An User Interface Adaptation Architecture for Rich Internet Applications

Kay-Uwe Schmidt[1], Jörg Dörflinger[1], Tirdad Rahmani[1], Mehdi Sahbi[1], Ljiljana Stojanovic[2], and Susan Marie Thomas[1]

[1] SAP AG, Research, Vincenz-Prienitz-Straße 1, 76131 Karlsruhe
`~http://www.sap.com`
[2] FZI Forschungszentrum Informatik, Haid-und-Neu-Straße 10-14, 76131 Karlsruhe
`~http://www.fzi.de`

Abstract. The need for adaptive and personalized Rich Internet Application puts a new dimension to already existing approaches of Adaptive Hypermedia Systems. Instead of computing the adaptation steps at the server, Rich Internet Applications need a client-side approach that can react immediately on user input. In this paper we present a novel approach that holistically combines page annotations, semantic Web usage mining, user modeling, ontologies and rules to adapt AJAX pages. The focus of our pater is the conceptual introduction of the autonomous client. An autonomous client directly executes all necessary adaptation steps based on a user model, without requesting any logic on the server. In order to realize this, we use ontologies to annotate Rich Internet Applications and to describe the user model as well as semantic Web usage mining for detecting adaptation rules. Additionally, we provide a detailed overview and evaluation of how we moved resource-intensive ontology processing and rules execution from the server to the client.

1 Introduction

In Adaptive Hypermedia Systems (AHSs) adaptation strategies have been intensively studied [1] and are well understood for conventional Web applications adhering to the Web page paradigm[1]. With conventional techniques, the tracking of user clicks, the user modeling, as well as the adaptation take place on the server. This limits the possibilities of user tracking to the user requests seen by the server [2], which is actually a subset of the user clicks. Furthermore adaptation can only take place when a user requests a new page, which then is adapted to his/her needs. On-the-fly adaptation, without reloading the whole page, is not obtainable. Recently, with the rise of AJAX [3], new possibilities appeared for user tracking and user interface adaptation. With AJAX the look and feel of Web pages are transformed to that of desktop applications and users are accustomed to highly responsive user interfaces. State of the art Web applications

[1] The Web paradigm determines that very Web page in a series of pages is downloaded separately.

obtain a responsive user interface by encoding the adaptation logic in static script languages like JavaScript.

In this paper we present a novel solution for on-the-fly adaptation of Rich Internet Applications (RIAs) applications. We introduce a holistic framework covering the whole adaptation cycle, from obtaining rules, over ad-hoc user modeling, to on-the-fly user interface adaptation. Compared to common AHSs our approach goes two steps beyond, as we introduce not only ontologies, for capturing and storing the user model, and declarative logical rules, for carrying out the adaptation, but also advance the state of the art through the client-side realization of user modeling and portal adaptation. We consider the evaluation and execution of the adaptation rules on the client-side as the major contribution of this paper, as it is the prerequisite to responsive user interfaces for RIAs. Client-side rule processing has several advantages, such as reduction of client-server communication to a minimum, and in-time response to user interactions. The acquisition of adaptation rules is an indispensable pre-requisite to adaptation. Adaptation rules declaratively encode adaptation logic based on the user's behavior, which means based on the recorded user interactions with the RIAs. Adaptation rules can be gained by mining user access log data. With the help of annotations, added to the Web application in advance, we present a semantic approach of Web usage mining in order to find common Web usage patterns. In turn, the most useful patterns can be directly modeled as adaptation rules, guiding the user while interacting with the RIA. The user interactions are stored at run-time in a user model residing together with the application rules on the client-side. We do not assume that the system has any previous information about a user like explicitly specified user preferences or user roles determined by log-in information. In each user session the user model has to be acquired from scratch. The user model, the RIA and the rules are modeled in ontologies that are transformed at design-time into executable AJAX snippets. The adaptation ontologies as a backbone of our adaptive solution were already comprehensively described in [4] and are not the focus of this paper.

The rest of the paper is structured as follows. Section 2 an example is presented in order to motivate our work. In Section 3 we give an overview of the logical system architecture and illustrate the adaptation loop. The following two Sections 4 and 5 go into the details of the design-time and run-time architecture accordingly. An evaluation of our approach is given in Section 6, and in Section 7 we discuss related work. In Section 8 acknowledgments are given, and, finally, the paper closes with conclusions and prospects for future work.

2 Motivating Example

Searching for the right form is an widespread problem in portals especially in e-Government portals. The average user of an e-Government portal is usually not an expert but rather a novice regarding the use of online forms. E-Government Web applications are designed for end users without special training. Two major requirements for e-Government Web applications are: Citizen-centric services and ease of use [5]. To meet these requirements a form of non-intrusive user guidance can be provided.

In our first motivating example we pick up the idea of user guidance. We want to recommend links related to the forms the user already filled in. Lets consider the following use case: Building application. The citizen officially has to apply for building permission at the local department of housing and urban development. The building application can be filled in online and consists of several forms like the main building application form, building license form, building description form, start of construction form, to mention just a few. We assume no predefined workflow determining the number and order of forms the citizen has to fill in. After filling in the main form the user wants to know which form to fill in next. This can be accomplished by suggesting related forms based on the forms filled in by the current user compared to the collaboratively filtered Web usage behavior of past users.

3 Logical System Architecture: The Adaptation Loop

Our user interface adaptation architecture for RIAs, as depicted in Figure 1, is a two-stage approach consisting of three cycles forming the adaptation loop. The design-time stage and the run-time stage logically divide the components of our architecture into off-line and online components respectively. That is, the stages refer to the invocation time of the components comprising our architecture. The three cycles, the modeling cycle on the left, the adaptation cycle on the right and the larger transfer cycle in the middle illustrate the self-adaptive character of our architecture.

The modeling cycle stands for the design time components in charge of constructing the adaptation rules. At design time, an indispensable prerequisite for our RIA adaptation approach, the portal must be annotated by using a portal annotation tool. After annotating the structure and content of the RIA, user access log data, collected in the past, can be mined for useful Web usage patterns. This is done by the semantic Web usage mining component. Once useful patterns are found, they can be formulated as adaptation rules by using a rule design tool. The adaptation rules are stored in an ontology format. After designing the adaptation rules on a conceptual level, based on the annotated RIA, they are translated at design-time by the rule transformer into a client-readable format like JavaScript.

The rightmost cycle in Figure 1, the adaptation cycle, is executed, like the transfer cycle in the middle of the figure, at run-time. The aim of the adaptation cycle is to adapt the RIA based on the predefined adaptation rules and the current user model. The adaptation rules are obtained from the modeling cycle via the transfer adaptation rules component of the transfer cycle. The user model is built up by the user tracking component, which records the user's interactions with the RIA. Based on the user model, which is constructed on-the-fly, the adaptation rules are evaluated and, if the condition part holds, are fired. It is the task of the rule evaluation component and the rule execution component to

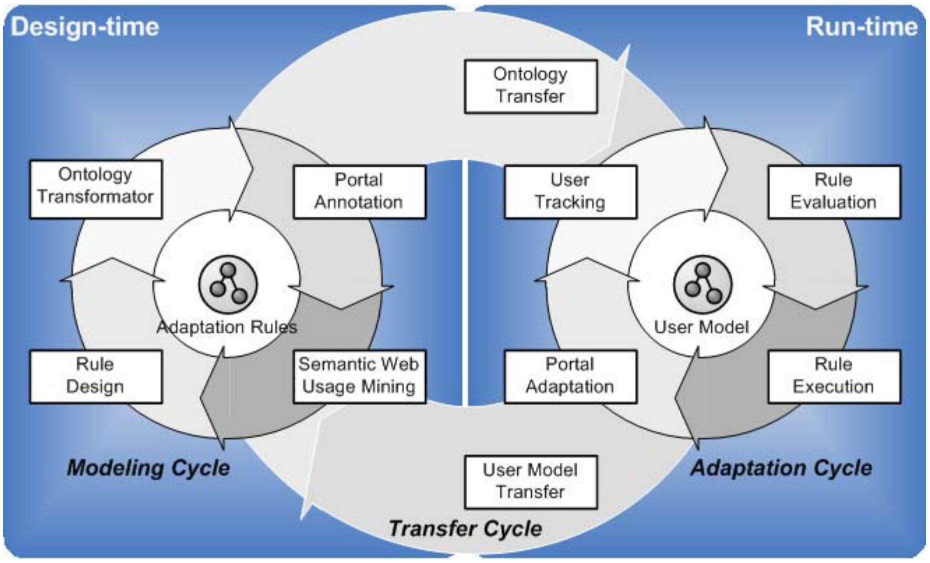


Fig. 1. Logical System Architecture: The Adaptation Loop

carry out rule processing. If a rule fires the corresponding actions are executed and the RIA adapts itself directly on the client-side without server requests. At the end of the session the tracked user model is sent back to the server. On the server-side all user models are collected and fed back into the modeling cycle in order to mine new behavioral patterns. Thus, the adaptation loops starts again.

4 Design-Time Architecture

The design-time architecture consists of the components constituting the modeling cycle as depicted in Figure 1. These tools and components are executed off-line during the annotation, mining, design and transformation phases.

4.1 Ontology Creation and Portal Annotation

Suitable ontologies are crucial for our RIA adaptation approach. We developed an approach amalgamating ontology learning [6], ontology refinement [7] and annotating RIAs into one coherent tool. The main ideas behind this approach are described in detail in [8]. Based on this approach we developed a domain-specific e-Government ontology, as well as an annotation knowledge-base linking concepts of the domain ontology to contents of the RIA. Additionally, a RIA ontology, describing the structural aspects, and a user model ontology were developed using standard ontology development tools like Protege[2]. An overview

[2] http://protege.stanford.edu/

and description of the developed ontologies are given in [4]. As a proof of concept we annotated our internal demo portal and the e-Government portal of the city of Vöcklabruck[3] with our ontolgies under supervision of e-Government experts. All ontologies are described using the Web Ontology Language (OWL) [9].

4.2 Semantic Web Usage Mining

Web Usage Mining is the application of data mining algorithms on Web server access logs to gain a better understanding of user behavior. Besides the access logs, metadata describing the Web resources and their content are conceptually helpful for data-mining analysis. The utilization of the metadata is strongly dependent on its organization and the way it can be combined with the log entries [10]. In recent years the research areas semantic Web and Web mining have become more important and are merged together into a new research field called semantic Web mining which has been deeply analyzed [11,12]. In particular, semantic Web Usage Mining as a subcategory of semantic Web mining enables tracking of user behavior at a conceptual level.

Figure 2 shows the different stages of our semantic Web usage mining architecture. The first stage is the preprocessing stage. Here the ontologies are designed, the Web resources are annotated and the user sessions are reconstructed. Reconstructing user sessions is a difficult task, because of the lack of log-in data. The user sessions are reconstructed using common reconstruction methods as detailed in [13]. The available data sources for the second stage, the data mining step, comprise the reconstructed sessions, the annotated Web resources and the domain ontology. The last two items form a knowledge-base of available

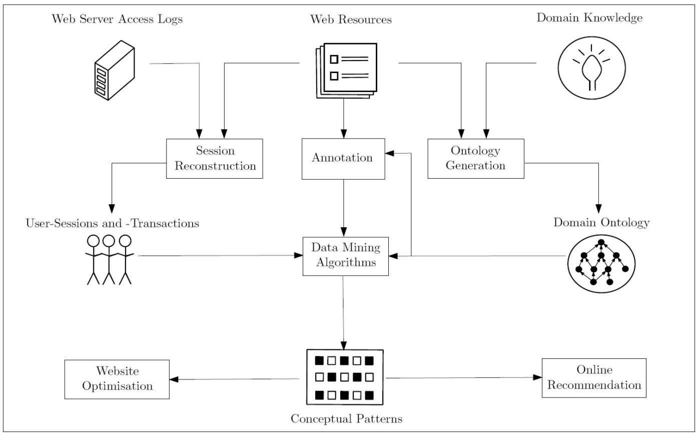


Fig. 2. Semantic Web usage mining architecture

[3] http://www.voecklabruck.at/

metadata. The Web usage mining algorithms used for this purpose are association rules, sequential rules, and multi level rules based on a concept hierarchy. Moreover, clustering approaches which consider the user behavior as well as semantic contents are considered. More technical details are given in [14].

The results of the data mining step can be used for website optimization or online recommendation based on the user behavior and semantic content. The new item problem nicely illustrates the benefits of using semantics in Web usage mining. New items can be recommended directly after their annotation. That is possible, because our data mining approach, as well as our recommendation engine, work on concepts rather than concrete URLs or IDs. As an example the following rule is considered: $C1 \wedge C2 \rightarrow C3$. This rule states that if pages annotated with $C1$ and $C2$ are visited, all pages annotated with concept $C3$ are candidates to be recommended. If now a new Web resource annotated with $C3$ is introduced, it can be added to these candidates, because the rules are on the conceptual level.

4.3 Design of Adaptation Rules

After applying semantic Web usage mining to access-log files, the discovered patterns need to be analyzed by an e-Government expert. The domain expert has to judge, whether the patterns are useful or not. Patterns, which have been judged useful, are then encoded into a rule language as adaptation rules by an ontology engineer with the help of customary ontology and rule editors. So, for instance, we discovered the following rule after evaluating the patterns found in the annotated access log file of the city of Vöcklabruck: 85% of all users that filled in the marriage certificate form and the wedding day form also filled in the birth certificate form[4].

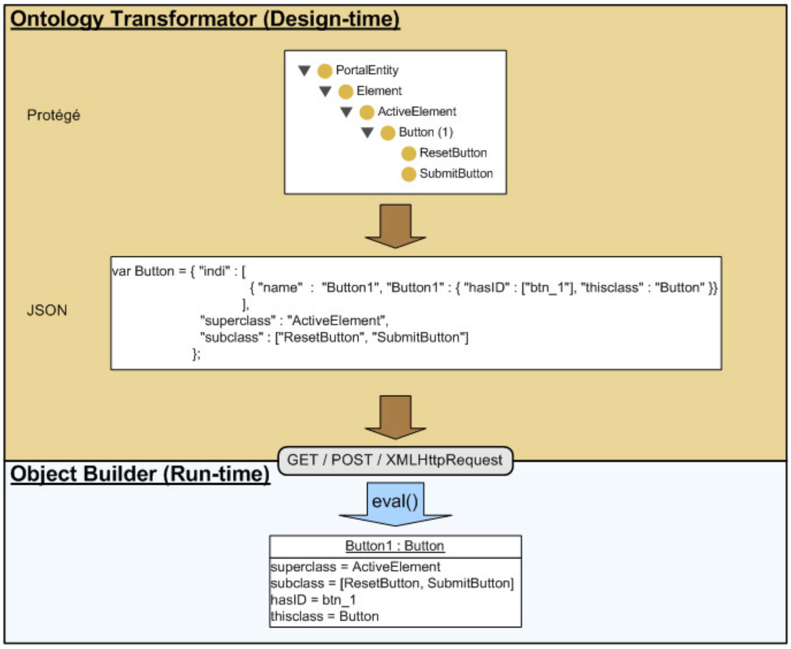
Example 1 (Adaptation rule in SWRL)
portal:Form(?a) $\wedge$ portal:isVisited(?a, true) $\wedge$
domain:WeddingDay(?b) $\wedge$ portal:isAnnotated(?a, ?b) $\wedge$
portal:Form((?c)) $\wedge$ portal:isVisited(?c, true) $\wedge$
domain:MarriageCertificate(?d) $\wedge$ portal:isAnnotated(?c, ?d) $\wedge$
domain:BirthCertificate(?e) $\rightarrow$ portal:showLink(?e)

We are using the Semantic Web Rule Language (SWRL) [15] because it nicely fits to our OWL ontologies. Example 1 shows how an ontology and rule engineer could formulate the rule described above in SWRL. Translated to English the rule states, that whenever a form annotated with WeddingDay and a form annotated with MarriageCertificate were visited show all links to forms annotated with BirthCertificate as link recommendations. WeddingDay, DateOfWedding and BirthCertificate are concepts taken from the domain ontology. The functionality of displaying recommended links is realized as a SWRL built-in. The built-in finds all forms annotated with BirthCertificate, reads the link from the appropriate property and, finally, recommends these links.

[4] Support: 0,01; confidence: 0,85.

4.4 Ontology Transformer

Having the ontologies, annotations and adaptation rules in place, the last step in the modeling cycle is still the transformation of all of these parts into a client-readable format that can be executed by a browser's JavaScript engine. As an Internet browser on a client machine has only limited processing power and main memory capacity, both ontologies and rules must be translated beforehand in an easy-to-parse format that can be effortlessly executed on the client-side. Due to the lack of a client-side reasoner we materialize all ontologies at the server-side. By using an OWL reasoner we check the consistency of the ontology at design-time, classify the instances and infer the class hierarchy. There are two possibilities to represent ontologies on the client-side: XML or JSON (JavaScript Object Notation) [16]. XML is very verbose and adds additional overhead to the payload. Furthermore, XML rules encoded in XML cannot be executed directly on the client, but have to be parsed, an added expense. Therefore, we decided to represent ontologies, annotations and rules in the compact and directly executable data interchange format JSON.


Ontology Transformator (Design-time)
Protégé
PortalEntity
Element
ActiveElement
Button (1)
ResetButton
SubmitButton
JSON
GET / POST / XMLHttpRequest
Object Builder (Run-time)
eval()

```
var Button = { "indi" : [
                        { "name"  :  "Button1", "Button1" : { "hasID" : ["btn_1"], "thisclass" : "Button" }}
                        ],
            "superclass" : "ActiveElement",
            "subclass" : ["ResetButton", "SubmitButton"]
           };
```

```
Button1 : Button
superclass = ActiveElement
subclass = [ResetButton, SubmitButton]
hasID = btn_1
thisclass = Button
```

Fig. 3. Transformation of ontologies

Figure 3 shows what the JSON format looks like, after transforming the concept Button. As depicted in the concept hierarchy of Protege the class Button has several super and sub concepts. Additionally, there exist one instance of

Button in the ontology. Not shown are the properties of the Button class, which in fact are: *hasID* and *thisclass*. A class is represented as object in JSON and its instances are collected in a property of type Array called *indi*. This array contains all instances as objects whereas the objects in turn hold their properties as attributes. The last attribute *thisclass* is a reference to the instantiated class and is automatically added during the translation. The class hierarchy is stored directly in the JSON object representing the class as Array attributes: superclass and subclass. SWRL rules are also encoded as objects consisting of a condition and action part. SWRL build-ins are encoded manually as JavaScript functions beforehand. A call to these external functions is placed into the JSON translation whenever the transformer detects a built-in in the original adaptation rules format. At run-time the object builder generates real objects from the JSON string, as depicted in the lower part on Figure 3.

5 Run-Time Architecture

The core responsibility of the run-time architecture is to ensure the user-centric adaptiveness of the RIA. The run-time architecture is constituted by the adaptation and transfer cycle as depicted in Figure 1. After transforming the ontologies, annotations and adaptation rules into a client-readable format at design-time, they can be transmitted as JavaScript code in answer to a client request at any point in time. When a user requests the RIA, not only content and layout data are send to the client, but also the JSON representation of the ontlogies, annotations and rules. On the client-side the user model is built up and the portal is adapted by tracking user interactions and executing adaptation rules. Figure 4 shows the interplay of the constituent run-time components.

In a Web browser HTML pages are internally represented as a DOM (Document Object Model). Whenever a user interacts with the Web page the DOM fires appropriate events which can be caught by the event handler component. In order

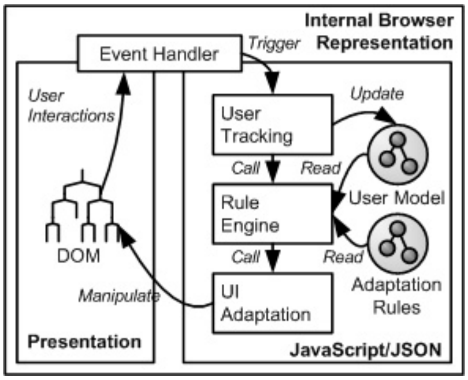


Fig. 4. Logical client-side run-time architecture

to catch events the event handler has to register first to specific event types. In our current implementation this is done manually. The Web programmer has to explicitly specify which kinds of events shall be tracked. Each recognized event results in a call of the user tracking component and, in a second step, the invocation of the rule engine. The user tracking component resolves the relationships between the JavaScript events, the user interface elements and their annotations. Furthermore it records the events to the user model. In this way the user model materializes the browsing history of the current user on the level of JavaScript events.

Based on the Web usage data stored in the user model the rule engine evaluates the adaptation rules. We implemented a stateless rule evaluation based on the sequential algorithm [17]. The rationale behind this approach is that each of the independent rules fires once its conditions hold. There is no agenda to resolve any eventual conflicts caused by executing rules. Furthermore, this approach implies that the rules do not affect each other. Despite the disadvantages of this approach we chose the sequential algorithm because of its simple loop-like implementation. Once a rule has fired, the rule body, in most of the cases translated SWRL built-ins, is executed by the UI adaptation component and the user interface is manipulated.

At the end of the session the user model is sent back to the server using the asynchronous communication facility of AJAX. The accumulated user models form the basis for a further modeling cycle.

6 Evaluation

The implementation of the conceptual framework was realized using Java libraries for the design-time code generation and AJAX for the run-time components. We evaluated our prototypical implementation at different levels. First, we looked at the time consumption of rebuilding the ontologies and executing the adaptation rules at runtime on the client-side. Secondly, we evaluated our approach theoretically. That means we looked at the JSON format representing ontologies and rules and we also examined restrictions imposed by our rule execution algorithm.

6.1 Computational Evaluation

The design-time modeling of ontologies and rules, as well as the subsequent translation into JSON is the non-time critical part of the application. The transformation from RDF/XML syntax into JSON rules and the mapping of OWL concepts, instances and relations into JSON occurs at design-time. But already at this stage optimization is a crucial issue. The preparation of the JSON file for later usage on client-side requires an effective mapping-method to keep the amount of data represented on the client to a minimum. The compressed JSON format, as the result of translating ontologies, annotations and rules, lets the file size shrink to 50% of it original size. The file size decreased from 42,1 KB (RDF/XML) to 20,3 KB (JSON).

The more time critical issues are the run-time tasks, like the initial loading and creation of OWL concepts on client-side as well as the execution of rules and user interface adaptation on the client-side. At the time of accessing the RIA the ontolgies, annotations and rules have to be up-loaded to the client in a first step. The JSON file is executed using the JavaScript function *eval()* and concepts, instances and rules are represented as JSON objects on the client-side. An evaluation of this initial client-side concept creation is depicted in the following Figure 5 a). The initial transfer and construction of the rules and ontologies does not affect the usability too much, since it takes place within the first couple of seconds a user accesses a new page which is a time period of almost no interaction. As the diagram shows, the time consumption is below 200 ms for up to 10000 concepts, which is not recognized by the user when loading the page.

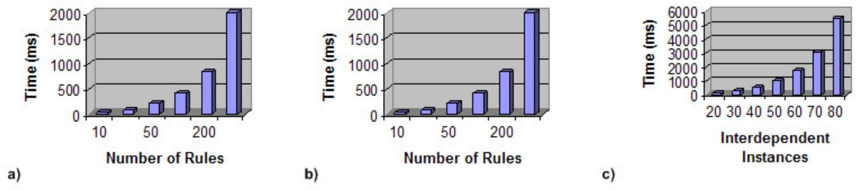


Fig. 5. Evaluation: a) Client side initial JSON concept creation; b) Rule execution time evaluation; c) Interdependent instances

During run-time the most important task is the execution of rules and the subsequent adaption of the user interface. The client-side rule engine is implemented as a stateless sequential algorithm. The rules are mapped to nested *IF/THEN* statements and executed in the order they are received. There is no conflict solving, complex event processing or any kind of inference during the execution since rules cannot be triggered by another rule. Each rule is evaluated and, if all conditions hold, the body (action) of the rule is executed. In Figure 5 b) the performance of the rule execution is evaluated. As an evaluation constraint we let each rule fire, that means that all conditions of our evaluation rule set hold. Each rule manipulates the user interface of our exemplary RIA. The evaluation of each rule starts with loading the rule from the JSON structure into a local variable. This loading process is realized with a non-optimized algorithm and needs refinement. A big system performance improvement will be reached by revising this algorithm which occupies most of the execution time. Future development will also include the evaluation and integration of more matured rule execution algorithms like Rete [18] to achieve better performance.

The number of instances of ontology concepts have a considerable bearing on the execution time of the application rules on the client-side. The most time-intensive parts of instance-handling are *N-to-M* relations between instances. The rationale behind this is: If there is one instance Ia of concept A and one instance Ib of concept B it is easy to utilize a rule because there is only one relation between the two instances. But if there are several instances $Ia - n$ for concept A and several instances $Ib - n$ for concept B the rule engine has to evaluate the Cartesian product of the instances, which leads to an exponential time consumption. Figure 5 c) depicts the time consumption of rules coping with interdependent instances in more detail. However, in our motivating examples we did not have to deal with that, because of carefully designed adaptation rules.

During the performance evaluation a slight distinction in the measurement results between the two tested web browsers (Internet Explorer[5] and Mozilla Firefox[6]) has been determined. The diagrams are based on the average measurement results of both web browsers.

6.2 Theoretical Evaluation

First we evaluated the JSON format we created in order to represent OWL ontologies. Our JSON ontology serialization is conceived to minimize time of accessing instances at the client-side. The format puts some limitations on the representation of ontologies. On the other side, these limitations have no restrictive effects to the overall approach of client-side adaptation of RIAs as we solely rely on the concept taxonomy in our adaptation rules. By computing the subsumption hierarchy at design-time we can construct the entire class hierarchy graph. However, in doing so we lose all informations regarding OWL class axioms like equivalent classes and class descriptions like union or intersection. All axioms and descriptions are mapped to a simple sub class relation. Also all information about relations are lost. Relations only appear as attributes in objects and are no longer represented as discrete entities. Only individuals are transformed without any information loss. But as already mentions, as consistency checks are performed at design-time and the adaptation rules only rely on instances, their relations, and class hierarchy, this puts practically no constrains to our approach.

In a second step we evaluated the gains and losses of the sequential algorithm implemented by our client-side rule engine. One advantage of the sequential algorithm is its simplicity. It is quick to implement and easy to maintain. Furthermore, in the computational evaluation, we showed the feasibility of our approach even with a simple rule evaluation strategy. But the sequential algorithm does not come for free. So, we have exponential time consumption when evaluating the Cartesian product of class instances. This could be reduced to by using the forward-chaining Rete algorithm for the evaluation of the adaptation rules. Rete as efficient pattern matching algorithm would introduce real inferencing and a stateful rule evaluations. Our current implementation of the client-side rule engine imposes some restrictions to the design of the adaptation rules. So, the rule engine can only evaluate 2-ary predicates which have a model consisting exactly of pairs of the form $(a, b_i)_{i \in I}$ or conversely $(a_i, b)_{i \in I}$, where I

[5] http://www.microsoft.com/windows/products/winfamily/ie/default.mspx
[6] http://www.mozilla.com/en-US/firefox/

is an arbitrary finite index set. In the case of $(a_i, b_j)_{i,j \in I}$ the outcome is currently undefined. We are working on a solution to that. Furthermore all variables have to be explicitly introduced as we can not conclude the type of a variable for its occurrence in an 2-ary predicate. This means that in the Example 1 the atoms portal:Form($?a$), domain:WeddingDay($?b$), domain:MarriageCertificate($?d$) and domain:BirthCertificate($?e$) are mandatory. Also the order matters. These atoms have to occur before the variables are used in an arbitrary 2-ary relation.

One of our goals is the autonomous working of the application (without client-server communication). It means that our main instrument for the detection of events is JavaScript. This fact constitutes a restriction with respect to complex events. Suppose that some complex event CE is modelled as the conjunction of two other not necessarily atomic events E_1 and E_2. We must have a rule $E_1 \wedge E_2 \rightarrow CE$ expressing this situation. It is possible to create a new instance for the complex event CE[7], however the application remains unaware of the latter, since it was not detected by JavaScript. We have solved this issue by coding the occurrence of such a complex event in some attributes of the existing instances, and then replacing the above rule by a checking of the attributes. This fact remains, however, a handicap toward an elegant and natural modeling of the rules. The above evaluation confirms our presentiment in [4], SWRL is in fact not sufficient for our purposes. It does not constitute a suitable framework for the modeling of complex rules. The simulation of the *seq* operator does not allow an efficient processing of the events and rules. A recent investigation has shown that Event-Condition-Action (ECA) rules[8] may be more appropriate.

7 Related Work

In [19] the integration of semantics in Web usage mining techniques is shown applied to a movie website. On the basis of a movie ontology and the user behaviour, user profiles were constructed, which are used for online recommendations. In the center of our aproach are e-Government websites consisting of forms, services and information.

Comparing our work with standard models for adaptive hypermedia systems like e.g. AHAM [20], we observe that they use several models like conceptual, navigational, adaptational, teacher and learner models. Compared to our approach, these models correspond to ontologies presented in Section 4, but miss their formal representation. Moreover, we express adaptation functionalities as encapsulated and reusable OWL-DL rules, while the adaptation model in AHA uses a rule based language encoded into XML.

The Personal Reader [21] provides a framework for designing, implementing and maintaining Web content readers, which provide personalized enrichment of Web content for each individual user. The adaptive local context of a learning

[7] This situation is even impossible with respect to the ontology. However, we make use of SWRL bultins to achieve the addition of new instances to the ontology.

[8] ECA rules are event driven rules in the form of: *ON* (event) *IF* (condition) *DO* (action).

resource is generated by applying methods from adaptive educational hypermedia in a semantic Web setting. Similarly [22] focuses on content adaptation, or, more precisely, on personalizing the presentation of hypermedia content to the user. However, both approaches do not focus on the on-line discovery of the profile of the current user that is one of the main features of our approach. Another difference would be the self-adaptivity.

In [23] the authors suggest 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 not only adapting the content based on concept similarity but rather based on accumulated Web usage data. Furthermore we show a way how to link semantics and content. Still the main difference remains the introduction of the autonomous client, as we are dealing with Rich Internet Applications and not with common dynamic Web applications executed on a Web server.

8 Conclusions and Future Work

In this paper we presented a novel approach that holistically combines page annotations, semantic Web usage mining, user modeling, ontologies and rules to adapt AJAX pages. We showed and evaluated how our concept of an autonomous client works. With our prototypical implementation we demonstrated the proof of concept and our motivating example taken form the e-Government domain emphasized the practical relevance of our work. Currently, we are investigating the Rete algorithm with respect to its adoption for client-side rule processing. We expect a better run-time performance by substituting the sequential algorithm with the Rete algorithm. Furthermore we plan to extend SWRL from deduction rules to event ECA rules to allow complex event processing directly on the client side. Then the transformer component of the design-time cycle can automatically generate event handlers based on the ECA rules.

Acknowledgements

The work is based on research done within the FIT project - Fostering self-adaptive e-Government service improvement using semantic technologies. The FIT project is co-funded by the European Commission under the "Information Society Technologies" Sixth Framework Program (2002-2006).

References

1. Brusilovsky, P.: Methods and techniques of adaptive hypermedia. User Model. User-Adapt. Interact. 6(2-3), 87–129 (1996)
2. Mobasher, B., Cooley, R., Srivastava, J.: Automatic personalization based on web usage mining. Commun. ACM 43(8), 142–151 (2000)
3. Garrett, J.J.: Ajax: A new approach to web applications (2005), http://www.adaptivepath.com/publications/essays/archives/000385.php

4. Schmidt, K.-U., Stojanovic, L., Stojanovic, N., Thomas, S.: On enriching ajax with semantics: The web personalization use case. In: Franconi, E., Kifer, M., May, W. (eds.) ESWC 2007. LNCS, vol. 4519, pp. 686–700. Springer, Heidelberg (2007)
5. Thomas, S., Schmidt, K.-U.: D4: Identification of typical problems in e-government portals. Technical report, FIT consortium (July 2006), `http://www.fit-project.org/Documents/D4.pdf`
6. Maedche, A., Staab, S.: Semi-automatic engineering of ontologies from text. In: Proceedings of the 12th International Conference on Software Engineering and Knowledge Engineering (2000)
7. Stojanovic, L., Ma, J., Stojanovic, N.: D9: Methods and tools for semi-automatic learning of a domain ontology that models the content of a front office. Technical report, FIT consortium (January 2007), `http://www.fit-project.org/Documents/D9.pdf`
8. Stojanovic, L., Stojanovic, N., Ma, J.: An approach for combining ontology learning and semantic tagging in the ontology development process: egovernment use case. In: Benatallah, B., Casati, F., Georgakopoulos, D., Bartolini, C., Sadiq, W., Godart, C. (eds.) WISE 2007. LNCS, vol. 4831, pp. 249–260. Springer, Heidelberg (2007)
9. Bechhofeer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D., Patel-Schneider, P., Stein, L.A.: Owl - web ontology language reference. Recommendation, W3C, February 10 (2004)
10. Dai, H., Mobasher, B.: Using ontologies to discover domain-level web usage profiles. In: 2nd Semantic Web Mining Workshop at ECML/PKDD-2002 (2002)
11. Berendt, B., Hotho, A., Mladenic, D., van Someren, M., Spiliopoulou, M., Stumme, G.: A roadmap for web mining: From web to semantic web. In: Berendt, B., Hotho, A., Mladenič, D., van Someren, M., Spiliopoulou, M., Stumme, G. (eds.) EWMF 2003. LNCS (LNAI), vol. 3209, pp. 1–22. Springer, Heidelberg (2004)
12. Stumme, G., Hotho, A., Berendt, B.: Semantic web mining: State of the art and future directions. Semantic Grid –The Convergence of Technologies 4(2), 124–143 (2006)
13. Spiliopoulou, M., Mobasher, B., Berendt, B., Nakagawa, M.: 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 (2003)
14. Stojanovic, L., Ma, J., Yu, J., Stojanovic, N., Schmidt, K.-U., Thomas, S., Rahmani, T.: D18: Methods and tools for mining the log data by taking into account background knowledge. Technical report, FIT consortium (July 2007), `http://www.fit-project.org/Documents/D18.pdf`
15. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M.: Swrl: A semantic web rule language combining owl and ruleml. Technical report, W3C Member submission, May 21 (2004)
16. Crockford, D.: Rfc4627: Javascript object notation. Technical report, IETF (2006)
17. Berstel, B., Bonnard, P., Bry, F., Eckert, M., Patranjan, P.-L.: Reactive rules on the web. In: Antoniou, G., Aßmann, U., Baroglio, C., Decker, S., Henze, N., Patranjan, P.-L., Tolksdorf, R. (eds.) Reasoning Web. LNCS, vol. 4636, pp. 183–239. Springer, Heidelberg (2007)
18. Forgy, C.L.: Rete: a fast algorithm for the many pattern/many object pattern match problem. Artificial Intelligence 19, 17–37 (1982)
19. Dai, H., Mobasher, B. (eds.): Using Ontologies to Discover Domain-Level Web Usage Profiles (2002)

20. Romero, C., Ventura, S., Herváas Martinez, C., De Bra, P.: In: Proceedings of the Fifth International Conference on Human System Learning, ICHSL, Europia (November 2005)
21. Dolog, P., Henze, N., Nejdl, W., Sintek, M.: The personal reader: Personalizing and enriching learning resources using semantic web technologies. In: De Bra, P.M.E., Nejdl, W. (eds.) AH 2004. LNCS, vol. 3137, pp. 85–94. Springer, Heidelberg (2004)
22. Frasincar, F., Houben, G.-J.: Hypermedia presentation adaptation on the semantic web. In: De Bra, P., Brusilovsky, P., Conejo, R. (eds.) AH 2002. LNCS, vol. 2347, pp. 133–142. Springer, Heidelberg (2002)
23. Ankolekar, A., Tran, D.T., Cimiano, P.: Rules for an ontology-based approach to adaptation. In: 1st International Workshop on Semantic Media Adaptation and Personalization, Athen, Greece (December 2006)