Computing and Informatics, Vol. 28, 2009, 399–427, V 2009-Sep-22

RULE-BASED USER CHARACTERISTICS ACQUISITION FROM LOGS WITH SEMANTICS FOR PERSONALIZED WEB-BASED SYSTEMS

Michal Barla, Michal Tvarožek, Mária Bieliková

Institute of Informatics and Software Engineering
Faculty of Informatics and Information Technologies
Slovak University of Technology in Bratislava
Ilkovičova 3, 842 16 Bratislava, Slovakia
e-mail: {michal.barla, michal.tvarozek, maria.bielikova}@fiit.stuba.sk

Revised manuscript received 24 February 2009

Abstract. Personalization of web-based information systems based on specialized user models has become more important in order to preserve the effectiveness of their use as the amount of available content increases. We describe a user modeling approach based on automated acquisition of user behaviour and its successive rule-based evaluation and transformation into an ontological user model. We stress reusability and flexibility by introducing a novel approach to logging, which preserves the semantics of logged events. The successive analysis is driven by specialized rules, which map usage patterns to knowledge about users, stored in an ontology-based user model. We evaluate our approach via a case study using an enhanced faceted browser, which provides personalized navigation support and recommendation.

Keywords: User modeling, ontologies, log analysis, usage pattern, adaptive faceted browser

Mathematics Subject Classification 2000: 68T05, 68T10, 68Nxx

1 INTRODUCTION

Both growing end user expectations and requirements as well as commercial considerations drive the demand for ever improving state-of-the-art solutions in (web)

application development. In this respect, several contemporary initiatives pursue the advancements in their respective fields of interest with the ultimate aim of improving end user computing experience:

The Adaptive Web focuses on personalized services tailored to the specific needs of individual users (i.e., the user context, composed of user characteristics, device and environment properties, etc.) in terms of content recommendation, navigation adaptation and presentation customization. It is a response to the "one-size-fits-all" approach which became increasingly unsuitable as the amount of available information, its complexity and the diversity of its consumers increased. Consequently, the Adaptive Web addresses issues such as information overload, the "lost in hyperspace" syndrome as well as usability problems and performance aspects [6, 7].

The Semantic Web as envisioned by Tim Berners-Lee aims at improving upon the existing Web by adding a semantic layer of metadata (i.e., "meaning"), which would allow for advanced machine processing of information and thus ultimately improving application interoperability, data integration, sharing and availability. This is accomplished by using ontologies (e.g., in RDF/RDFS or OWL), which describe concepts and their relations with defined semantics using unique identifiers – URIs, for data representation and reasoning [29].

Since each initiative aims at improving end user experience by addressing a somewhat different set of problems, the exploration of hybrid solutions taking advantage of the synergetic effects of combining advantages of individual approaches is vital for future research and its practical applications.

In practice, user needs can be classified as informational, navigational and transactional [4]. Levene and Wheeldon describe four steps that are required to satisfy either kind of need – the query, selection, navigation, query modification [22]. If the target site is known beforehand (e.g., a home page or bookmarked page), users iterate only through the selection and navigation steps. While the query, selection and query modification steps are supported by existing search engines, these currently do not aid users during the crucial navigation step in which users browse the information space and locate the required information.

The difficulties of navigation are even more pronounced in large heterogeneous information spaces, where each subspace has different properties. If we consider the Web, then finding specific information typically involves a query in a web search engine, the selection of a suitable web page – often the root of a web site, and navigation through the web site towards pages containing the required information. However, each web site has a different layout and presentation style, which also change over time making effective navigation difficult.

In order to address user support during the critical navigation step we propose a solution based on the aforementioned initiatives. We take advantage of the Adaptive Web approaches and provide automated user model creation with minimal

user involvement, thus preventing disruptions to user experience. The user model thereafter serves as a base for personalized navigation.

We gather semantically enriched evidence of user interaction via a user modeling server, which is then evaluated by inference agents and stored as semantically described user characteristics in the user model. These are then used to drive the personalization engine of the presentation layer, which performs user adaptation based on the acquired user characteristics and also collects new evidence of user interaction within the browser.

The rest of the paper is structured as follows. Section 2 describes current approaches to both data collection and data processing as two major stages of the user modeling process. In Section 3 we describe our ontology-based user model. Section 4 contains description of our approach to the collection of evidence about user actions. In Section 5 we describe the background knowledge used in our user modeling approach, while Section 6 presents the user modeling process itself. Section 7 describes our evaluation environment and elaborates on the experiments we performed to evaluate our approach. We summarize our contribution in Section 8.

2 RELATED WORK

The adaptation process in adaptive systems consists of three stages (see Figure 1):

1. collection of data about users,
2. creation of the respective user models, and
3. adaptation based on the created user models.

Moreover, since the whole process works in an endless loop, the user model is constantly updated as the system collects more data about users.

In this paper, we present contribution to the first two stages of the process. Data collection is obviously based on user activity logging where two main approaches exist: the use of standard web server logs or the deployment of proprietary logging/reporting solutions often tailored to the needs of specific systems. User model creation is based either on explicit user feedback or on automatic discovering user characteristics based on collected data. The last, personalization stage is only briefly mentioned in connection with evaluation of our approach to automated acquisition of user characteristics.

2.1 Standard Web Server Logs

Standard web server logs are commonly used as input for various data mining techniques in *Web Usage Mining* [13] whose results are mostly common sequential patterns or clusters of users and pages as was shown for example in [24]. These techniques match the active user session (or its previously stored profiles) to usage

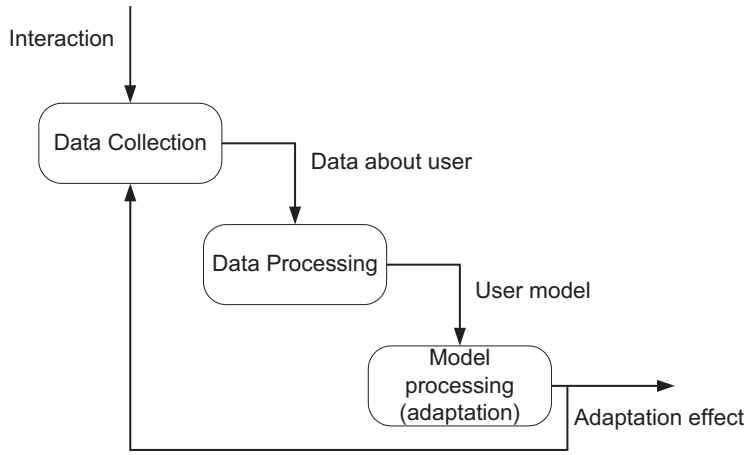

Fig. 1. Loop "user modeling—adaptation" in adaptive systems, according to [5]

patterns of user groups. Because of their social aspect, they do not reveal characteristics of an *individual* user, which is of our prime interest.

Another drawback of web server logs comes from their processing complexity, incompleteness of information and system dependency. Web server logs cannot be used directly and require very complex preprocessing, trying to identify all accesses of an individual user within a particular session [10]. Use of proxy servers and tunnels makes this task non-deterministic, producing only estimations of log interpretation.

Web server logs are often incomplete due to the caching mechanism of web browsers, which usually do not re-request a resource which was accessed earlier during previous sessions. As a result, server side logs cannot serve as a satisfactory standalone source of all user behaviour evidence.

Web server log processing is entirely tied to a system's implementation, because of the basic principles of the HTTP protocol, which is a low-level stateless protocol without clearly defined semantics of performed actions (e.g., GET and POST do not supply adequate information). Furthermore, a web-based *adaptive* system with dynamic page generation can produce two similar or even equal log entries which result in completely different system responses due to changes in the user and domain models.

To address some shortcomings of this approach and to capture information about purely client-side interactions (e.g., hovering on tooltips) client-side logging methods were proposed. Most are either pure client-side only systems [23] or standalone desktop applications communicating with a server [15]. Standalone client-side applications can provide a high level of detail but may present serious threats to user

privacy. The use of standard client web technologies such as JavaScript or Java applets [14] appears to be more suitable due to its restricted scope of operation and higher user acceptance though still without event semantics.

Pure client-side only solutions suffer from the loss of control over the logging process as users can disable their execution. Therefore, it is suitable to combine both client side logging with server side logging.

2.2 Proprietary Logging Solutions

Adaptive applications often imply tight integration of proprietary and application tailored data collections and data analysis subsystems. However, there are efforts that aim at separating the user model and enable its reusability across several adaptive applications in the form of user modeling server. We can thus analyze the logging (often called reporting) subsystems of these servers, which gather logs from several (adaptive) applications.

Personis [20], based on the UM toolkit [19], does not expect applications to report information about user actions, but already partial user models, consisting of the *component* and *evidence* elements. A component represents either preference, knowledge, belief or a simple attribute, depending on its type. It has a name, value and is either supported or negated by one more evidences (where each evidence has a given reliability).

The Cumulate user modeling server [8, 38] for the educational domain is an example of a server, which uses event reporting. It provides a servlet (report receiver) which listens to applications reporting events about user activity. Each request contains an identification of the *reporting application*, reference to a *learning object/action* the user has been working with, a *fragment* of learning object/action (where applicable), identification of the *user*, *group* and user *session*. Finally, it contains the *result* of interaction (successful or unsuccessful). Applications may also report custom string values, which are simply stored for later use and are not processed by the server.

2.3 User Model Creation

Data processing which leads to a user model update is obviously limited by the nature of the acquired data and the used representation of the user model. Many approaches exploit explicit user feedback (e.g., level of knowledge or interest in a given topic) and transform this feedback directly into the user model. An enhancement of this approach is based on the spreading activation model, which spreads the feedback from one particular concept of the respective domain model to other associated concepts [30].

Some approaches leverage *machine learning* techniques to create and maintain user models [35, 16]. Certain approaches use predefined model schemes and try to fill them with attributes, while others infer both the structure and attributes of the model, e.g., using clustering and classification techniques. A disadvantage of

most approaches based on machine learning techniques is the need for large data sets. They require not only a complex training data set, but need to see a relatively large number of examples from the user in order to create a model with acceptable accuracy.

Due to the uncertainty of the user modeling process, many solutions rely on probabilistic approaches. The uncertainty comes from the difficulty of assessing the reasons behind a user actions in complex environments such as the Web. Likely the most widely used probabilistic approaches to user modeling leverage Bayesian networks [27, 9]. Other known probabilistic approaches are based on Dempster-Schafer theory of evidence [18] and fuzzy logic [12]. However, a model construction using probabilistic approaches is not a straightforward process, requiring a lot of training data. Many proposed methods for model construction are theoretical only, or applicable only for small scale problems.

Many approaches do not build an explicit user model containing references to the domain model, but use pre-trained classifiers and predictors to perform adaptation, e.g., to recommend a next page to visit. The classifiers are either *content-based* or *collaborative*, taking into account the domain content and the user's interaction with it or the experience of other – similar users respectively. The classifiers are trained using user-specific data and thus correspond to an implicit user model. An example of such approach can be found in [39]. The authors propose a specialized web browser with enhanced tracking features (detailed client-side logging) collecting the list of visited web pages along with the user's feedback whether the page contained useful information (information content page). Consequently, they extract some domain and site-independent features from the pages such as the page type (search engine, static, dynamic etc.), URL depth or user clickstream-related page features (does the page follow the search engine page?, is it the last page in site session?, how many pages have been visited since the last information content page? etc.). The authors perform similar processing also on the content level of pages (i.e., text). All these features are used to train standard classifiers such as C4.5 or Naïve Bayes used during the adaptation process.

Another example of an implicit user model creation is presented in [26], where a user model represented by a vector is created by using various attributes of documents read by the user. This user model serves as an additional context used for enhancing (re-ranking) the search results returned by an ordinary search engine.

3 ONTOLOGY-BASED MODEL OF USER CHARACTERISTICS

The user model stores all persistent information about users which serve for adaptation purposes. We employ an ontology-based representation of the model [1] and take advantage of RDF/OWL technologies to define classes of user characteristics and their properties. The idea of ontological representation of user model is not new [11, 17]. Probably the main advantage is the possibility to use standard inference mechanisms based on description logic to infer additional knowledge about

users. Another significant advantage is the simplification for exchanging user model between different user-adaptive systems, if they agree on the used vocabulary and structure. Such an agreement might be replaced by an ontology mapping mechanisms such as the one presented in [36], which would however still need a human intervention to control the results and to repair potential errors [34].

Our model consists of two parts. The domain independent part defines characteristics like age or sex as well as the overall structure of a user characteristic and could be therefore easily reused across domains and applications. The domain independent part can be associated with multiple domain dependent models, which reflect user characteristics typical for a particular domain and which reference concepts from the respective domain models.

Each type of characteristic is derived from the *Characteristic* class, which defines the common attributes of characteristics (Figure 2):

- timestamp – the date and time of the characteristic's last update;
- goal to which it contributes (e.g., find a set of suitable publications);
- relevance of the characteristic to accomplishing a given goal;
- confidence – the quality of the characteristic's estimation;
- count of updates – how many times the characteristic was updated;
- source – defines the inference agent which updated the characteristic.

In this paper, we focus on two characteristic types:

AttributePreference – which describes user preferences with respect to specific domain properties (e.g., the *duty location* property of a job offer may be important for a user);

RuleCharacteristic – which describes user preferences also with respect to *values* (object type or data type) of properties (e.g., the preferred duty location *is* Bratislava, Slovakia).

Our approach to user model representation is similar to the one described in [3]. We are not storing the user specific information in relation with one specific job offer, but rather to a particular *attribute of that job offer*, which is surely repeated in more job offers within the whole information space. Therefore, we can take advantage of the model even in the case when the original job offer is no longer available.

The advantage of the proposed model structure is its enhanced reusability. Mentioned characteristics could be used in any domain with more complex ontological representations such as the job offers or publications domain, where we can expect many attributes along with their possible values. User model consisting of such characteristics can serve as a basis for personalization of navigational tasks as well as for personalized recommender systems.

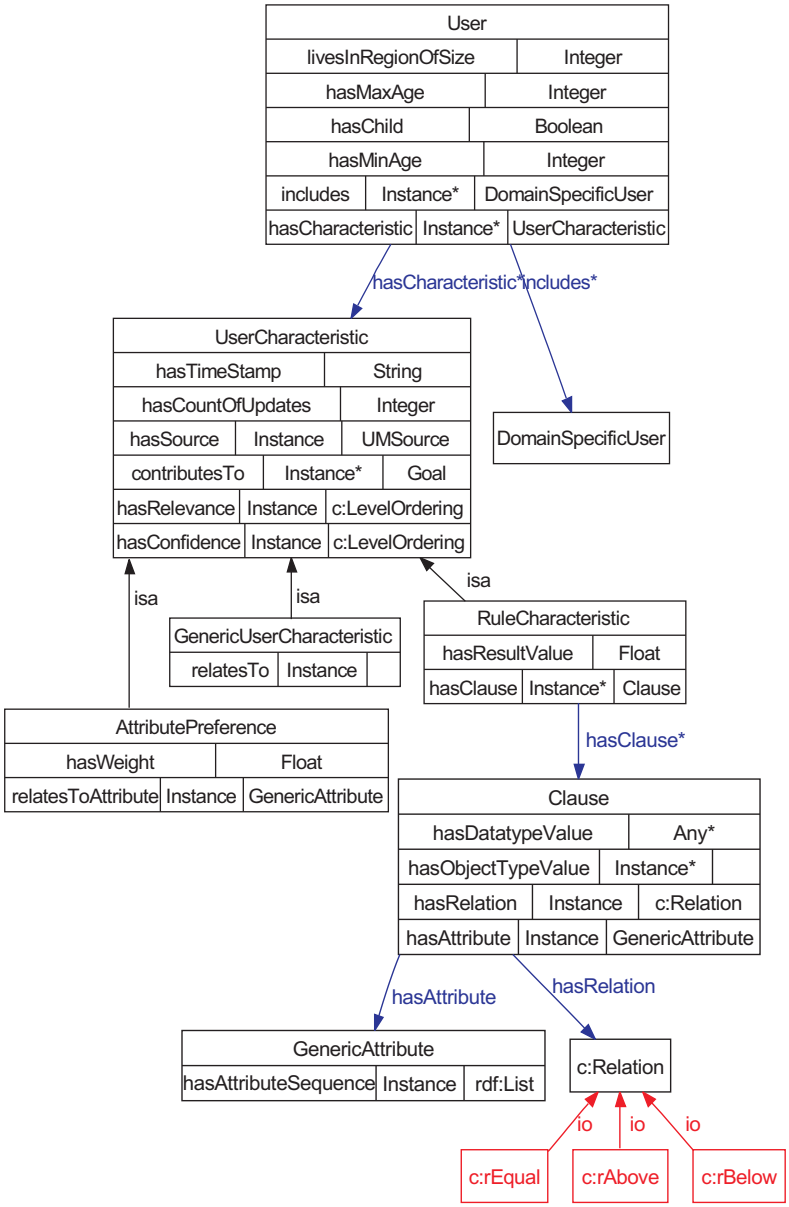

Fig. 2. Representation of a user characteristic in the used user model

4 USER ACTIVITY LOGGING WITH SEMANTICS

Standard web server logs are unsuitable for the estimation of individual user characteristics, since they are often incomplete, require complicated preprocessing and lack the semantics of performed actions. Furthermore, if the presentation layer of an adaptive web-based information system is comprised of several cooperating presentation tools [31] (Figure 3), the events occurring in each of them should be captured in order to allow for the discovery of meaningful user characteristics.

We propose a proprietary logging subsystem combining and enhancing both client-side and server-side logging approaches to create a comprehensive log of user actions while also preserving the semantics of individual actions (Figure 3, bottom right). We believe that the combination of the two approaches is necessary, since we have no direct control over the execution of client-side logging mechanisms (no data can be gathered if the user disables monitoring on the client side). The combination with the server side logging mechanisms guarantees that at least some data is always available.

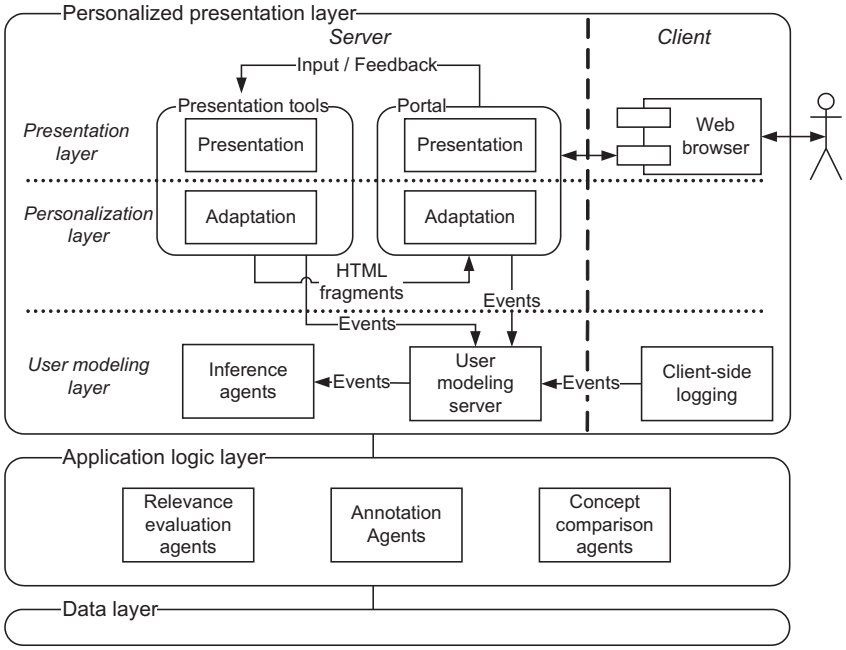

Fig. 3. Web-based IS architecture of the personalized presentation layer

4.1 Client-side Logging

Client-side logging captures precise time-related data describing individual actions (events) that might be missed by server-side logging. The monitored events include:

- Load: when a page is being displayed to the user,
- Unload: when the user leaves a page,
- Click: when the user follows a hyperlink on a page,
- Mouseover: when the user points the cursor at an active page element,
- Mouseout: when the user moves the cursor away from an active page element.

For each captured event, we collect the following data: *type of event*, *time* when it occurred and the *context* of the captured event, e.g. what link was followed.

Furthermore, additional events invisible to the server-side might be captured such as *Change*, invoked when the content of a form control changes. The sequence of such events indicates the order in which a user fills a form. Another example is the *Scroll* event, invoked when the user scrolls the content of a page.

4.2 Server-Side Logging

We propose a server-side event logging method that supports the logging of events with defined semantics by both server-side and client-side tools, and their integration into continuous streams of events for a particular user and user session. Moreover, since only individual presentation/interaction tools "understand" the semantics of events that they process we propose them to be responsible for the logging of their own events by means of a specialized logging interface. Consequently, server-side logging aggregates events from various sources and stores their semantics which mostly include references to concepts from a domain ontology.

A key point of our logging approach, which further separates log analysis from log creation, is the use of a common *event ontology* which defines the semantics of individual events and their attributes (see Figure 4). Furthermore, to evaluate the reasons behind user actions we also log the logical display state of the user interface at the time when an event took place. This allows us to analyze user decisions (i.e., the occurred events) based on what the user saw in the web browser interface (i.e., what her reasons were).

If for instance the system was providing navigation in the information space of job offers, one event could be to display details about a specific job offer, while another might be to show only job offers in New York. For both of these events, the meaning of the action would be logged (i.e., URI of the respective event) together with attributes such as the URI of New York. Lastly, the logical display state would be logged indicating amongst others which other job offers were displayed when the user chose the details of a specific one.

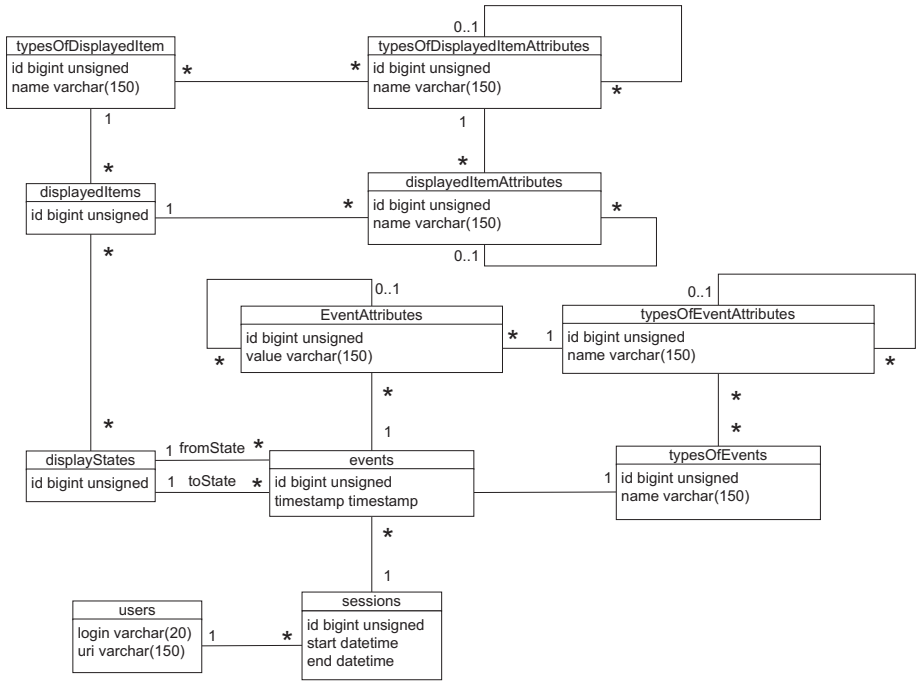

Fig. 4. Data model of logs with semantics. The model consists of two layers: meta-layer defining types of events, displayed items, their attributes and relationships among them; and operational layer, which holds the actual data. Displayed items together form a display state, a semantic description of what was displayed on the screen when the event occurred. The result of an event is a transition between two display states.

5 USER BEHAVIOUR PATTERNS

Produced logs of user actions with semantics serve as an input to the process of user model creation and maintenance. Our approach emphasizes the reusability aspect in both user modeling process as well as in the user model itself.

We designed the user modeling process to be independent from the adaptive applications which contributed to the logs of user actions, working only with the semantics of individual events from aforementioned event ontology. This allows us to employ several reasoning agents, which process records of user interactions and update the user model. The agents can "compete" against each other (i.e., the best estimation will be used) or cooperate and use the results from other agents to improve the estimation of user model.

We designed and implemented an inference agent, driven by a rule formalism (see Figure 5) capturing interesting patterns of user-system interaction. All knowledge

the agent requires in order to process the log of events and update the user model is stored in these rules, thus making the agent highly configurable and reusable. Each rule consists of a pattern and a consequence part as follows.

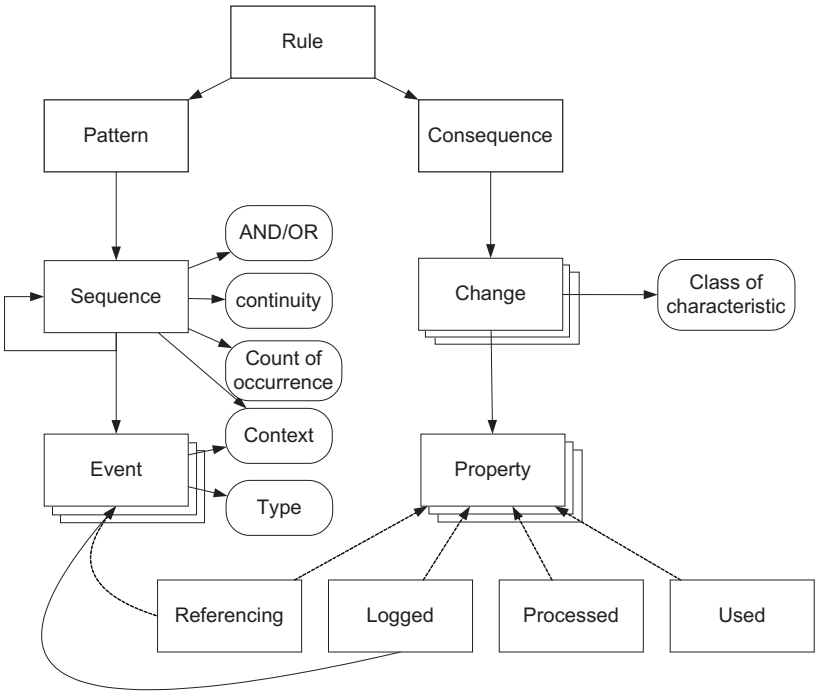

Fig. 5. Structure of rules used for estimation of characteristics. A pattern (left) consists of sequences of events with various attributes and restrictions. A consequence (right) is a set of changes applied on properties of particular characteristic instance in the ontological user model.

5.1 Pattern

Our data analysis approach is based on predefined patterns detected in the user action log, which at the top-level are defined as sequences of event types and other subsequences. A pattern is detected when events prescribed by a sequence are successfully mapped to specific events found in the user event log.

Sequence. A sequence can require the occurrence of all its events and subsequences (equivalent to the logical AND function), or the occurrence of just one of its events or subsequences (equivalent to the logical OR function). Furthermore, we divide sequences into *continuous* and *discrete*. A continuous sequence requires that

all of its events must succeed each other without interruption while the events of a discrete sequence can be separated by any number of other events and sequences. A sequence can thus span through multiple user sessions.

We define the following sequence attributes:

Count-of-occurrence prescribes the required count of sequence repetitions in a pattern. The execution engine will continue to process the next sequence only if this count was reached. This attribute is also used to define an optional sequence as optional.

Context is an optional attribute which defines the restrictions for events mapped to the current sequence. For example, a context restriction can define the types of displayed items' attributes which must stay constant for all events mapped to the sequence.

Event. An event represents an elementary part of a pattern. During pattern detection, we map events from the user activity log to events prescribed by patterns. The type of each event corresponds to a known event type from the meta-level of the event model. Each event can be assigned a weight determined by considering various factors such as time to the next event or a predefined constant.

Similarly to sequences, events can also have contextual restrictions, which define restrictions solely on the attributes of the event while the context of a sequence deals with changes of displayed items (i.e., the display state).

SameAsPrevious/*DifferentThanPrevious* are event context conditions restricting the value of a defined event attribute to be the same as in/different from the previous event of a sequence. Another possible context condition *MinValueOfWeight* requires the weight of an event to be higher than some defined value. For instance, if an event "show detail" is immediately followed by a "show overview" event, it will be assigned a low weight, since it does not satisfy a defined contextual restriction (the user did not have the time to read the page with the details). Therefore, this event will not be mapped onto the sequence.

Figure 6 shows an example of the pattern *"result browsing"*. Let us consider a job offer repository and a user who is interested in job offers satisfying some criteria. When the user selects some restrictions on information space presented, a set of results that satisfy these restrictions is returned. The user can browse the set for further details.

At the top level a pattern is formed by a discrete *OneRequired* sequence. Our heuristic assumes that users *browse in the search results* if they perform a browsing action at least four times. That means the sequence has to be found four times in the user activity log for the pattern to be matched. The sequence also has a contextual restriction which refers to an attribute of a displayed item. All mapped events have to be connected to display states, which have for all displayed items of type *facet* a constant value of the actually chosen restriction. In other words, the user is not changing the currently selected restrictions and is only browsing the list of search results. Thus, events can be of the following types: *PageNext*, *PagePrevious*,

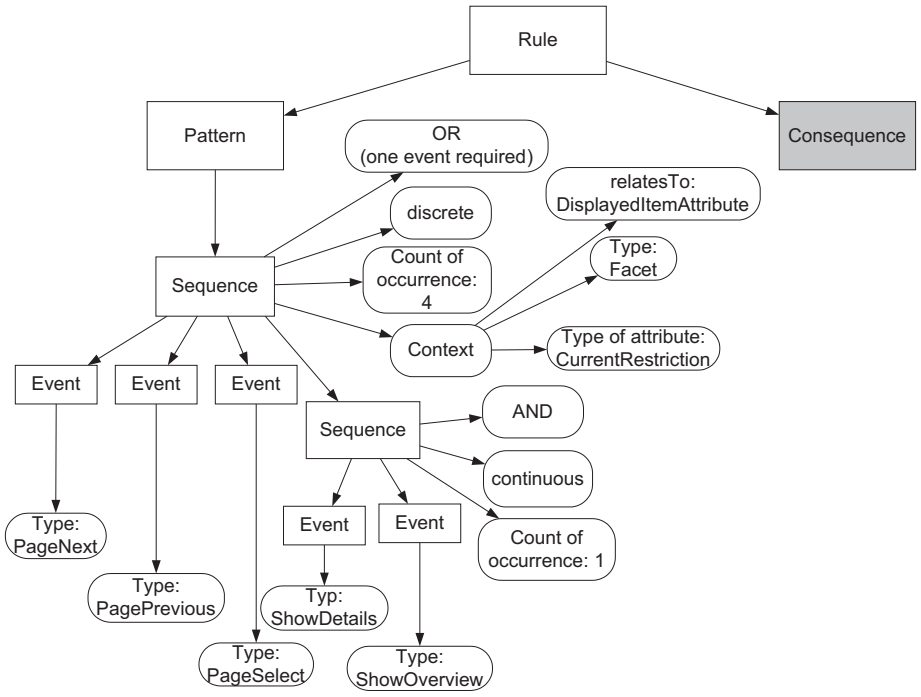

Fig. 6. Example of the pattern part of the rule *"results browsing"*

PageSelect, *ShowDetails* and *ShowOverview*. The former three event types represent navigation through individual results pages while the latter two, joined in a continuous subsequence, represent the display of details and the navigation back to the list of results.

5.2 Consequence

Consequences consist of an unlimited number of changes of user characteristics and determine what and how should be updated in the user model if an instance of a pattern is matched. Each change describes the type of the updated user characteristic (the *class* of the characteristic) and several *property* attributes prescribing the changes of object and data type properties of the respective characteristic instance. Changes can have the following properties:

Used property – directly defines the value of a property (i.e., a constant).

Processed property – used for numerical data-type properties such as confidence or relevance, it defines how to compute the value of a given property, such as

increasing/decreasing its existing value, what change strategy to use (e.g., progressive or uniform), or what increment/decrement and what interval boundaries (where the rule takes effect) to use.

Referencing property – references a given event attribute in the rule pattern and defines the property value as the value of that attribute. Referencing properties distinguish (optionally also with used properties) a characteristic, i.e., their values are used in order to retrieve an instance of a characteristic from the repository.

Logged property – references a given event attribute in the rule pattern and defines the property value as the value of that attribute, identically to any referencing property. However, a logged property is not used to retrieve the instance of characteristic and is therefore suitable for storing various context and numerical values, which are to be changed to values from the event log each time a characteristic is updated.

Figure 7 shows an example of the consequence part of the rule "result browsing". The consequence changes instances of the *RuleCharacteristic* and defines referencing properties which take their values from attributes of currently displayed items of the type *Facet*. Namely, we are interested in the attribute *CurrentDimension* which will be used as the type of stored value in the characteristic and in the attribute *CurrentRestriction*, which will serve as the value itself. Thus, we are interested in the restriction of information space the user has defined by selecting values in facets. The meaning of Used and Processed properties is understandable from the figure.

6 USER LOG ANALYSIS

Our method for log analysis uses the knowledge represented by the aforementioned rule-based mechanism. The analysis process consists of *entry preprocessing*, *pattern detection* and *user model update*.

6.1 Data Preprocessing

The nature of the collected data requires no complex preprocessing, since user sessions and accesses are already present in the log and thus need not be discovered, unlike in other solutions that use web server logs [10].

This stage consists mainly of filtering consecutive events of the same type and context, which often result due to user impatience and repeated clicks on the same item (e.g., because of slow system response times). In this stage, weights are assigned to individual events as described in the previous section.

6.2 Pattern Detection

Pattern detection is the key part of our log analysis approach. It works similarly to standard forward chaining production systems and maps events prescribed by rules

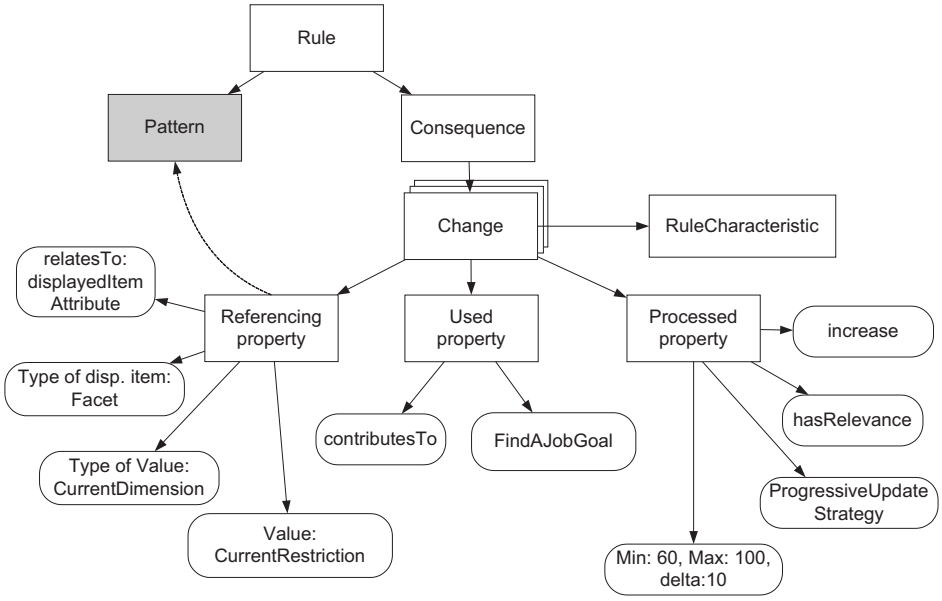

Fig. 7. Example of the consequence part of the rule *"results browsing"*

to specific events in the user action log. Events are mapped to instances of rules for each user, with each rule instance referencing the instances of its sequences in order to track their current *count-of-occurrence*. Outline of proposed method of pattern detection is as follows:

Method DetectPatterns()

Input: Event

```
1 candidateRules ← FindCandidateRules(Event);
2 foreach rule in candidateRules do
3 |   applicableRuleInstances ← FindApplicableRuleInstances(rule, Event);
4 |   foreach ruleInstance in applicableRuleInstances do
  |   ruleInstance.apply(Event);
5 end
```

FindCandidateRules eliminates rules (from the further processing) which have no potential to change the state of inference for the current user and processed event. *Candidate rules* are found as follows:

Step FindCandidateRules(Event) (*line No. 1 of DetectPatterns()*)

Output: *candidateRules*

```
1  foreach rule in knownRules do
2  |  if Event.type = rule.pattern.firstExpectedEvent.type then
3  |  |  candidateRules ← candidateRules + rule;
4  |  |  create ruleInstance of the rule for currentUser;
5  |  end
6  |  else if currentUser has ruleInstance of rule that
7  |  expectedEvent.type = Event.type then
8  |  |  candidateRules ← candidateRules + rule;
9  |  end
10 end
11 return candidateRules
```

After finding a set of *candidate rules*, we examine their *instances* belonging to the current user and decide whether they can be mapped to an event (map the event to the pattern part) or not. This process is described by the following pseudo-code:

Step FindApplicableRuleInstances() (*line No. 3 of DetectPatterns()*)

Input: *rule*, *Event*
Output: *applicableRuleInstance*

```
1  foreach ruleInstance of Rule belonging to currentUser do
2  |  checkContextOfCurrentSequence(Event);
3  |  checkContinuity(Event);
4  |  if all checks passed then
5  |  |  applicableRuleInstance ← applicableRuleInstance + ruleInstance;
6  |  end
7  end
8  return applicableRuleInstances
```

Having applicable rule instances, we map an event onto the pattern part and start the user model update process if the pattern was matched:

Step RuleInstance.Apply(Event) (*line No. 4 of DetectPatterns()*)

```
1  map(Event, RuleInstance.expectedEvent) ;
2  updateState(RuleInstance);
   // updating nextExpectedEvent, count-of-occurrences, ...;
3  if Pattern was detected then
4  |  performConsequence(RuleInstance);
5  end
```

6.3 User Model Update

The update of the user model is driven by changes specified in the consequence parts of rules. It performs these steps for each change:

Method UserModelUpdate()

```
1 characteristic ← RetrieveInstanceOfCharacteristic;
2 foreach property in processedProperties do
3 |   update value according to given strategy;
4 end
5 foreach property in loggedProperties do
6 |   update value according to log entry;
7 end
8 update timestamp;
9 update count-of-updates;
```

Step RetrieveInstanceOfCharacteristic() *(line No. 1 of UserModelUpdate)*

```
1 check value of all referencing properties;
2 check value of all used properties;
3 if source of characteristic set strictly to the current UM agent then
4 |   check value of source of characteristic;
5 end
6 if no instance fulfills these criteria then
7 |   create a new instance;
8 |   set all referencing and used properties;
9 |   set source;
10 end
11 return found or created instance;
```

7 EVALUATION

For evaluation, we developed *LogAnalyzer* – a prototype of our rule-based user modeling agent, which implements selected parts of the proposed user modeling method. Since LogAnalyzer relies on a user interface front-end for user interaction, we evaluated it as part of the personalized presentation layer proposed in [31] and integrated it with the personalized faceted browser – Factic [33], which also facilitates automatic user model acquistion via semantic logging of user interaction evidence.

We performed experiments in three different application domains – online job offers (project NAZOU [25], `nazou.fiit.stuba.sk`), scientific publications (project MAPEKUS, `mapekus.fiit.stuba.sk`) and digital images.

For each domain, we have constructed both a domain and a user ontology describing the main domain concepts and their properties. The job offer ontology had the most complex schema consisting of some 740 classes with hierarchical classifica-

tions up to 6 levels deep. The publication ontology was of medium complexity with only one hierarchical classification (the ACM classification), while the digital image ontology had a relatively simple flat schema.

We populated the ontologies with instance data of different sizes acquired from publicly available web resources (e.g., `www.careerbuilder.com`, `eurojobs.com`, `profesia.sk`, DBLP, Springer and ACM DL). We worked with manually/semi-automatically created "toy-size" datasets having 100 s–1000 s of instances to automatically acquired, large integrated datasets in excess of 100 000 s of instances and several times that many triples.

Our experiments indicated that our approach is best suited for the job offers domain due to its rich structure and broad user interaction possibilities. Therefore, we present the results achieved in the job offers domain.

We opted for formative evaluation based on the layered evaluation approach [28] and focused on two key aspects of our solution:

User modeling – how well and how fast can we estimate the user's real/perceived preferences using our automated rule-based user modeling approach. We examine the estimated confidence of the acquired user characteristics with respect to the user's own perceived needs. We also examine how fast we acquire (changing) user characteristics and make them available to the personalization engine of the faceted browser.

Personalization – the impact of personalization on overall user experience. We examined how much does personalization improve user experience (e.g., time, click count, result quality, user satisfaction).

7.1 Personalized Faceted Browser Overview

Faceted browsers employ faceted navigation, which is based on faceted classification – an orthogonal multidimensional classification of information artefacts, which was originally developed in library sciences [37]. Its basic principle lies in the use of facets, describing individual properties of instances in an information space, to specify the desired properties of instances in the visible information space. The final search query combines restrictions from individual facets via the logical AND function resulting in an unordered list of instances that satisfy all specified restrictions.

Faceted browsers allow users to perform view-based search by providing graphical user interfaces (GUI) that enable users to interactively select specific subspaces of the original information space and ultimately view the details of its instances – information artefacts (e.g., documents, job offers or publications). This is done by visually constructing (semantic) search queries via navigation by defining one or more restrictions in the set of available facets of inventing and writing keywords.

In practice, faceted browsers can be effectively used for faceted browsing of an information space without a specific goal, e.g. to get an overview of what information

is available, or for faceted search if the specific properties of the desired search results are known, e.g. if journal papers on adaptive hypermedia and the Semantic Web not older than 3 years should be returned.

The GUI of our personalized faceted browser builds upon the generic faceted browser layout and functionality (see Figure 8). Furthermore, Factic extends "classical" faceted browsers with personalization support based on an automatically acquired user model [32].

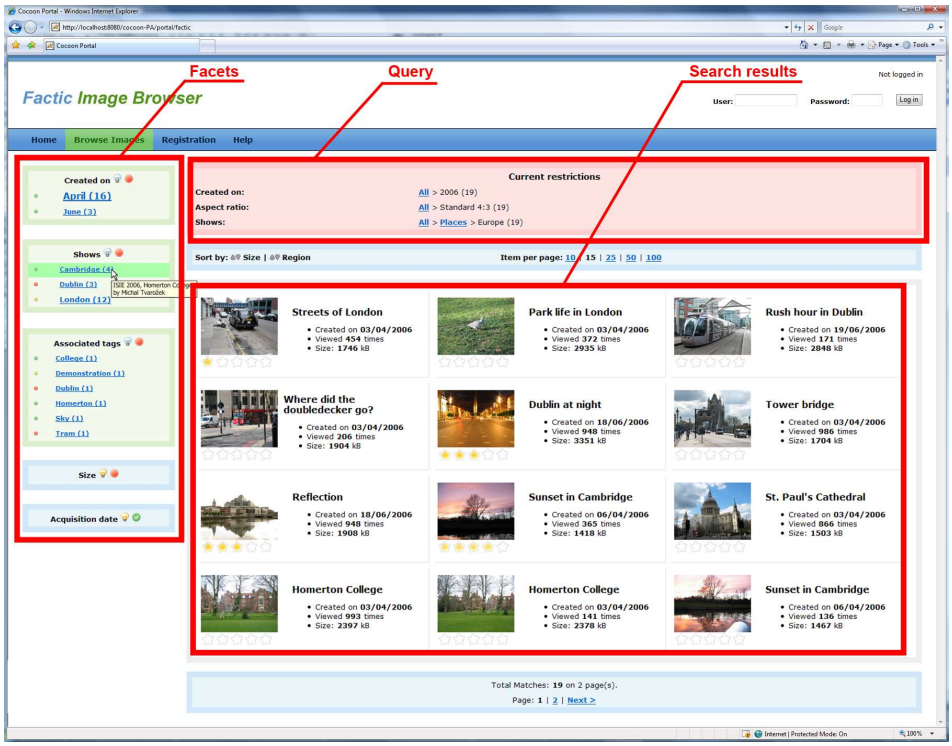

Fig. 8. Example of our personalized faceted browser Factic in the digital image domain, employing the general faceted browser layout (facets on the left, query at the top, search results in the centre, optional manual search result customization, e.g. sorting, above search results).

Facet personalization adapts the set of available facets and restrictions based on the in-session user behavior and based on more long term user characteristics stored in the user model also considering the characteristics of other users. Individual facets are hidden or disabled if they seem less relevant to the current user task, or reordered based on their relevance. Restrictions are annotated with additional information (e.g., instance count or relevance based on the user model) and/or

recommended (e.g., shown with different background colors). Consequently, we evaluate personalization with respect to these adaptation options:

Active facet count determines the number of simultaneously active facets (i.e., with visible/annotated contents) thus affecting the complexity of the GUI and the time required to process user actions (i.e., facet and restriction annotation and processing time).

Active facet selection identifies active facets, whose contents are shown, and inactive facets, whose contents are hidden yet available on demand.

Facet recommendation simplifies access to facets by showing often used and/or relevant facets at the top of the facet list.

Restriction recommendation provides users with navigational shortcuts to relevant restrictions by showing additional facet restrictions with different background color.

7.2 Rules Used for User Model Inferencing

Based on the observation of user behaviour in the enhanced faceted browser Factic, we devised the following set of user modeling rules:

R01 – After login restriction. This rule defines a pattern which is matched if the first user action after login is a restriction selection, i.e. the selection of a value in a facet. The pattern allows for the refinement of the restriction. If, for example, the user chose *hasDutyLocation–Europe*, then *Slovakia* and then *Bratislava*, all events are mapped to the pattern.

The pattern is associated with two changes: the first change indicates the relevance of the property *hasDutyLocation* represented by a facet. This is stored as an *AttributePreference* characteristic in the user model. The second change focuses on the last chosen value (*Bratislava* from our example) and stores it in the user model using *RuleCharacteristic*.

R02 – Facet enable and restriction selected. This rule defines a pattern which expects the user to enable a previously disabled facet (so the user would see facet values) and to choose a restriction within this facet. The pattern allows for refinement of the restriction as described for rule R01. Similarly to R01, it stores the property associated with the facet in the *AttributePreference* characteristic while the chosen value is stored in the *RuleCharacteristic*.

We also defined a similar rule which distinguishes this behavior if it was the first action of the user after login, which leads to higher relevance of the revealed characteristic.

R03 – Result browsing. This rule defines a pattern representing user browsing in the current result set, which signals a deep interest in the currently selected restrictions. The pattern is matched when any event from *ShowDetails-*

ShowOverview, *PageNext*, *PagePrevious* or *PageSelect* occurs four times in a row without changing the currently selected restrictions.

All currently selected restriction values are stored in a *RuleCharacteristic* in the user model.

R04 – Refining restriction. This rule defines a pattern corresponding to at least three restriction refinements on one property in a row, yet also allows for further refinement. As a consequence, the restricted property is stored in an *AttributePreference* characteristic while the last chosen restriction is stored in a *RuleCharacteristic* in the user model.

R05 – Disabling a Facet. This rule defines a simple pattern consisting of only one event: *FacetDisable*. If such an event occurs in the log, relevance of the property represented by the facet is lowered in an *AttributePreference* characteristic.

R06 – Selecting Sorting Order. This rule defines a simple pattern consisting of only one event: *Selection of sorting order* (e.g., order by Salary). If such an event occurs in the log, relevance of the chosen property is raised via an *AttributePreference* characteristic.

7.3 Users and Scenarios

We performed several experiments in the job offers domain with participants from our research lab. Each participant was asked to prepare a job search profile describing his or her ideal job offer. Thereafter, all participants performed two navigation sessions in our faceted browser Factic, impersonating the created search profile.

Their goal for the first session was to find the information subspace containing suitable job offers matching their predefined profile and browse some of them, or alternatively to find out whether there are no job offers available that would match the participant's search profile.

The second session simulated a repeated search for job offers matching the same profile (e.g., newly added job offers). Participants were supposed to reach the same goal though not necessarily by the same means – sequences of actions.

We evaluated the total user effort (number of clicks) that was necessary to complete the scenario. Our hypothesis was that the required effort would be lower in the second session due to facet recommendation. As all users were familiarized with the system and the faceted browsing paradigm prior to the experiment, we did not expect the results of the second session to be significantly biased by previous user experience in the first session.

Furthermore, each participant was asked to rate the level of conformance of his or her current user model with the predefined search profile after each session. We expected higher ratings after the second session, as the model would be more refined after being updated several times.

7.4 Results

Table 1 provides an overview of the most interesting experimental results on selected participants, which show how the number of clicks affects the user model. For each user we show the model state (count of characteristics) after the first and second session along with count of user model updates, which were performed during the session.

User	*Session*	*#Clicks*	*#Characteristics*	*Rating*	*#Model updates/session*
A	1	13	5	2	5
	2	8	5	2	4
B	1	27	8	2	8
	2	20	8	2	4
C	1	42	9	2	10
	2	22	9	2	6
D	1	5	2	0	5
	2	12	4	1	4

Table 1. Examples of collected data

We observed that most users used predominantly two facets: *hasDutyLocation* and *offersPosition* denoting the location and type of the job offer respectively. Consequently, the evaluation of these two facets rose quickly to high levels of both relevance and confidence in their user models.

This was apparently due to the fact that for our dataset, these two facets could often narrow down the information space enough to the users' predefined search profiles, optionally using sorting based on the salary attribute. This also indicates that most users view job search as a two dimensional task involving the position and location – what and where, which explains why only few new characteristics were discovered in the second user session.

We also observed lower click count in the second session compared to the first one indicating successful adaptation via navigational shortcuts based on the user model acquired during the first session (which was still updated in the second session).

Moreover, the initially identified characteristics were reinforced in the second session while only few new ones were found indicating consistent user behaviour (i.e., good estimation of characteristics on the same search goals). These findings were confirmed since users rated their estimated profiles after each session with predominantly positive ratings implying successful user characteristics acquisition.

Lastly, we point out that in the performed experiments, significantly higher numbers of clicks did not result in significantly more detected user characteristics (i.e., users behaved consistently with their profiles). The acquired characteristics, however, tended to have higher confidence and relevance estimates if more clicks were evaluated.

While most users chose to follow the recommended navigational shortcuts in the second session (which led to reinforcement of already present characteristics), users who were not successful at finding their desired job offers in the first session changed their search behaviour and tried to find results by different means (thus the count of required clicks did not decrease). This behaviour led to the discovery of different characteristics.

The last row of Table 1 shows such a situation (5 clicks in the first session against 12 in the second session). A user wanted to find a programmer position in Central Europe, and during the first session he quickly discovered that there were only few offers available after selecting Central Europe as the location. User model inference discovered only two characteristics: Central Europe as the preferred duty location along with duty location attribute preference, which led to lower rating of the user model by the user. However, in the second session, he chose instead to search for job offers using restrictions on *offersPosition*, resulting in more newly discovered characteristics.

A somewhat similar situation could occur if a user tried to look for a completely different type of job offer. As our adaptation engine does not force the user to "stay within" his or her already discovered characteristics, the system would simply learn something new with each user interaction. Since every characteristic has additional metadata (confidence, relevance, timestamp and count-of-updates), we can distinguish between more recent (recent timestamp) or more stable characteristics (higher count-of-updates, confidence) and old and rather accidental characteristics. This in turn is reflected in the adaptation engine which weights recent short-term user characteristics higher than older long-term characteristics (ideally) resulting in seamless adaptation to changing user preferences (e.g., a new type of desired job offer).

8 CONCLUSIONS

We presented a novel method of automated user modeling for adaptive semantic web-based systems, based on comprehensive logs of user actions with semantics and a rule formalism designed to support potentially elaborate navigational patterns.

We evaluated our approach with promising results in three application domains – job offers (presented in this paper), publications and digital images. The tool *LogAnalyzer*, which realizes the proposed method, served as the user modeling agent for an adaptive presentation front-end – the enhanced faceted browser Factic. The evaluation shows that the approach is able to reveal relevant user characteristics which form a solid basis for personalization.

During evaluation, we also encountered several bottlenecks that at present seriously limit widespread deployment of applications for the Semantic Web, mainly the general immaturity of ontological repositories in terms of their processing speed (slow processing of ontological queries).

The key advantages of our approach are:

Complete separation of the user model inference process from the presentation layer by means of semantic user interaction evidence logging based on an *Event ontology*. Logs contain detailed information about each occurred event including its context (i.e., the description of displayed user interface items).

Log processing flexibility – all domain-dependent logic required to transform logged events into the resulting user model is stored in externally supplied rules. This, along with *ontology based user model representation* allows for extremely straightforward execution and maintenance of the user modeling process. If for example a new presentation tool is introduced into the system or an existing one is changed, or if the user model representation changes, the functionality of the user modeling process does not break with the only requirement being up-to-date rule definitions.

The pattern parts of our user model inference rules can be defined manually (e.g., by domain experts) or identified by observing real usage of the used presentation tools and applying various web usage data mining techniques.

We see several directions for future work. Enhancements of our rule mechanism such as the introduction of dynamic parameters (e.g., count-of-occurrence) that with values from the current user context, or meta-rules that could drive the changes in the working set of rules for individual users appear interesting. Meta-rules could prescribe the maximum number of uses of a particular rule, and could also influence the parameters of the rule so a particular pattern would become "harder" to detect if it was already detected once, etc.

Another enhancement is the incorporation of feedback processing. Implicit feedback (replacement of an explicit user rating of displayed content) is in its nature a sequence of specific events, thus a pattern, which can be defined using our rule mechanism and subsequently identified during log processing. Once the ratings of various items are available, we can estimate the user's preferences by comparing similarly/differently rated domain instances (e.g., using various similarity measures for ontology instances comparison [2] or reguler expression patterns for detecting similarity [21]). If two different instances were rated similarly, the small fraction, which is common for both of them can be considered as relevant for the user and vice versa.

Acknowledgement

This work was partially supported by the Cultural and Educational Grant Agency of the Slovak Republic, grant No. KEGA 3/5187/07, the State programme of research and development "Establishing of Information Society" under the contract No. 1025/04 and the Scientific Grant Agency of the Slovak Republic, grant No. VG1/0508/09.

REFERENCES

[1] Andrejko, A.—Barla, M.—Bieliková, M.:. Ontology-based User Modeling for Web-based Information Systems. In Information Systems Development (ISD 2006), Budapest, Hungary, Springer, 2006.

[2] Andrejko, A.—Bieliková, M.: Estimating Similarity of the Ontological Concepts Instances for the Adaptive Applications Based on Semantic Web. In V. Snášel (Ed.), Znalosti 2008, pp. 30–41, 2008 (in Slovak).

[3] Bieliková, M.—Matušíková, K.: Social Navigation for Semantic Web Applications Using Space Maps. Computing and Informatics, Vol. 26 2007, No. 6, pp. 281–299.

[4] Broder, A.: A Taxonomy of Web Search. SIGIR Forum, Vol. 36, 2002, No. 2, pp. 3–10.

[5] Brusilovsky, P.: Methods and Techniques of Adaptive Hypermedia. User Model. User-Adapt. Interact., Vol. 6, 1996, No.2–3, pp. 87–129.

[6] Brusilovsky, P.: Adaptive Hypermedia. User Model. User-Adapt. Interact., Vol. 11, 2001, No. 1–2, pp. 87–110.

[7] Brusilovsky, P.—Kobsa, A.—Nejdl, W. Eds.: The Adaptive Web, Methods and Strategies of Web Personalization. LNCS 4321, Springer 2007.

[8] Brusilovsky, P.—Sosnovsky, S.—O. Shcherbinina, O.: User Modeling in a Distributed E-Learning Architecture. In L. Ardissono, P. Brna, A. Mitrovic (Eds.), User Modeling 2005, LNCS 3538, Edinburgh, Scotland, UK, pp. 387–391, Springer 2005.

[9] Bunt, A.—Conati, C.: Probabilistic Student Modelling to Improve Exploratory Behaviour. User Model. User-Adapt. Interact., Vol. 13, 2003, No. 3, pp. 269–309.

[10] Chen, Z.—Fu, A.—Tong, F.: Optimal Algorithms for Finding User Access Sessions from Very Large Web Logs. World Wide Web, Vol. 6, 2003, No. 3, pp. 259–279.

[11] Denaux, R.—Dimitrova, V.—Aroyo, L.: Integrating Open User Modeling and Learning Content Management for the Semantic Web. In L. Ardissono, P. Brna, Mitrovic (Eds.), User Modeling 2005, LNCS 3538, Edinburgh, Scotland, UK, pp. 9–18, Springer, 2005.

[12] Eckhardt, A.—Horváth, T.—Vojtáš, P.: Learning Different User Profile Annotated Rules for Fuzzy Preference Top-k Querying. In H. Prade and V. S. Subrahmanian (Eds.), SUM 2007, LNCS 4772, pp. 116–130, Springer 2007.

[13] Eirinaki, M.—Vazirgiannis, M.: Web Mining for Web Personalization. ACM Trans. Internet Techn., Vol. 3, 2003, No. 1, pp. 1–27.

[14] Etgen, M.—Cantor, J.:. What Does Getting WET (Web Event-Logging Tool) Mean for Web Usability? In 5th Conference on Human Factors & The Web, Gaithersburg, Maryland, USA, 1999.

[15] Fenstermacher, K.—Ginsburg, M.: Mining Client-Side Activity for Personalization. In WECWIS, pp. 205–212, 2002.

[16] Gurský, P.—Pázman, R.—Vojtáš, P.: Ontea: On Supporting Wide Range of Attribute Types for Top-k Search. Computing and Informatics, Vol. 28, 2009, No. 4, pp. 483–513.

[17] Heckmann, D. et al.: GUMO – The General User Model Ontology. In L. Ardissono, P. Brna, A. Mitrovic (Eds.): User Modeling 2005, LNCS 3538, Edinburgh, Scotland, UK, pp. 428–432, Springer, 2005.

[18] Jameson, A.: Numerical Uncertainty Management in User and Student Modeling: An Overview of Systems and Issues. User Model. User-Adapt. Interact., Vol. 5, 1995, No. 4, pp. 193–251.

[19] Kay, J.: The um Toolkit for Cooperative User Modeling. User Model. User-Adapt. Interact., Vol. 4, 1995, No. 3.

[20] Kay, J.—Kummerfeld, B.—Lauder: Personis: A Server for User Models. In P. de Bra, P. Brusilovsky, R. Conejo (Eds.), Adaptive Hypermedia and Adaptive Web-Based Systems, AH 2002, LNCS 2347, Malaga, Spain, pp. 203–212, Springer 2002.

[21] Laclavík, M.—Šeleng, M.—Ciglan, M.—Hluchý, L.: Ontea: Platform for Pattern Based Automated Semantic Annotation. Computing and Informatics, Vol. 28, 2009, No. 4, pp. 555–579.

[22] Levene, M.—Wheeldon, R.: Navigating the World Wide Web. In M. Levene, A. Poulovassilis (Eds.), Web Dynamics – Adapting to Change in Content, Size, Topology and Use, pp. 117–152, Springer 2004.

[23] Lu, H.—Luo, Q.—Shun, Y.: Extending a Web Browser with Client-Side Mining. In X. Zhou, Y. Zhang, Orlowska (Eds.), Web Technologies and Applications, APWeb 2003, LNCS 2642, pp. 166–177, Springer 2003.

[24] Machová, K.—Bednár, P.—Mach, M.: Various Approaches to Web Information Processing. In Computing and Informatics, Vol. 26, 2007, No. 6, pp. 301–327.

[25] Návrat, P.—Bartoš, P.—Bieliková, M.—Hluchý, L.—Vojtáš, P. (Eds.): Tools for Acquisition, Organization and Presenting of Information and Knowledge, Research Project Workshop, Bystrá Dolina, Low Tatras, Slovakia, 2006.

[26] Návrat, P.—Taraba, T.: Context Search. In Y. Li, V. V. Raghavan, A. Broder, H. Ho (Eds.), 2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology (Workshops), Silicon Valley, USA, pp. 99–102. IEEE Computer Society, 2007.

[27] Ono, Ch.—Kurokawa, M.—Motomura, Y.—Asoh, H.: A Context-Aware Movie Preference Model Using a Bayesian Network for Recommendation and Promotion. In Rule-based User Characteristics Acquisition from Logs with Semantics, C. Conati, K. F. McCoy, G. Paliouras (Eds.), User Modeling 2007, LNCS 4511, pp. 247–257, Springer 2007.

[28] Paramythis, A.—Weibelzahl, S.: A Decomposition Model for the Layered Evaluation of Interactive Adaptive Systems. In L. Ardissono, P. Brna, A. Mitrovic (Eds.), UM 2005, LNCS 3538, pp. 438–442, Edinburgh, Scotland, UK, Springer 2005.

[29] Shadbolt, N.—Berners-Lee, T.—Hall, W.: The Semantic Web Revisited. IEEE Intelligent Systems, Vol. 21, 2006, No. 3, pp. 96–101.

[30] Šimún, M.—Andrejko, A.—Bieliková, M.: Ontology-Based Models for Personalized E-Leaning Environment. In ICETA 2007 – 5th Int. Conference on Emerging E-Learning Technologies and Applications, pp. 335–340, 2007.

[31] Tvarožek, M.—Barla, M.—Bieliková, M.: Personalized Presentation in Web-Based Information Systems. In van Leeuwen, J. et al. (Eds.), SOFSEM'07, LNCS 4362, pp. 796–807, Springer 2007.

[32] Tvarožek, M.—Bieliková, M.: Personalized Faceted Navigation for Multimedia Collections. In SMAP'07: Proc. of the 2nd Int. Workshop on Semantic Media Adaptation and Personalization, pp. 104–109, IEEE CS, 2007.

[33] Tvarožek, M.—Bieliková, M.: Personalized Faceted Navigation in the Semantic Web. In L. Baresi, P. Fraternali, G.-J. Houben (Eds.), ICWE 2007, LNCS 4607, pp. 511–515, Springer 2007.

[34] Wang, P.—Xu, B.: Debugging Ontology Mappings: A Static Approach. Computing and Informatics, Vol. 27, 2008, No. 1, pp. 21–36.

[35] Webb, G. I.—Pazzani, M. J.—Billsus, D.: Machine Learning for User Modeling. User Model. User-Adapt. Interact., Vol. 11, 2001, No. 1–2, pp. 19–29.

[36] Wong, A. K. Y.—Yip, F.—Ray, P.—Paramesh, N.: Towards Semantic Interoperability for IT Governance: An Ontological Approach. In Computing and Informatics, Vol. 27, 2008, No. 1, pp. 131–155.

[37] Wynar, B. S.—Taylor, A. G.: Introduction to Cataloging and Classification. Libraries Unlimited Inc., 1992.

[38] Yudelson, M.—Brusilovsky, P.—Zadorozhny, V.:. A User Modeling Server for Contemporary Adaptive Hypermedia: An Evaluation of the Push Approach to Evidence Propagatation. In C. Conati, K. McCoy, G. Paliouras (Eds.), User Modeling 2007, LNAI 4511, pp. 27–36, Corfu, Greece, Springer 2007.

[39] Zhu, T.—Greiner, R.—Häubl, G.: Learning a Model of a Web User's Interests. In P. Brusilovsky, A. T. Corbett, F. de Rosis (Eds.), User Modeling 2003, LNCS 2702, pp. 65–75, Springer 2003.

Michal Barla received his Master degree in 2007 from the Slovak University of Technology in Bratislava. Currently he is a Ph. D. student at the Faculty of Informatics and Information Technologies of the same university. His research interests include web-based systems, with focus on the user modelling in adaptive web-based systems.

Michal Tvarožek received his Master degree in 2007 from the Slovak University of Technology in Bratislava. Currently he is a Ph. D. student at the Faculty of Informatics and Information Technologies of the same university. He is doing research in personalized navigation, information retrieval and user interfaces for adaptive Semantic Web applications.

Mária Bieliková received her Master degree (with summa cum laude) in 1989 and her Ph. D. degree in 1995, both from the Slovak University of Technology in Bratislava. Since 2005, she has been a Full Professor, presently at the Institute of Informatics and Software Engineering at the Slovak University of Technology. Her research interests include software knowledge engineering and web information systems, especially adaptive web-based systems including user modelling.