). Adaptive interfaces and agents. *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies, and Emerging Applications, Second Edition (Human Factors and Ergonomics)*.

Jensen, K. (1992). *Coloured Petri Nets: Basic Concepts, Analysis Methods, and Practical Use*. Springer.

Kaplan, C. A., Fenwick, J., and Chen, J. (1993). Adaptive Hypertext Navigation Based On User Goals and Context. *User Model. User-Adapt. Interact.*, *3*(3), 193–220.

Kenny, D., and Marshall, J. (2000). Contextual Marketing — The Real Business of the Internet. *Harvard Business Review*, *78*(6), 119–25.

Khare, R., and Çelik, T. (2006). Microformats: A Pragmatic Path to the Semantic Web. In *WWW '06: Proceedings of the 15th International Conference on World Wide Web*, (pp. 865–866). New York, NY, USA: ACM.

Kim, H. R., and Chan, P. K. (2003). Learning Implicit User Interest Hierarchy for Context in Personalization. In *Proceedings of the 2003 International Conference on Intelligent User Interfaces, January 12–15, 2003, Miami, FL, USA*, (pp. 101–108). ACM.

Kirsh, D. (2000). A Few Thoughts on Cognitive Overload. *Intellectica*, *1*(30), 19–51.

Kobsa, A. (2001). Tailoring Privacy to Users' Needs. In M. Bauer, P. J. Gmytrasiewicz, and J. Vassileva (Eds.) *User Modeling 2001, 8th International Conference, UM 2001, Sonthofen, Germany, July 13–17, 2001, Proceedings*, vol. 2109 of *Lecture Notes in Computer Science*, (pp. 303–313). Springer.

Kobsa, A. (2007). Privacy-Enhanced Web Personalization. In Brusilovsky et al. (2007), (pp. 628–670).

Koyani, S. J., Bailey, R. W., and Nall, J. R. (2006). *Research-Based Web Design & Usability Guidelines*. U.S. Department of Health and Human Services.

Krug, S. (2005). *Don't Make Me Think: A Common Sense Approach to the Web (2nd Edition)*. Thousand Oaks, CA, USA: New Riders Publishing.

Kunze, M., and Rösner, D. (2004). Issues in Exploiting GermaNet as a Resource in Real Applications. *LDV Forum*, *19*(1/2), 19–30.

Kushmerick, N., McKee, J., and Toolan, F. (2000). Towards Zero-Input Personalization: Referrer-Based Page Prediction. In Brusilovsky et al. (2000a), (pp. 133–143).

Le Hors, A., Le Hégaret, P., Wood, L., Nicol, G. T., Robie, J., Champion, M., and Byrne, S. (2004). Document Object Model (DOM) Level 3 Core Specification. World Wide Web Consortium, W3C Recommendation, REC-DOM-Level-3-Core-20040407.

Lieberman, H. (1995). Letizia: An Agent That Assists Web Browsing. In *IJCAI (1)*, (pp. 924–929).

Lieberman, H. (1997). Autonomous Interface Agents. In *CHI*, (pp. 67–74).

Likert, R. (1932). A Technique for the Measurement of Attitudes. *Archives of Psychology*, *22*(140), 1–55.

Liu, B. (2007). *Web Data Mining*. Data-Centric Systems and Applications. Springer Berlin Heidelberg.

Luckham, D. C. (2001). *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Luckham, D. C., and Schulte, R. (2008). Event Processing Glossary - Version 1.1. Online resource, `http://www.ep-ts.com/component/option,com_docman/task,doc_download/gid,66/Itemid,84`. Created July 2008. Accessed on March 5, 2009.

Machill, M., Neuberger, C., Schweiger, W., and Wirth, W. (2003). Wegweiser im Netz: Qualität und Nutzung von Suchmaschinen. In M. Machill, and C. Welp (Eds.) *Wegweiser im Netz. Qualität und Nutzung von Suchmaschinen*, (pp. 13–491). Gütersloh: Verlag Bertelsmann Stiftung.

Magoutas, B., Mentzas, G., Halaris, C., Konstantatos, M., Vladimirou, E., Legal, M., Schmidt, K.-U., Thomas, S. M., Rahmani, T., Feldkamp, D., Thoenssen, B., Hinkelmann, K., and Stoiljkovic, B. (2007). D16: Tools to Monitor the Quality of e-Government Services. Tech. rep., FIT Consortium, http://www.fit-project.org/D16.pdf. Accessed on March 5, 2009.

Magoutas, B., Schmidt, K.-U., Mentzas, G., and Stojanovic, L. (2010). Using Ontologies in an Adaptive Interface for Measuring User Perceived Portal Quality. *International Journal of Human-Computer Studies*. Accepted to be bublished in 2010.

Maloof, M., and Kochut, K. (1993). Modifying Rete to Reason Temporally. *Tools with Artificial Intelligence, 1993. TAI '93. Proceedings., Fifth International Conference on*, (pp. 472–473).

Marais, H., and Bharat, K. (1997). Supporting Cooperative and Personal Surfing with a Desktop Assistant. In *UIST '97: Proceedings of the 10th annual ACM symposium on User interface software and technology*, (pp. 129–138). New York, NY, USA: ACM.

Mardle, E. (2007). E-Government — A Citizen's Perspective. *Multilingual*, *18*(4), 41ff.

Mathe, N., and Chen, J. R. (1996). User-Centered Indexing for Adaptive Information Access. *User Model. User-Adapt. Interact.*, *6*(2-3), 225–261.

May, W., Alferes, J. J., and Amador, R. (2005). An Ontology- and Resources-Based Approach to Evolution and Reactivity in the Semantic Web. In R. Meersman, Z. Tari, M.-S. Hacid, J. Mylopoulos, B. Pernici, Ö. Babaoglu, H.-A. Jacobsen, J. P. Loyall, M. Kifer, and S. Spaccapietra (Eds.) *On the Move to*

Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE, OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2005, Agia Napa, Cyprus, October 31–November 4, 2005, Proceedings, Part II, vol. 3761 of *Lecture Notes in Computer Science*, (pp. 1553–1570). Springer.

Micarelli, A., and Gasparetti, F. (2007). Adaptive Focused Crawling. In Brusilovsky et al. (2007), (pp. 231–262).

Micarelli, A., Gasparetti, F., Sciarrone, F., and Gauch, S. (2007). Personalized Search on the World Wide Web. In Brusilovsky et al. (2007), (pp. 195–230).

Miniwatts Marketing Group (2008). Internet World Stats: Usage and Population Statistics. Online resource, `http://www.internetworldstats.com/stats.htm`. Updated on February 28, 2009. Accessed on March 3, 2009.

Mobasher, B. (2004). Web Usage Mining and Personalization. In M. P. Singh (Ed.) *Practical Handbook of Internet Computing*. Baton Rouge: Chapman Hall & CRC Press.

Mobasher, B., Cooley, R., and Srivastava, J. (2000). Automatic Personalization Based on Web Usage Mining. *Commun. ACM*, *43*(8), 142–151.

Morris, M. (1996). The Internet as Mass Medium. *J. Computer-Mediated Communication*, *1*(4).

Motik, B., Patel-Schneider, P. F., Parsia, B., Bock, C., Fokoue, A., Haase, P., Hoekstra, R., Horrocks, I., Ruttenberg, A., Sattler, U., and Smith, M. (2008). OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax. Candidate recommendation, W3C.

Nardi, D., and Brachman, R. J. (2003). An Introduction to Description Logics. In F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider (Eds.) *The Description Logic Handbook: Theory, Implementation, and Applications*, (pp. 1–40). Cambridge University Press.

Neale, M. (2008). Drools 5.0 M1 - New and Noteworthy. Online Article. `http://blog.athico.com/2008/07/drools-50-m1-new-and-noteworthy.html`.

Nielsen, J. (1989). Usability Engineering at a Discount. In *Proceedings of the third International Conference on Human-Computer Interaction on Designing and Using Human-Computer Interfaces and Knowledge Based Systems (2nd ed.)*, (pp. 394–401). New York, NY, USA: Elsevier Science Inc.

Nielsen, J. (1990). Big Paybacks from 'Discount' Usability Engineering. *IEEE Softw.*, *7*(3), 107–108.

Nielsen, J. (1993). *Usability Engineering*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Nielsen, J. (1994). Guerrilla HCI: Using Discount Usability Engineering to Penetrate the Intimidation Barrier. *Cost-justifying usability*, (pp. 245–272).

Nielsen, J. (1995). *Multimedia and Hypertext: The Internet and Beyond*. San Diego, CA, USA: Academic Press Professional, Inc.

Nielsen, J., and Loranger, H. (2006). *Prioritizing Web Usability*. Berkeley, Calif.: New Riders.

Norman, D. A., and Draper, S. W. (1986). *User Centered System Design; New Perspectives on Human-Computer Interaction*. Hillsdale, NJ, USA: L. Erlbaum Associates Inc.

Noy, N. F., and McGuiness, D. L. (2001). Ontology Development 101: A Guide to Creating Your First Ontology. Tech. rep., Stanford University.

Oppermann, R. (Ed.) (1994). *Adaptive User Support: Ergonomic Design of Manually and Automatically Adaptable Software*. Hillsdale, NJ, USA: L. Erlbaum Associates Inc.

Palvia, S. C. J., and Sharma, S. S. (2007). E-Government and E-Governance: Definitions/Domain Framework and Status around the World. In A. Agarwal, and V. Ramana (Eds.) *Foundations of E-Government, 5th International Conference on e-Governance, ICEG 2007, Hyderabad, India, December 28–30, 2007, Proceedings*.

Paschke, A., Kozlenkov, A., and Boley, H. (2007). A Homogenous Reaction Rules Language for Complex Event Processing. In *International Workshop on Event Drive Architecture for Complex Event Process*.

Peffers, K., Tuunanen, T., Rothenberger, M., and Chatterjee, S. (2008). A Design Science Research Methodology for Information Systems Research. *J. Manage. Inf. Syst.*, *24*(3), 45–77.

Pirolli, P., Pitkow, J. E., and Rao, R. (1996). Silk from a Sow's Ear: Extracting Usable Structures from the Web. In *CHI*, (pp. 118–125).

Pitkow, J., Schütze, H., Cass, T., Cooley, R., Turnbull, D., Edmonds, A., Adar, E., and Breuel, T. (2002). Personalized Search. *Commun. ACM*, *45*(9), 50–55.

Pixley, T. (2000). Document Object Model (DOM) Level 2 Events Specification. World Wide Web Consortium, Recommendation, REC-DOM-Level-2-Events-20001113.

Pospisil, J. (2007). Google Launches Web History, Privacy Fears Raised. Online resource, `http://tech.blorge.com/Structure:/2007/04/24/google-launches-web-history-privacy-fears-raised`. Created on April 24, 2007. Accessed on March 5, 2009.

Proctor, M. (2007). Relational Declarative Programming with JBoss Drools. In *SYNASC '07: Proceedings of the Ninth International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, (p. 5). Washington, DC, USA: IEEE Computer Society.

Rahmani, T., Thomas, S. M., Schmidt, K.-U., and Stojanovic, L. (2008). Using Semantic Web Usage Mining To Improve E-Government Websites. In E. Ferro, H. J. Scholl, and M. A. Wimmer (Eds.) *Electronic Government: Proceedings of ongoing research and projects of EGOV 08. 7th International Conference, EGOV 2008.*, vol. 27 of *Schriftenreihe Informatik*. Trauner Druck, Linz.

Rambøll Management (2004). Top of the Web: User Satisfaction and Usage Survey of eGovernment Service. Online resource, `www.eprocnet.gov.ie/other-documents/topofthewebsurveyresults.pdf`. Created September 2004. Accessed on March 9, 2009.

Romero, C., Ventura, S., Martinez, C. H., and Bra, P. D. (2005). Extending AHA! In *Proceedings of the Fifth International Conference on Human System Learning, ICHSL*. Europia.

Rosati, L., Lai, M. E., and Gnoli, C. (2004). Faceted Classification for Public Administration. In *Semantic Web Applications and Perspectives (SWAP) - 1st Italian Semantic Web Workshop*. Ancona, Italy.

RSS Advisory Board (2009). RSS 2.0 Specification Version 2.0.11. Online resource, `http://www.rssboard.org/rss-specification`. Created on March 30, 2009. Accessed on July 08, 2009.

Russell, A. (2006). Comet: Low Latency Data for the Browser. Online resource, `http://alex.dojotoolkit.org/?p=545`. Created March 2006. Accessed on April 12, 2009.

Scerri, S., Sintek, M., van Elst, L., and Handschuh, S. (2007). NEPOMUK Annotation Ontology Specification. Tech. rep., NEPOMUK Consortium.

Schmidt, A., Beigl, M., and Gellersen, H.-W. (1999). There is more to Context than Location. *Computers & Graphics*, *23*(6), 893–901.

Schmidt, K.-U. (2004). *Ad-hoc-Interoperabilität heterogener Personal Information Manager durch semantische Web Services*. Master's thesis, Technische Universität Berlin.

Schmidt, K.-U. (2007). Enterprise SOA and Beyond. Key note. E-Business-Infoday, SOA und Web Services, Berlin, Germany.

Schmidt, K.-U., Anicic, D., and Stühmer, R. (2008a). Event-driven Reactivity: A Survey and Requirements Analysis. In M. Hepp, N. Stojanovic, K. Hinkelmann, D. Karagiannis, and R. Klein (Eds.) *Proceedings of the 3rd International Workshop on Semantic Business Process Management. Tenerife, Spain, June 2nd, 2008*, vol. 472 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Schmidt, K.-U., Dörflinger, J., Rahmani, T., Sahbi, M., Stojanovic, L., and Thomas, S. M. (2008b). An User Interface Adaptation Architecture for Rich Internet Applications. In S. Bechhofer, M. Hauswirth, J. Hoffmann, and M. Koubarakis (Eds.) *The Semantic Web: Research and Applications, 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, June 1–5, 2008, Proceedings*, vol. 5021 of *Lecture Notes in Computer Science*, (pp. 736–750). Springer.

Schmidt, K.-U., Oberle, D., and Deissner, K. (2008c). Taking Enterprise Search to the Next Level. In C. Bizer, and A. Joshi (Eds.) *Proceedings of the Poster and Demonstration Session at the 7th International Semantic Web Conference (ISWC2008), Karlsruhe, Germany, October 28, 2008*, vol. 401 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Schmidt, K.-U., Sarnow, T., and Stojanovic, L. (2009a). Socially Filtered Web Search: An Approach Using Social Bookmarking Tags To Personalize Web Search. In S. Y. Shin, and S. Ossowski (Eds.) *Proceedings of the 2009 ACM Symposium on Applied Computing (SAC), Honolulu, Hawaii, USA, March 9–12, 2009*, (pp. 670–674). ACM.

Schmidt, K.-U., and Stojanovic, L. (2008). From Business Rules to Application Rules in Rich Internet Applications. In W. Abramowicz, and D. Fensel (Eds.) *Business Information Systems, 11th International Conference, BIS 2008, Innsbruck, Austria, May 5–7, 2008. Proceedings*, vol. 7 of *Lecture Notes in Business Information Processing*, (pp. 447–458). Springer.

Schmidt, K.-U., Stojanovic, L., Stojanovic, N., and Thomas, S. (2007). On Enriching Ajax with Semantics: The Web Personalization Use Case. In E. Franconi, M. Kifer, and W. May (Eds.) *The Semantic Web: Research and Applications, 4th European Semantic Web Conference, ESWC 2007, Innsbruck, Austria, June 3–7, 2007, Proceedings*, vol. 4519 of *Lecture Notes in Computer Science*, (pp. 686–700). Springer.

Schmidt, K.-U., Stojanovic, L., Stojanovic, N., and Thomas, S. M. (2010a). Personalization in E-Government: An Approach that Combines Semantics and

Web 2.0. In V. Peristeras, T. Vitvar, and K. Tarabanis (Eds.) *Semantic Technologies for E-Government: A European Perspective*. Springer.

Schmidt, K.-U., Stühmer, R., Dörflinger, J., Rahmani, T., Thomas, S. M., and Stojanovic, L. (2010b). Adaptive Reactive Rich Internet Applications. *Annals of Information Systems: Web 2.0 & Semantic Web*, *6*, 79–104.

Schmidt, K.-U., Stühmer, R., and Stojanovic, L. (2008d). Blending Complex Event Processing with the Rete Algorithm. In D. Anicic, C. Brelage, O. Etzion, and N. Stojanovic (Eds.) *iCEP2008: 1st International Workshop on Complex Event Processing for the Future Internet, collocated with the Future Internet Symposium (FIS2008)*, vol. Vol-412. CEUR Workshop Proceedings (CEUR-WS.org, ISSN 1613-0073).

Schmidt, K.-U., Stühmer, R., and Stojanovic, L. (2008e). From Business Rules to Application Rules in Rich Internet Applications. *Scalable Computing: Practice and Experience, Scientific International Journal for Parallel and Distributed Computing, Special Issue: The Web on the Move*, *9*(4), 329–340.

Schmidt, K.-U., Stühmer, R., and Stojanovic, L. (2009b). Gaining Reactivity for Rich Internet Applications by Introducing Client-side Complex Event Processing and Declarative Rules. In N. Stojanovic, and O. Etzion (Eds.) *The 2009 AAAI Spring Symposium on Intelligent Event Processing*. Association for the Advancement of Artificial Intelligence.

Schmidt, K.-U., and Thomas, S. (2007). D10: Methods and Tools for Capturing Users' Interaction with a Front-Office on the Semantic Level. Tech. rep., FIT Consortium, `http://www.fit-project.org/D10.pdf`.

Schneider-Hufschmidt, M., Kühme, T., and Malinowski, U. (Eds.) (1993). *Adaptive User Interfaces: Principles and Practice*. Amsterdam: North-Holland.

Schonfeld, E. (2008). Proximic Signs Deals With Yahoo and eBay to Turn Product Listings into Contextual Ads; Taking on AdSense. Online resource, `http://www-origin.proximic.com/info/press/?p=53`. Created on January 15, 2008. Accessed on October 10, 2009.

Schreiber, A. T., Dubbeldam, B., Wielemaker, J., and Wielinga, B. J. (2001). Ontology-Based Photo Annotation. *IEEE Intelligent Systems*, *16*(3), 66–74.

Segaran, T. (2007). *Programming collective intelligence*. O'Reilly.

Shen, X., Tan, B., and Zhai, C. (2005). Implicit User Modeling for Personalized Search. In O. Herzog, H.-J. Schek, N. Fuhr, A. Chowdhury, and W. Teiken (Eds.) *Proceedings of the 2005 ACM CIKM International Conference on Information and Knowledge Management, Bremen, Germany, October 31 – November 5, 2005*, (pp. 824–831). ACM.

Spiliopoulou, M., Mobasher, B., Berendt, B., and Nakagawa, M. (2003). A Framework for the Evaluation of Session Reconstruction Heuristics in Web Usage Analysis. *INFORMS Journal of Computing, Special Issue on Mining Web-Based Data for E-Business Applications*, *15*.

Srivastava, J., Cooley, R., Deshpande, M., and Tan, P.-N. (2000). Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data. *SIGKDD Explorations*, *1*(2), 12–23.

Staab, S., and Studer, R. (Eds.) (2004). *Handbook on Ontologies*. International Handbooks on Information Systems. Springer.

Stojanovic, L., Ma, J., and Stojanovic, N. (2007a). D9: Methods and Tools for Semi-Automatic Learning of a Domain Ontology that Models the Content of a Front Office. Tech. rep., FIT Consortium, `http://www.fit-project.org/D9_Resubmission.pdf`.

Stojanovic, L., Ma, J., Yu, J., Stojanovic, N., Schmidt, K.-U., Thomas, S., and Rahmani, T. (2007b). D18: Methods and Tools for Mining the Log Data by Taking into Account Background Knowledge. Tech. rep., FIT Consortium, `http://www.fit-project.org/D18.pdf`.

Stojanovic, L., Schmidt, K.-U., Stojanovic, N., and Thomas, S. M. (2007c). Adaptive Portals Based on Combining Semantics and Ajax. In M. Cunningham, and P. Cunningham (Eds.) *Expanding the Knowledge Economy: Issues, Applications, Case Studies. Proceedings of E-Challenges 2007*, (pp. 230–237). Amsterdam: IOS Press.

Stojanovic, L., Stojanovic, N., and Ma, J. (2007d). On the Conceptual Tagging: An Ontology Pruning Use Case. In *2007 IEEE / WIC / ACM International Conference on Web Intelligence, WI 2007, 2–5 November 2007, Silicon Valley, CA, USA, Main Conference Proceedings*, (pp. 344–350). IEEE Computer Society.

Stojanovic, L., Stojanovic, N., and Ma, J. (2007e). Web Information Systems Engineering - WISE 2007, 8th International Conference on Web Information Systems Engineering, Nancy, France, December 3-7, 2007, Proceedings. In Benatallah et al. (2007), (pp. 249–260).

Stojanovic, N., Stojanovic, L., and Ma, J. (2008). On the Conceptual Tag Refinement. In R. L. Wainwright, and H. Haddad (Eds.) *Proceedings of the 2008 ACM Symposium on Applied Computing (SAC), Fortaleza, Ceara, Brazil, March 16–20, 2008*, (pp. 2331–2335). ACM.

Studer, R., Benjamins, V. R., and Fensel, D. (1998). Knowledge Engineering: Principles and Methods. *Data Knowl. Eng.*, *25*(1–2), 161–197.

Stühmer, R., Anicic, D., Sen, S., Ma, J., Schmidt, K.-U., and Stojanovic, N. (2009a). Client-Side Event Processing for Personalized Web Advertisement. In R. Meersman, T. S. Dillon, and P. Herrero (Eds.) *On the Move to Meaningful Internet Systems: OTM 2009, Confederated International Conferences, CoopIS, DOA, IS, and ODBASE 2009, Vilamoura, Portugal, November 1–6, 2009, Proceedings, Part II*, vol. 5871 of *Lecture Notes in Computer Science*, (pp. 1069–1086). Springer.

Stühmer, R., Anicic, D., Sen, S., Ma, J., Schmidt, K.-U., and Stojanovic, N. (2009b). Lifting Events in RDF from Interactions with Annotated Web Pages. In A. Bernstein, D. R. Karger, T. Heath, L. Feigenbaum, D. Maynard, E. Motta, and K. Thirunarayan (Eds.) *The Semantic Web - ISWC 2009, 8th International Semantic Web Conference, ISWC 2009, Chantilly, VA, USA, October 25–29, 2009. Proceedings*, vol. 5823 of *Lecture Notes in Computer Science*, (pp. 893–908). Springer.

Stumme, G., Hotho, A., and Berendt, B. (2006). Semantic Web Mining: State of the Art and Future Directions. *Web Semantics: Science, Services and Agents on the World Wide Web*, *4*(2), 124–143.

Sun, J.-T., Zeng, H.-J., Liu, H., Lu, Y., and Chen, Z. (2005). CubeSVD: A Novel Approach to Personalized Web Search. In A. Ellis, and T. Hagino (Eds.) *Proceedings of the 14th International Conference on World Wide Web, WWW 2005, Chiba, Japan, May 10–14, 2005*, (pp. 382–390). ACM.

Teevan, J., Dumais, S. T., and Horvitz, E. (2005). Personalizing Search via Automated Analysis of Interests and Activities. In R. A. Baeza-Yates, N. Ziviani, G. Marchionini, A. Moffat, and J. Tait (Eds.) *SIGIR 2005: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Salvador, Brazil, August 15–19, 2005*, (pp. 449–456). ACM.

Terveen, L., and McDonald, D. W. (2005). Social Matching: A Framework and Research Agenda. *ACM Trans. Comput.—Hum. Interact.*, *12*(3), 401–434.

Theng, Y. L., Thimbleby, H., and Jones, M. (1996). 'Lost in Hyperspace': Psychological Problem or bad Design? In *Proceedings of the Asia-Pacific Computer-Human Interaction, APCHI'96*, (pp. 387–396). Singapore.

Thomas, C. G. (1995). BASAR: A Framework for Integrating Agents in the World Wide Web. *IEEE Computer*, *28*(5), 84–86.

Tidwell, J. (2006). *Designing Interfaces*, (1. ed. ed.). O'Reilly.

Tsandilas, T., and Schraefel, M. C. (2004). Usable Adaptive Hypermedia Systems. *Hypermedia*, *10*(1), 5–29.

United Nations Population Division (2006). World Population Prospects: The 2006 Revision Population Database. Online resource, `http://esa.un.org/unpp`. Updated on September 20, 2007. Accessed on February 3, 2009.

van Kesteren, A. (2008). XMLHttpRequest Level 2. World Wide Web Consortium, Working Draft, WD-XMLHttpRequest2-20080930.

Vinson, N. G. (1999). Design Guidelines for Landmarks to Support Navigation in Virtual Environments. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, (pp. 278–285). New York, NY, USA: ACM.

Völkel, M., Krötzsch, M., Vrandecic, D., Haller, H., and Studer, R. (2006). Semantic Wikipedia. In *WWW '06: Proceedings of the 15th International Conference on World Wide Web*, (pp. 585–594). New York, NY, USA: ACM.

Volokh, E. (2000). Personalization and Privacy. *Commun. ACM*, *43*(8), 84–88.

Walzer, K., Breddin, T., and Groch, M. (2008). Relative Temporal Constraints in the Rete Algorithm for Complex Event Detection. In *DEBS '08: Proceedings of the second International Conference on Distributed Event-based Systems*, (pp. 147–155). New York, NY, USA: ACM.

Walzer, K., Schill, A., and Löser, A. (2007). Temporal Constraints for Rule-Based Event Processing. In A. S. Varde, and J. Pei (Eds.) *Proceedings of the First Ph.D. Workshop in CIKM, PIKM 2007, Sixteenth ACM Conference on Information and Knowledge Management, CIKM 2007, Lisbon, Portugal, November 9, 2007*, (pp. 93–100). ACM.

Waterman, D. A., and Hayes-Roth, F. (1978). *Pattern-Directed Inference Systems*. Orlando, FL, USA: Academic Press, Inc.

Waterworth, J. A. (1994). A Pattern of Islands: Exploring Public Information Space in a Private Vehicle. In P. Brutsilosky, P. Kommers, and N. A. Streitz (Eds.) *Multimedia, Hypermedia, and Virtual Reality: Models, Systems, and Applications, First International Conference, MHVR '94, Moscow, Russia, September 14–16, 1994, Selected Papers*, vol. 1077 of *Lecture Notes in Computer Science*, (pp. 265–278). Springer.

Webber, A. E., Leganza, G., Boehm, E., and Holmes, B. J. (2005). Best Practices: Making eGovernment Web Sites Usable. Online resource, `http://www.forrester.com/Research/Document/Excerpt/0,7211,37805,00.html`. Created on September 28, 2005. Accessed on May 19, 2006.

Wiesner, H., and Pacnik, H. (2006). E-Government Style Guide for E-Forms 2.0. Online resource, `http://www.ref.gv.at/uploads/media/sg-stg_2-0-0_ENGL.pdf`. Created January 2006. Accessed on March 27, 2009.

Witten, I. H., and Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques*, (second ed.). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Yang, Y., Aufaure, M.-A., and Claramunt, C. (2007). Towards a DL-Based Semantic User Model for Web Personalization. In *ICAS '07: Proceedings of the Third International Conference on Autonomic and Autonomous Systems*, (p. 61). Washington, DC, USA: IEEE Computer Society.

Zou, K., Fielding, J., Silverman, S., and Tempany, C. (2003). Hypothesis Testing I: Proportions. *Radiology*.