Estimation of User Characteristics using Rule-based Analysis of User Logs

Michal Barla and Mária Bieliková

Institute of Informatics and Software Engineering,
Faculty of Informatics and Information Technologies,
Ilkovičova 3, Bratislava, Slovakia,
{barla,bielik}@fiit.stuba.sk

Abstract. Personalization is becoming more important if we want to preserve the effectiveness of work with information, providing larger and larger amount of the content. Systems are becoming adaptive by taking into account characteristics of their users. In this paper we describe our work in the field of automatized user characteristics acquisition based on capturing user behavior and its successive rule-based analysis. We stress on re-usability aspect by introducing rules used for the analysis of logs. Resulting user characteristics are stored in an ontology-based user model.

1 Introduction

Many teachers around the world are using information technologies to support the learning process. A lot of tools and frameworks exist, which allow for creation and publication of study materials for students online. There exist specifications like LOM or IMS-LD to support re-usability and interoperability of learning objects. However, learning objects are really re-usable and of great value for students only if they are integrated in such a way that their presentation reflects the needs and skills of the students – individual users, i.e., it is adaptive.

E-learning is an ideal domain for adaptation since every student might prefer different style of learning (e.g., top-down, bottom-up), might have different background and experience in a topic of e-course. If an educational system is aware of these user characteristics (represented explicitly in a user model) it can noticeably improve user's experience with the system and ease the learning process.

In this paper, we present a rule-based approach to analysis of logs of user activity (user modeling based on observing user behavior). We focus on aspect of re-usability and interoperability of the solution. We explicitly defined logs of user activity and devised a generic method of its processing according to a given set of rules. The method produces instances of user characteristics in an ontology-based user model. As a result, it is easy to incorporate a user modeling feature into existing systems and thus enable personalization.

The paper is structured as follows. In section 2 we describe current trends and problems in the field of user characteristics acquisition. Next in section 3 we introduce our user model. Section 4 contains description of proposed rule-based adaptation knowledge representation. In section 5 we describe a process of user characteristics discovery using proposed rules. We evaluate our work and describe future work in section 6. Finally, we give conclusions.

2 Related Works

On the top-level, user modeling consists of two stages: data collection and data processing (analysis) [1]. It is important to recognize that the first stage has a substantial impact on possibilities of the second stage.

If we consider automatized approaches to data collection, it is popular to use logs produced by a web server as a basis for the analysis. *Web Usage Mining* [2] is a special branch of data mining techniques applied on the web server logs (clustering, classification, association rule and sequential patterns mining). These techniques are based on a social aspects, where the actual user session is mapped to some patterns of a group of users and as a result

they can not be used directly to acquire characteristics of an individual. Still, techniques of *Web Usage Mining* can be used effectively to support students in learning process. A good example can be found in [3].

In the case where the web server log is produced by some specialized tools, it is crucial to transform the log into usable form [4]. Even if it is done, the log lacks semantics of majority of user–system interactions (records are based on low-level HTTP protocol). The user model is then often realized as a simple statistical model expressing whether (and how many times) a user visited some page. Such a model is very system-dependent and is of no use to other systems. What we need is to have a model filled by characteristics rather than statistics.

Because of the mentioned problems, many researchers have proposed separate logging subsystem (often on a client side of the system) which replaces web server log or is used together with it [5]. However, in a majority of current approaches we are missing explicitness of the produced log. Semantics of acquired logs is used implicitly in the part of log analysis which results in tightly coupled modules with limited re-usability. In [6] authors refer to a *Log Ontology* but do not provide more details of it.

A re-usable and interoperable user modeling solutions already exist. For instance the *Duine* toolkit [7] allows any information system to incorporate recommendation services. Disadvantage is that the user model produced by *Duine* is closed, stored in relational database and thus not enough shareable. We found similar problem also in *BGP-MS* system [8].

3 User Model

Our user model consists of two parts: logs of user actions and ontology-based part used for actual adaptation. Because collected logs represent huge amount of data, we are taking advantage of maturity of existing relational databases for the storage. Ontological representation of user characteristics allows easy interconnection of several models, sharing and re-usability of constructed user model.

3.1 Logs of User Actions

Because we consider logs produced by a web server as not sufficient for the estimation of characteristics of an individual user, we designed and developed a logging sub-system [9] which is responsible for creation of detailed logs of a user activity. Basic requirement is to have *self-contained* records, so we would not need any other additional data to be able to process and interpret them. The semantics of the action should be expressed in the log itself. Therefore we log *event* (as a result of the user action) together with all its *attributes* as well as with a description of current *display state* (description of *items* and their *attributes* which were displayed when the action was performed).

Our next requirement is to have a flexible enough representation of the logs which allows for uniform storing of records of user interactions from several types of user interfaces. Simultaneously, we required a representation which can deal flexible with changes of the adaptive application and its presentation layers.

As a result, we designed a generic data model, whose flexibility was achieved by using a two layer model:

- meta-layer, which prescribes associations between types of entities. We defined types of known events, types of their attributes;
- operating layer, which contains specific run-time values.

3.2 Ontology-based model of user characteristics

We use ontology as a mean for representation of user characteristics. It is divided into domain independent and domain dependent parts [10]. The domain independent part defines characteristics like age or sex as well as structure of a characteristic. We combine several ontologies where the domain dependent part is always connected to the appropriate domain model. In this paper we consider representing the ontology by RDF/OWL formalisms.

In our model used for e-learning domain, we currently use two types of characteristics (*CoursePropertyPreference* which informs about a relation of the user to the specific domain properties and *CourseSpecificUserCharacteristic* which informs also about *values* of properties), derived from a common super-class (*gu:UserCharacteristic*), which gives common attributes to all characteristics (see Figure 1). Each characteristic has a time stamp and a source. For the purpose of adaptation we define a user goal, and the characteristics are somehow relevant to the user in achieving this goal. Because our method produces estimations of user characteristics, we store a level of confidence for each characteristic. Confidence informs about quality of the estimation [1].

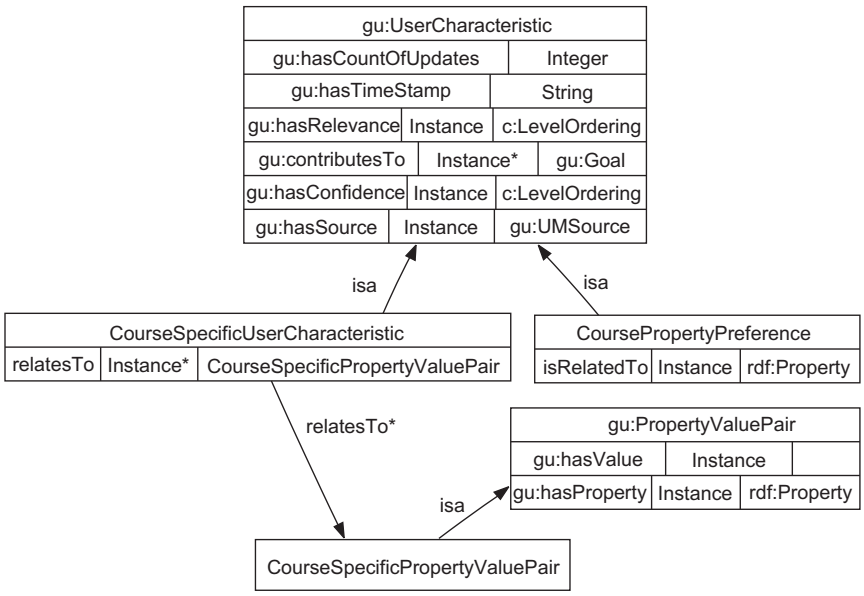

Fig. 1. Representation of a user characteristic in used user model.

4 Knowledge on User Characteristics Acquisition

Knowledge on user characteristics acquisition is represented by rules. Each rule consists of two parts: a pattern and (at least one) consequence (see Figure 2). Knowledge representation formalism is crucial for devised method of user characteristics acquisition. The rules must be able to store various types of possibilities which can occur during user-system interaction.

4.1 Pattern

Pre-defined patterns detected in the user activity log form the base of data analysis. A pattern is on the top-level defined as a sequence of event types and other sub-sequences (see Figure 2). A pattern is detected when an occurrence of the top-level sequence is found. Finding occurrence of the sequence means that we are able to map prescribed events to specific events found in the log of the user activity.

Sequence. A sequence can be of two different types:

- *AllRequired* - basic sequence type which is detected in the log of user activity if we detect occurrence of all its events and subsequences (equivalent to logical operation AND);

[1] User models constructed using the same approach can be found at `nazou.fiit.stuba.sk` and `mapekus.fiit.stuba.sk`

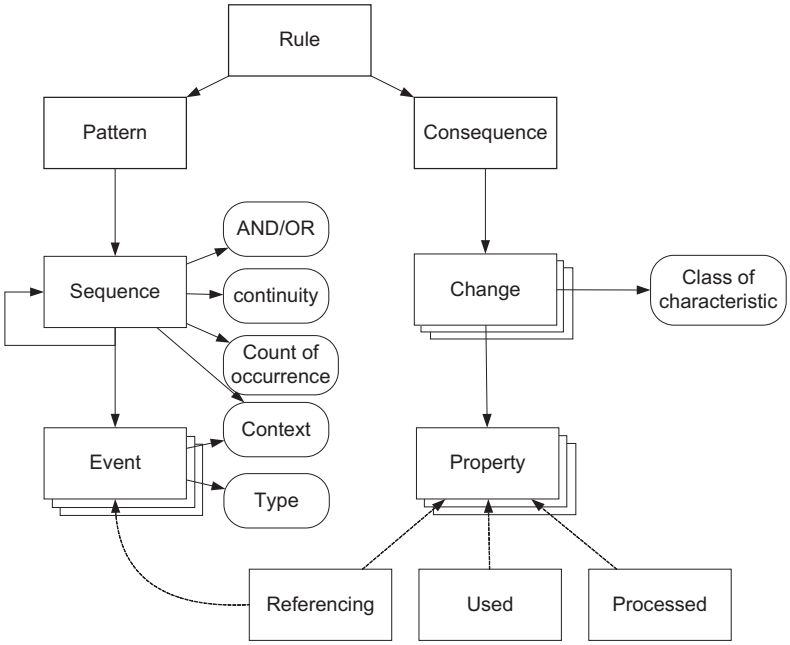

Fig. 2. Structure of rules used for estimation of characteristics.

– *OneRequired* - to detect this kind of a sequence, it is sufficient to detect occurrence of one of its events or subsequences (equivalent to logical operation OR).

Further, we divide sequences into *continuous* and *discrete*. A continuous sequence demands that all of its events must succeed directly one after another. Events of a discrete sequence can be separated by any count of other events and sequences. A sequence can thus span through multiple user sessions.

We define following attributes of a sequence:

– *Count-of-occurrence* - prescribes the required count of the sequence repetition in a pattern. The execution engine will continue to process the next sequence only if this count was achieved. This attribute can have a special value (negative number) to define a sequence as optional.
– *Context* - optional attribute which defines the restrictions on events being mapped to the current sequence. For example, a context restriction can define types of displayed items attributes which must stay unchanged for all events mapped to the sequence.

Event. An event represents an elementary part of a pattern. During the pattern detection, we map the events from log of user activity to events prescribed by the pattern. Each event has its type which corresponds to the known event type from meta-level of our user actions model. Each event can have a weight. Weight can be determined by considering various factors. We can use information such as time to next event to compute the weight or use a predefined one for a specific type of event.

Similarly to the sequence, an event can also have contextual restrictions. The context of an event defines restrictions solely on attributes of the event while the context of a sequence deals with display state.

Each event context condition can be of the following types:

– *SameAsPrevious* a value of some defined event attribute must be the same as in previous event of a sequence;
– *DifferentThanPrevious* a value of some defined event attribute must be different from the one in previous event of a sequence;

– *MinValueOfWeight* - this contextual condition requires a 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 not fulfilling defined contextual restriction (user did not have time to see the page with detail). Therefore, we will not map this event into the sequence.

Figure 3 illustrates an example of the pattern representing *"result browsing"*. We can consider for example a repository of available e-learning courses on various topics and a user which is interested in some topic and looks for relevant study materials. When the user selects some restrictions on the information space, a set of results which fulfills selected restrictions is returned and the user can browse them to find out more details.

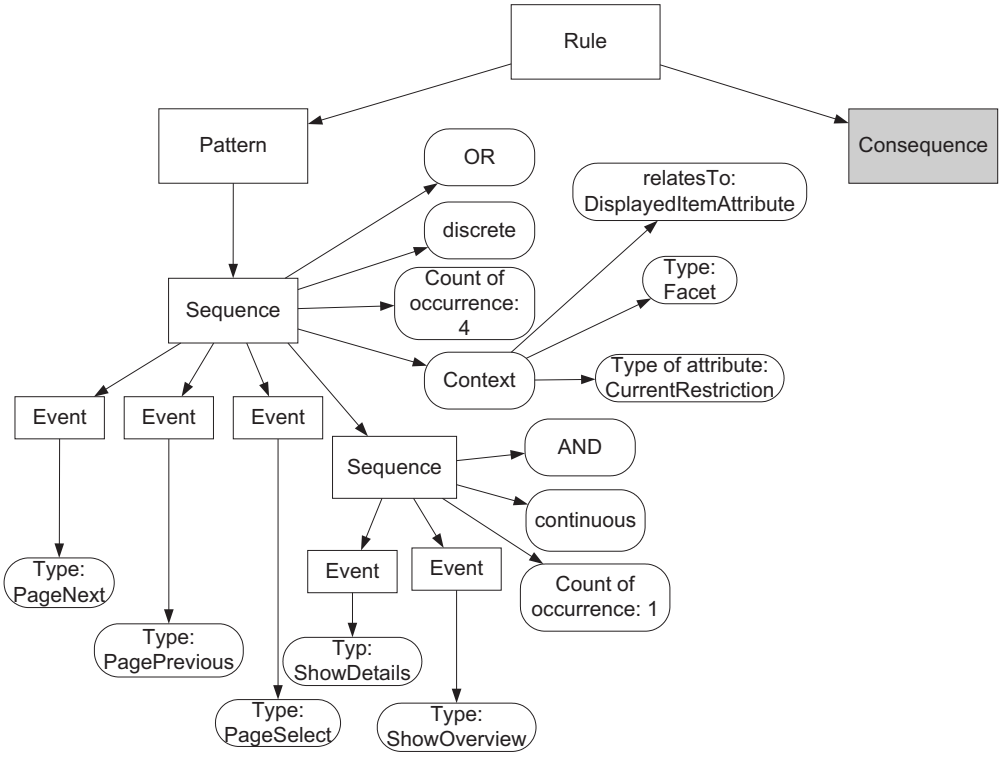

Fig. 3. Example of a pattern part of the rule *"results browsing"*.

A pattern is on the top level formed by a discrete sequence of type OneRequired. The sequence has to be found four times in the log of user action for the pattern to become detected. The sequence has a contextual restriction which refers to an attribute of displayed item. All mapped events have to be connected to such a display state, which have for all displayed items of type "facet" constant value of actually chosen restriction. In other words, the user is not changing currently chosen restrictions of the information space and is only browsing in the list of results. Events can be of type *PageNext*, *PagePrevious*, *PageSelect*, *ShowDetails* and *ShowOverview*. Former three types of events represent navigation through individual pages of results while latter two events, joined in a continuous subsequence, represent display of details and navigation back to the list of results.

4.2 Consequence

A consequence determines what and how should be changed in the user model in the case when the instance of a pattern is detected. The consequence consists of unlimited count of

changes of user characteristics (see Figure 2). Each change has an attribute *class*, which determines the type of the user characteristic being changed (a class where the instance of a characteristic belongs to) and several *property* attributes prescribing changes of the object and literal properties of an instance being changed. Each property is defined by its URI as defined in a T-box of the used ontology.

The change can have three different types of properties:

- *Used property* - a rule defines directly the value which should be used for a property.
- *Processed property* - contains an instruction how to compute value of given property. It is used for numerical data-type properties such as confidence or relevance. Basic information is whether the existing value will be increased or decreased. Processed property defines an increment/decrement for one step and boundaries of interval, where the actual rule can still change the value. Processed property also defines a strategy how to change the characteristic (e.g., progressively or uniform).
- *Referencing property* - refers to an event from the pattern part of the rule. A value of property is actually a value of an attribute of the referenced event.

On the Figure 4 is displayed an example of a consequence part of the rule "result browsing". The consequence changes instances of the *CourseSpecificCharacteristic*.

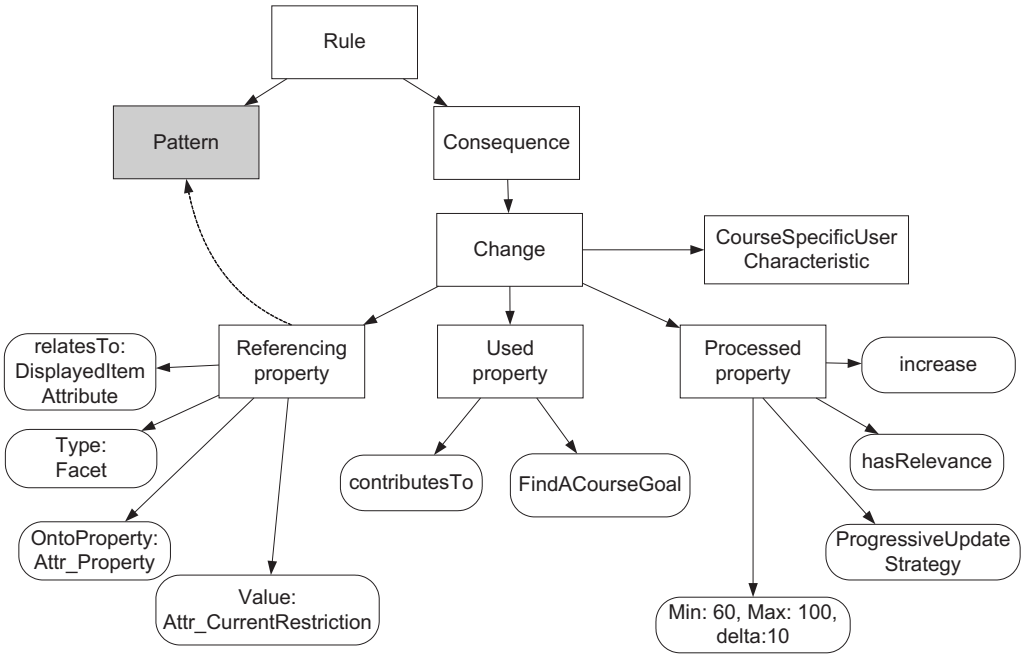

Fig. 4. Example of a consequence part of the rule *"results browsing"*.

5 User Characteristics Acquisition Process

We have proposed a method for user log analysis based on adaptation knowledge represented by above described rule-based mechanism. The analysis process is depicted on Figure 5. It consists of following steps: pre-processing of the entry, pattern detection, and the user model update. If a pattern of implicit feedback was detected, we include feedback evaluation (acquirement of concept rating) and a comparison of rated concepts into the process.

5.1 Data Pre-processing

Thanks to the well defined semantics of data already in the data collection stage and used representation of collected data we do not need such a complex data-preprocessing as we

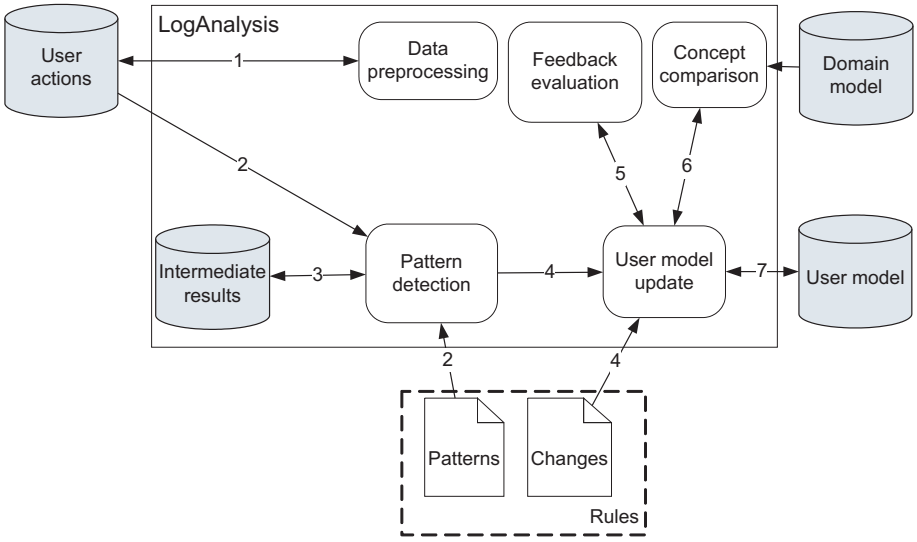

Fig. 5. Overview of a data analysis process.

can see in other solutions which work with logs produced by a web server [4]. This stage consists of mainly filtering consecutive events of the same type and context, which is often a result of user's repeated click on the same item (e.g., because of slow response times of the system). We also assign weights as described in 4.1 to individual events at this stage.

5.2 Pattern Detection

Pattern detection is a key task in the process of log analysis. Detection works similarly as it is in standard production systems, it maps an event prescribed by the rule to a specific event from the log of user actions. Events are mapped to the instances of the rule for a specific user. Each rule instance holds references to the instances of its sequences to have an evidence of reached count-of-occurrence for each sequence.

The basic idea of the algorithm is explained in the following pseudo-code:

```
DetectPattern(Event):
     find rule candidates for Event;
     for each rule in rule candidates
        find applicable rule instances(rule, event);
        for each rule instance in applicable rule instances
                    apply event on rule instance;

findRuleCandidates(Event):
     for each rule in known rules
         if type of event matches the first event of pattern part of the rule
            add rule to candidate rules;
         else if exists such rule instance of rule belonging to the current user
                    that type of expected event match type of upcoming event
             add rule to candidate rules;
     return candidateRules;

findApplicableRuleInstances(Rule, Event):
   for each ruleInstance of rule belonging to current user
       checkContextOfCurrentSequence(Event);
       checkContinuity(Event);
       checkContextOfEvent(Event);
       if all checks passed
```

```
            add ruleInstance to applicableRuleInstances;
   return applicableRuleInstances;

apply(Event, ruleInstance):
   map Event and expectedEvent of RuleInstance;
   update state of ruleInstance; //nextExpectedEvent, count-of-occurrences
   if Pattern was detected performConsequence part of the ruleInstance;
```

5.3 User Model Update

Update of a user model is driven by changes specified in the consequence part of the rule. It performs these steps for each change:

```
UMupdate():
   retrieveInstanceOfUserCharacteristic(); //which is being changed
   for each property in processed properties;
      update value according to given strategy;
   update timestamp;
   update count-of-updates;

retrieveInstanceOfUserCharacteristic():
   check value of all referencing properties;
   check value of all used properties;
   if rule does not allow for change of ‘‘foreign characteristic
      check value of source of characteristic;
   if no instance fulfills these criteria
      create a new instance;
      set all referencing and used properties;
      set source;
   return found or created instance;
```

5.4 Feedback evaluation and Concept Comparison

In case that a detected pattern represents an implicit feedback, we compute an evaluation of the concept, which is related to the feedback. This rating is an estimation of the user rating as if the user would rate the concept explicitly by choosing a level from a given scale.

There are several strategies to evaluate implicit feedback according to type of implicit feedback [11] and implicit interest indicators [12]. Transformation of an implicit feedback into numerical value of the user rating separates further processing of the feedback from its source. This allows replacing the implicit feedback by an explicit one with no impact on its processing.

Evaluation of the feedback gives us the user ratings of concepts. Our goal is to estimate user characteristics from these ratings. We aim at finding out which concept attributes and values were the reason of low (or high) ratings. This can be achieved by comparing concepts with different and similar ratings. The basic idea is that if the difference of two concepts in one attribute caused very different ratings we can infer importance of this attribute and its value. The comparison is a complex process, it needs to employ multiple strategies and approaches [13]. Another approach which use feedback evaluation to infer user characteristics is presented in [14].

6 Evaluation and Conclusions

In this paper we presented an approach to user characteristics acquisition. Our approach is based on rule-based analysis of logs of user actions. We consider the ease of incorporation of our user modeling subsystem into existing web-based systems architectures as a substantial advantage. Only requirement is to produce a log of user actions and to prepare a set of rules for log analysis which can be easily done as shown in [15]. We presented an idea of the rule mechanism which allows for creation of simple but also (if needed) more complicated

pairs of patterns and consequences. Advantage is that we can use the same mechanism for navigation patterns as well as for patterns of implicit feedback. Output of our method is the ontology-based user model filled by estimation of user characteristics. Thanks to chosen representation these characteristics can be shared among several systems and refined to better reflect reality.

To evaluate the proposed method of user log analysis we created a software tool *LogAnalyzer* which performs rule-based analysis of collected data – logs of user actions. Similarly as with design of the method we were focused on re-usability of the tool by separating it from the rest of the system by well-defined interfaces. It is implemented in Java SE 5.0 which means that it is platform independent.

LogAnalyzer uses three types of data sources:

- *logs of user actions* stored in relational database. The tool is separated from the actual implementation of RDBMS by an O/R mapper Hibernate;
- *rules* stored in a file using XML based language;
- *user model* stored as triples in RDF repository. The tool is separated from actual implementation of RDF repository by generic enough interface.

We integrated the tool in portal solutions of two different domains – job offers in a project NAZOU [16] (`nazou.fiit.stuba.sk`) and scientific publications in a project MAPEKUS [17] (`mapekus.fiit.stuba.sk`). Characteristics retrieved by the *LogAnalyzer* tool were used for presentation adaptation. In the first stage of the evaluation process we focused on finding an execution model of the user characteristic acquisition process. We let a test user to use the web application and measured execution time of analysis after each event. On the Figure 6 are depicted average values of execution time from multiple runs of the test with a set of four or six rules. As can be seen, there is a substantial difference in time when only instances of rules for individual users were updated and when also the user model stored in RDF repository was updated (a pattern prescribed by the rule was detected). Because of very time consuming call of RDF repository, we decided for asynchronous model of tool execution instead of a pure online mode.

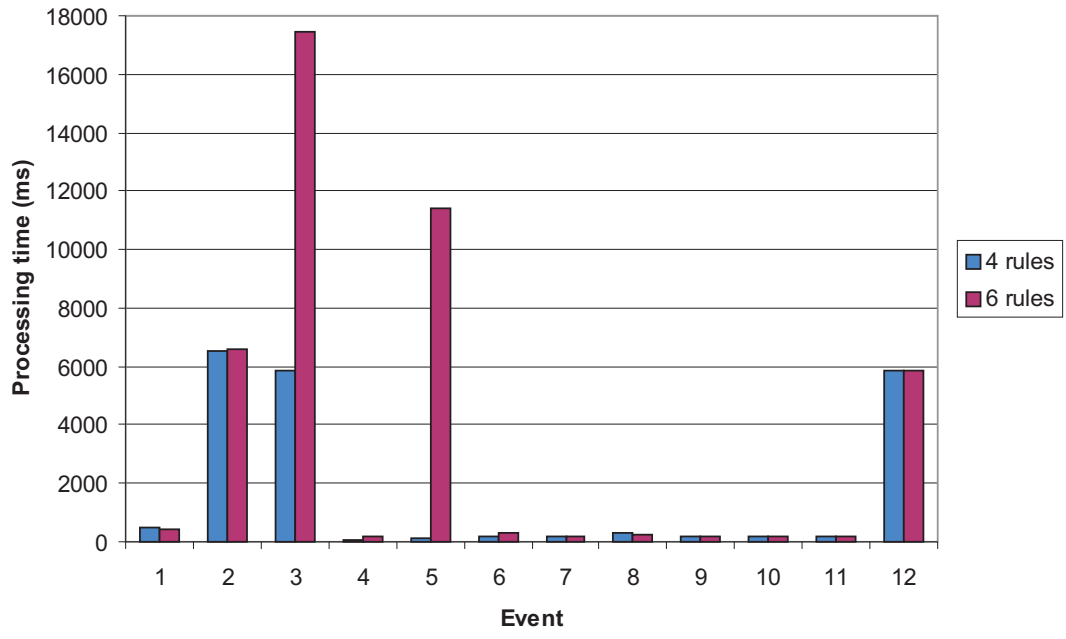

Fig. 6. Execution time of LogAnalyzer tool.

In the future work we plan to interconnect our user model with other top-level ontology-based user models [18]. Moreover we work on definition additional adaptation rules based on general heuristics related to navigation and specific heuristics related to an application domain and the evaluation of their impact for user characteristics acquisition.

Acknowledgment. This work was partially supported by the Slovak Research and Development Agency under the contract No. APVT-20-007104 and the State programme of research and development under the contract No. 1025/04.

References

1. Brusilovsky, P.: Methods and Techniques of Adaptive Hypermedia. User Model. User-Adapt. Interact. **6**(2-3) (1996) 87–129
2. Pierrakos, D., et al.: Web Usage Mining as a Tool for Personalization: A Survey. User Modeling and User-Adapted Interaction **13**(4) (2003) 311–372
3. Krištofič, A., Bieliková, M.: Improving adaptation in web-based educational hypermedia by means of knowledge discovery. In Reich, S., Tzagarakis, M., eds.: Hypertext 2005, Salzburg, Austria (2005) 184–192
4. Chen, Z., Fu, A., Tong, F.: Optimal Algorithms for Finding User Access Sessions from Very Large Web Logs. World Wide Web **6**(3) (2003) 259–279
5. Kay, J., Lum, A.: Creating User Models from Web Logs. In: Intelligent User Interfaces Workshop: Behavior-Based User Interface Customization, Funchal, Madeira, Portugal (2004) 17–20
6. Razmerita, L., Angehrn, A., Maedche, A.: Ontology-based user modeling for knowledge management systems. In Brusilovsky, P., Corbett, A., Rosis, F., eds.: User Modeling 2003. LNCS 2702, Johnstown, PA, USA, Springer (2003) 213–217
7. Setten, M.: Supporting People In Finding Information: Hybrid Recommender Systems and Goal-Based Structuring. PhD thesis, Telematica Instituut (2005)
8. Kobsa, A., Pohl, W.: The User Modeling Shell System BGP-MS. User Model. User-Adapt. Interact. **4**(2) (1995) 59–106
9. Barla, M., Tvarožek, M.: Automatic Acquisition of Comprehensive Semantic User Activity Log. In Návrat, P. et al., ed.: Tools For Acquisition, Organisation and Presenting of Information and Knowledge, Bystrá dolina, Slovakia, STU (2006) 169–174
10. 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)
11. Oard, D., Kim, J.: Implicit Feedback for Recommender Systems. In: AAAI Workshop on Recommender Systems, July 1998., Madison, Wisconsin, USA (1998)
12. Claypool, M., et al.: Implicit Interest Indicators. In: Intelligent User Interfaces, Santa Fe, New Mexico, USA (2001) 33–40
13. Andrejko, A., Barla, M., Tvarožek, M.: Comparing Ontological Concepts to Evaulate Similarity. In Návrat, P. et al., ed.: Tools For Acquisition, Organisation and Presenting of Information and Knowledge, Bystrá dolina, Slovakia, STU (2006) 71–78
14. Gurský, P., et al.: UPRE: User Preference Based Search System. In: Web Intelligence 2006 (WI'06), ACM (2006) 841–844
15. Tvarožek, M., Barla, M., Bieliková, M.: Personalized Presentation in Web-Based Information Systems. In J, van Leeuwen, et al., ed.: SOFSEM 2007. LNCS 4362, Harrachov, Czech Republic, Springer (2007) 796–807
16. Návrat, P. et al.: Acquiring, Organising and Presenting Information and Knowledge in an Environment of Heterogenous Information Sources. In Návrat, P. et al., ed.: Tools For Acquisition, Organisation and Presenting of Information and Knowledge, Bystrá dolina, Slovakia, STU (2006) 1–12
17. Bieliková, M., Návrat, P.: Modeling and acquisition, processing and employing knowledge about user activities in the Internet hyperspace. In Mikulecký, P., Dvorský, J., Krátký, M., eds.: Znalosti 2007, VSB, Ostrava, Czech Republic (2007) 368–371 (in Slovak).
18. Heckmann, D., et al.: GUMO – The General User Model Ontology. In Ardissono, L., Brna, P., Mitrovic, A., eds.: User Modeling 2005. LNCS 3538, Edinburgh, Scotland, UK, Springer (2005) 428–432