Towards an XML-based Framework for Web Usage Mining

Kai Honsel, Bernd Schneider
Information Systems I
University of Hohenheim
70593 Stuttgart, Germany
{kh,bs}@wi1.uni-hohenheim.de

Abstract: Current systems for Web Usage Mining (WUM) offer graphical and interactive ways of using the implemented methods, but do not support individual multilevel or complex analyses. Based on those systems it is neither possible to perform flexible nor regular automated evaluations based on predefined evaluation schemes for offering WUM as a service. We present an extensible framework for WUM, which describes a unified WUM process and hence supports the application of any existing WUM method. The core of our framework is a XML language called Web Usage Mining Markup Language (WUMML) that provides a vocabulary for representing and querying WUM data. WUMML uses XML standards like XPath or XQuery and supports the specification of new commands for previously not integrated WUM methods. By using WUM software that implements the WUM framework, a company is able to perform WUM analysis efficiently and is – in combination with the technical expertise – capable to professionally offer WUM as service.

1 Introduction

With the ongoing evolution of e-commerce an increasing number of people use the internet to get information, for education or entertainment purposes, and to purchase products and services. This also leads to an increasing number of companies using web sites for commercial purposes, e.g. representing themselves or offering their products and services to meet the demands of customers who visit their website. Competition requires analysis of the user's behaviour and evaluation of the web sites as well as the underlying processes in order to achieve a high service quality and provide better services via the web.
These analyses are carried out in the research area of Web Usage Mining (WUM) [CMS97, HMW02]. WUM aims at finding out insights about practices, interests and expectations of users respective customers by examining their behaviour on web sites to align the company's web site with their needs. The Web Usage Mining process formally structures these analyses. During the process data sources like web server log files are processed and so called Web Usage Mining methods are applied. The WUM methods are often based on descriptive statistics or Data Mining methods adapted to the data structures and specifics of the web.

For applying WUM practically, a set of software tools for automated or semi-automated analysis of the so-called web server log files is currently available on the market. These tools use more or less simple statistic methods to provide basic information about the use of the web site like the number of web pages accessed by users, the time spent on the web site etc. Most of the time these tools use only the web logs but no further information and are thus insufficient for an analysis including the whole underlying process. The tools are mostly equipped with a graphical user interface appropriate for end-users and employ an interactive way of using the build-in methods.
For more complex analysis special web usage mining methods are derived from data mining. These methods have to be applied on web logs in combination with additional data sources like transactional or billing data from other computer systems in order to achieve appropriate results. Software tools supporting those evaluations mostly implement only a single method and are designed for research purposes which results for example in a scarcer user interface like a command line interface. Furthermore, their usage needs expertise for preparing the input data and interpreting the results. This leads to a restricted number of users.
Even if those tools support more convenient user interfaces their usage still needs knowledge available only from experts or highly sophisticated users. For high-grade applications the tools have to be offered together with this expertise as a "complete service". To enable economical and attractive projects the tools have to support a powerful and flexible set of methods for complex analysis of web usage. They have to be capable of using a wide range of data sources and formats; analysis processes should be storable for periodical, automated and reproducible use.
These requirements lead to a Web Usage Mining framework able to combine different data sources and formats for a homogenous access from analysis methods that are as well covered with a unified access layer. Both, data sources and analysis methods will be accessed using a XML-based language called Web Usage Mining Markup Language.
Starting with a short look on the Web Usage Mining process in section 2 we will present our Web Usage Mining Framework in section 3 and give an example of an analysis in section 4. Section 5 finishes with a conclusion and outlook on further work.

2 Web Usage Mining process

The Web Usage Mining process [CMS97, HMW02] can be divided into two phases: collecting and pre-processing data sources to form an integrated database on the one hand and the analysis phase itself on the other hand (see figure 1).
During the first step – *data acquisition* – data from different sources is collected, e.g. log files from web and application servers, customer data or billing data. Each data source is piped through a *data pre-processing* to remove nonessential data, add

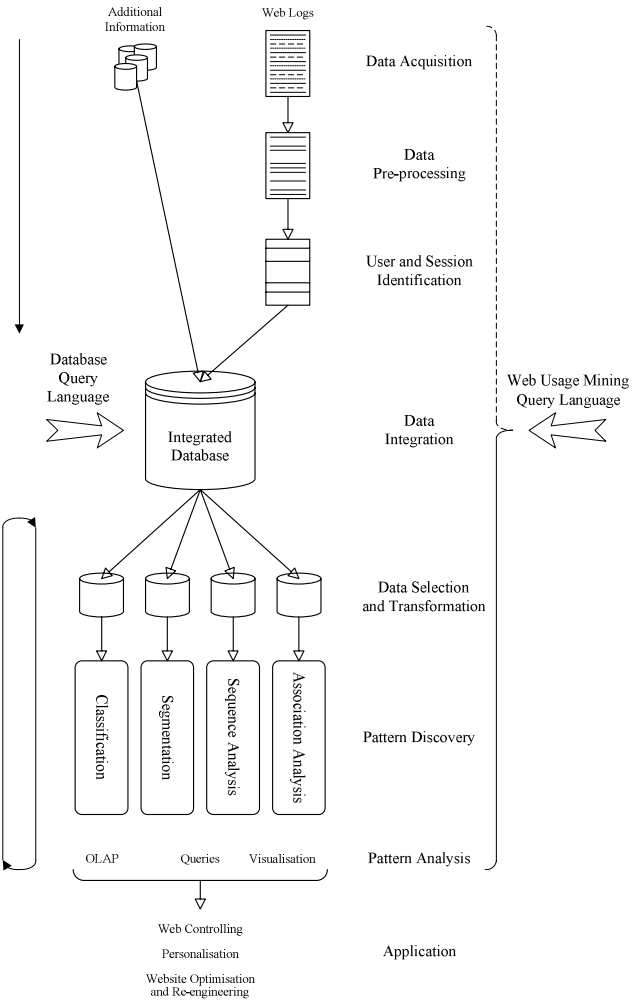

Figure 1: The Web Usage Mining process [cf. CMS02, HMW02]

specific indexes etc. In addition, web log files undergo a *user and session identification* in order to arrange the log files with their strictly time oriented order to a form where the action of users can be seen more easily.
After having pre-processed the relevant data an *integrated database* is formed that contains all data required for the analysis afterwards.
Analyses start with *data selection and transformation* to exclude actually unwanted data, make conversions to more efficient representations, or apply e.g. group functions. WUM methods are then carried out to perform *pattern discovery* and

pattern analysis where interesting but yet unknown relationships can be discovered or specific key data is derived. Whereas the pre-processing phase is passed only once, the analysis phase may be applied several times to achieve comprehensive results.
We developed a framework for performing WUM called WUM framework that structures and enables the application of WUM methods by means of a WUM language. As depicted in figure 1 this language is designed to control the whole WUM process whereas a database query language can only address the integrated WUM database. Our WUM framework takes into account that neither the development of WUM methods nor the variety of data sources for WUM is completed yet.

3 Web Usage Mining framework

Our WUM framework has been developed starting with an abstraction from the data sources involved in the WUM process: e.g. web or application log files, web site graphs, as well as additional information like customer or billing data, product catalogues and product group hierarchies. To form a flexible, extensible and easy-to-use database, all data as well as the querying results are represented in XML. We designed an XML language called Web Usage Mining Markup Language (WUMML, pronounce: WUM-M-L) that is capable of representing and processing these data. WUMML provides simple as well as several complex data structures like rules, sets, sequences, graph, or hierarchies to represent the query inputs from the database and the query results.
In the next step during development approaches for querying and manipulating these data sources and structures – like SQL, XQuery, XSL, or XPath – have been researched and integrated into WUMML. While using those concepts language terms for the depiction of all steps of the WUM process within WUMML were defined. They allow to control the WUM process and to execute WUM analysis methods. Due to this WUMML acts like an interface connecting the WUM data and the WUM methods. Since XML as the basic language supports a modular design WUMML inherits this capability and supports the extension with new language terms and the integration of other XML languages.
In the following sections we present an overview of the main building blocks of our WUM framework. The basis is formed by an abstract data model that enables the depiction of any WUM relevant data as XML structure. Based on this model we provide the WUMML in order to represent, process and analyse the examined data structures. An interface to WUM software tools is used to process WUMML terms. The functionality needed to answer a WUMML query is provided by the WUM tool in form of modules (see figure 2).
The design of our WUM framework follows the principles of classical 3-tier architectures: the data layer (see section 3.1) forms the basis for the functional layer (in our WUM framework called processing layer, see section 3.3) and the

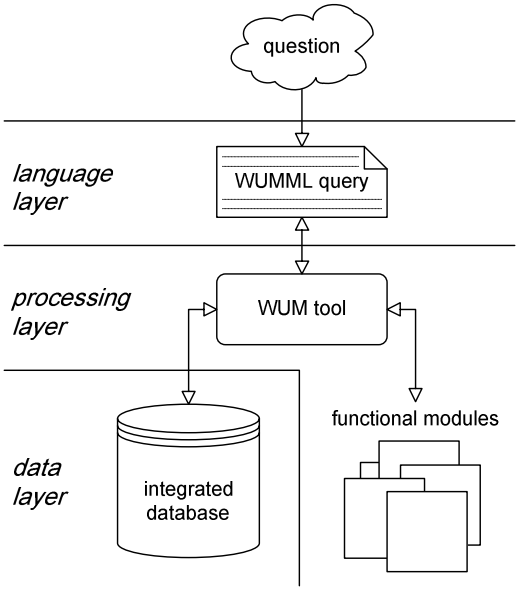

Figure 2: WUM framework

presentation layer (in our WUM framework called language layer, see section 3.2) as shown in figure 2.

3.1 Data layer

Modelling of the data layer is necessary for surveying the properties of the WUM data structures and taking them into account during further development. The resulting data model characterises the data layer adequately in order to define a language for processing the WUM data structures based on this data model. The model should demand as few requirements as possible for the data layer in order to keep the prerequisites for using the language on a low level. In order to develop a complete data model the WUM data sources and the data structures used by WUM methods have to be analysed. Since the integration of the existing WUM methods into our WUM framework is not completed yet, we will thus concentrate on the WUM data sources that have already been taken into account in our data model.
Due to their availability the most popular data sources for WUM are Web Server Logs. Enhancements build upon these logs include the Aggregated Log [SF98] or the Log Markup Language (LOGML) [PUK01]. Both are more efficient formats in respect to data processing than Web Server Logs. However, data from Web Server Logs allow no characterisation of cause of user's behaviour and thus have to be costly pre-processed for WUM. To avoid this one has to use other techniques of data collection. One of these is Application Server Logging in which the required data is individually generated at the application layer [Ko01]. Due to a better data

quality this technique is to be preferred for high quality WUM, however, compared to classical log formats the individuality of Application Server Logging yields to a multitude of formats.
Besides collection and representation of web usage data it is important to consider the other data sources that are incorporated into the WUM process. Often background knowledge (like the web site structure) or additional information (i.e. production and sales data) are integrated into the data basis for enlarging the analysis or enhancing the quality of the results. Such secondary data also has to be considered by the data model of WUM data. Even though the variation of secondary data is extremely great, its format usually is homogenous, because it is almost exclusively stored in (relational) databases and thus can be represented in relations.
The challenges of modelling the WUM data layer consist of developing a data model that represents any existing type of WUM data. The data model has to describe the structure of the considered data elements, but neither their typifying nor the operations that can be performed on it. Our aim is to characterise the data units syntactically for the purpose of processing them and not to restrict their range semantically. In the following the data structures so far covered by our data model are informally described.
An *attribute* characterises an entity concerning a specific property and consists of an attribute name and an attribute value. An attribute forms the smallest data structure to which any other data structure refers. The relationship to other data structures is circular, because an attribute can use any other data structure in its attribute value. With this mechanism any data structure might be created.
An *element* evolves from combination of several attributes. This data structure is used to encapsulate multiple attributes, which in conjunction characterise the same data unit. Whereas an attribute specifies one single property of a data unit, such a data unit itself is represented by an element.
Rules are used to describe correlations; they consist of condition and consequence. The consequence states what applies or occurs when the condition becomes true. Both parts of a rule are formed by a data structure of the type element.
With the term *set* we denote the disordered collection of data units of similar type. Instances of the data structure element are used to form sets. Because elements can be designed individually by nesting the other data structures, the design of sets is not restricted by constraints and arbitrary complex structures can be constructed.
A variation of a set is a *sequence* that contains data units of type element, too, but they are ordered like a linear linked list. Thus a sequence can be understood as an ordered set. An enlargement of a sequence is constituted by the *hierarchy*. Their elements may not only be linear but also hierarchical related to each other.
The greatest abstraction of sets, sequences and hierarchies is portrayed by a *graph*. It consists of data units of similar type, which are denoted as nodes and characterised with attributes. A relationship between two nodes is specified by

means of edges. Any number of directed edges between the nodes is permitted, even though leading to unconnected or cyclic graphs.
With the data structures mentioned beforehand, one is able to depict data suitable for processing it during WUM. The XML representation of these data structures used in our WUM framework is shown in figure 3. For readability we use a simplified notation as fusion of schema description and document instance. It can be read as XML instance document without contents supplemented with a complete enumeration of all XML elements and attributes allowed in the particular context. Parentheses, Boolean operators and wildcards known from DTDs are used to describe frequencies and conditions. Square brackets are used to refer to an XML element defined somewhere else. The *<meta>* elements are needed for information about a data structure instance itself.

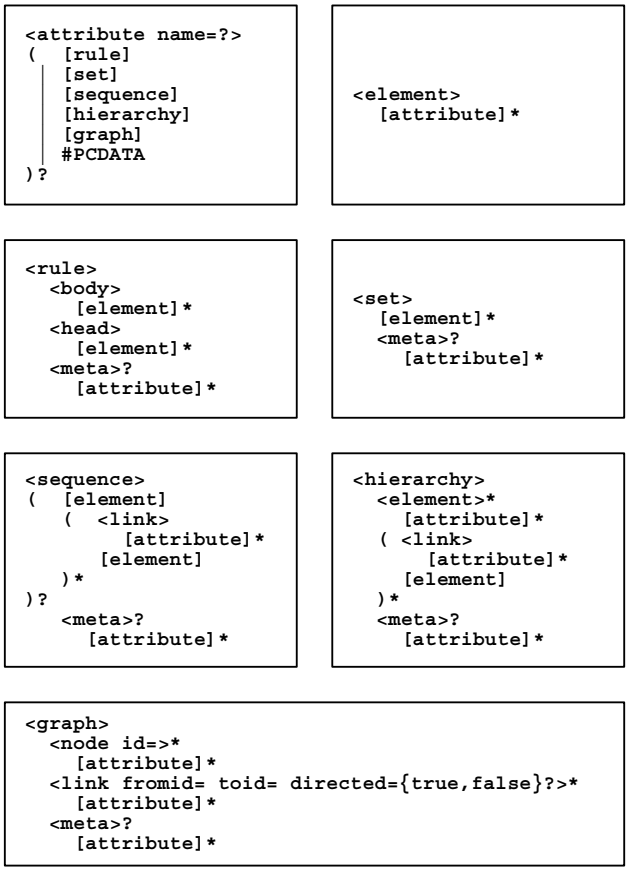

Figure 3: WUM data structures

3.2 Language layer

Central component of our WUM framework is a language called Web Usage Mining Markup Language (WUMML), which enables performing the WUM process. WUMML is a declarative language, which serves to manage WUM analysis by i.e. calling WUM methods to be performed and expressing properties of the results by defining constraints. This is to be done with so called WUMML *queries*. A WUMML query consists of one or more interrelated *statements*; the term query denotes a set of statements that are executed together as a whole. Statement refers to every complete command of the language, which can be executed autonomously. A statement can be broken down in *units*; these are the smallest entities of the WUMML vocabulary, which may depend on other language units and are not executable separately.
The vocabulary of WUMML is divided into four areas: (i) statements to structure WUMML queries, (ii) statements for basic data management and data pre-processing, (iii) statements for mapping of WUM methods as well as (iv) statements for defining functional modules. As shown in figure 4, the specification of WUMML is based on XML, which means that WUMML is designed as a XML language. The language units of WUMML are specified with XML Schema. Besides XML Schema for defining the WUMML schema other XML standards are used with WUMML like XLink for linking of documents, XPath for addressing of elements, XSLT for transformation of documents, XQuery for querying data and XSL-FO for visual representation of XML documents. Because there are various XML languages for a number of applications, a solid basis of related languages is

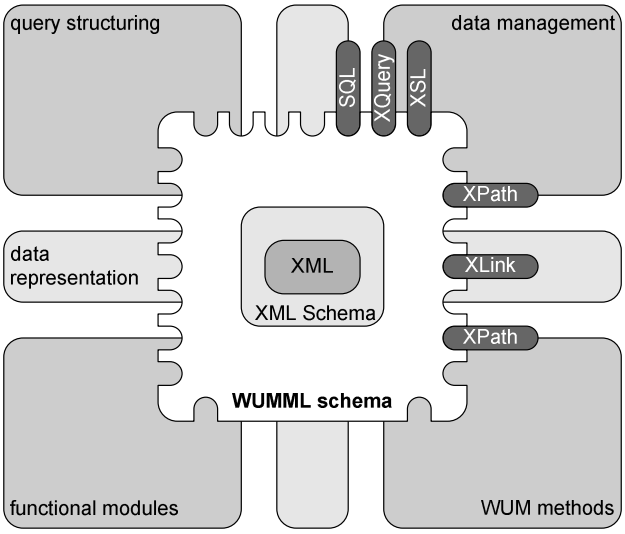

Figure 4: Language design of WUMML

available, that may be integrated into or combined with WUMML. For example SVG may be used for generation of graphical illustrations.
The definition of statements for mapping WUM methods into WUMML already has taken into account insights and constructs of languages for the particular WUM method such as SQL MINE RULE [MPC96, Me03] or MINT [SF98]. WUMML, for example, adopts SQL MINE RULE's concept of groups and clusters in association analysis as well as the way of defining constraints in sequence analysis used in MINT. Usually languages used by specific WUM methods are proprietary, whereas WUMML consistently integrates them into a uniform XML language. Due to the plurality of WUM methods their integration into the WUMML vocabulary is not yet completed, but this can easily be done with the implementation of the existing research results. We will neglect this aspect for the moment and concentrate on the function of specific WUMML statements.
For structuring WUMML queries two statements are available: *<session>* is to be used as the XML root element for encapsulating WUMML queries. The term *session* refers in substance and time to a connected interaction between WUM tool and user. During a session several statements are executed; besides administrative operations a session incorporates primarily performing tasks that are addressed by WUM. WUM *task* refers to the application of a WUM method against a defined data basis for solving a considered question. In WUMML this is done with *<task>*. With *<task>* any implemented WUM method may be executed. So far we have designed statements for the mapping of association and sequence analysis. Figure 5 shows the grammar of the WUMML statements *<session>* and *<task>*.

```
<session>
   (   [data]
     | [save]
     | [transformation]
     | [query]
     | [preprocessing]
     | [visualization]
     | [task]
   )*
```

```
<task name=? dataname= resultname=?>
    <description>?
(   [associationrules]
  | <clusters>
  | <classification>
  | <paths>
  | [sequences]
)?
```

Figure 5: WUMML statements *<session>* and *<task>*

```
<query dataname= resultname=?
  language={sql,xquery}
  xlink:href=? xlink:type="simple"?
>
  <description>?
  #PCDATA?
```

Figure 6: WUMML statement *<query>*

The data to be worked with has to be included into *<session>* with the *<data>* statement. For representing the WUM data in WUMML several XML elements exist for the data structures mentioned in the previous section. By the usage of *<data>*, instances of these data structures are generated. The application of a WUM method with *<task>* requires that a valid instance of the data structure –which is needed by the particular method – exists. The method result is again an instance of one of the defined data structures. Other statements for data management are *<save>* for saving instances of data structures, *<query>* for examining them and *<visualization>* for generating graphical representations of a data instance. *<preprocessing>* and *<transformation>* are available for manipulating data instances. While *<preprocessing>* is used to perform the known techniques (like user and session identification or path completion) for the preparation of data from Web Server Logs, *<transformation>* enables arbitrary manipulation of any data instance, provided that the result is complaint with the defined WUM data structures.
We will not discuss the mentioned statements in detail as this would go beyond the scope of this article. Only a short view on the *<query>* statement may exemplarily show the integration of other XML standards within WUMML (see figure 6). With *<query>* it is possible to declare a SQL or XQuery statement. Alternatively one may give a file that contains such a query by the use of the XLink attribute *href*. This query is then evaluated by the WUMML interpreter against the data structure denoted in the XML attribute *dataname*. The result may be used for further processing like visualisation and is referenced by the name declared with the XML attribute *resultname*.
As mentioned before the development of statements for the mapping of WUM methods is not finished yet. To take advantage of an individual mechanism for integration of new WUM methods, so called functional modules may be used. They are defined by the *<module>* statement with so called module definitions and will be explained in the next section.

3.3 Processing layer

A WUMML query is processed by a WUM software tool (see figure 2). Each statement contained in the query, i.e. for execution of WUM tasks or for pre-processing data, has an (if necessary empty) input and output and is abstractly called *function*. This is a notion for every command that the WUM tool in connection with WUUML provides to the user. The implementation of a function itself is called *algorithm*; several algorithms may exist for the same function and may be automatically or manually chosen depending on the demands.
The functional range of the WUM tool is variable and is affected by the existing *functional modules*. A functional module works like an add-in that provides conclusive functionality. Every module addresses exactly one function and contains at least one algorithm; the WUM tool may have several functional

modules respectively algorithms that implement the same WUM method. In this manner the functionality of WUM tools can be flexibly configured.
The functional modules respectively the provided functions and algorithms are defined in so called *module definitions*, which form a documentation of the functional range of the WUM tool. Simultaneously the module definitions declare a schema of the associated WUMML statement, which is necessary for using the functionality of the module. The schema is needed to decide whether a considered WUMML query correctly calls for an existing WUM method.
If a functional module merely provides new algorithms for existing functions of the WUM tool or the implementation of a method already integrated in WUMML, the schema will be omitted because its information is already available via existing module definitions. Thus a function module may either contain another implementation of an already specified method without providing a schema of the corresponding WUMML statement, or the module has to provide a new method including its implementation and the schema of the corresponding WUMML statement.
Finally, a module definition states, to which extent an instance of a WUM data structure is transferred into a new instance of another (possibly empty) data structure by application of the function specified in the module. For this purpose the valid input and output data structures of the function und thus WUM method are defined.
In contrast to the data or language layer the realisation of the processing layer uses existing solutions in the form of WUM tools, which have to be equipped with the extensions mentioned before and have to be integrated into the WUM framework. In the field of WUM there already exist software products or research prototypes that may play this role, if their functionality is adapted according to the requirements of the WUM framework.

4 Example of a WUMML statement

In the following we give an example of using WUMML for sequence analysis [cf. SF98]. We want to examine which pages users visited after having visited a specific web page. The necessary WUMML query is depicted in figure 8. It is presumed that the data source forms a data structure of type *set* whose elements are described by the attributes *sessionid*, *date* and *url* (see figure 7). It is supposed that we obtained session identifiers by the use of proactive logging techniques and that the data source is already pre-processed. Thus using *<preprocessing>* for data cleaning and path completion is not necessary.
The *<sequences>* statement for performing a sequence analysis asks for connected sequences without wildcards, since we want to reconstruct exactly the path followed by the user. *<identification>* specifies which data objects form the items of the sequence analysis (single page views) and by means of which criteria they can be distinguished (*select*) and chronologically ordered (*timestamp*). *<grouping>* declares which items belong to the same sequence.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<set>
  ...
  <element>
    <attribute name="sessionid">481</attribute>
    <attribute name="date">2005-25-04 14:10:05</attribute>
    <attribute name="url">http://www.uni-hohenheim.de</attribute>
  </element>
  ...
</set>
```

Figure 7: Sample input for sequence analysis

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<session xmlns="http://www.wi1.uni-hohenheim.de/2005/WUMML"
         xmlns:xlink="http://www.w3.org/1999/xlink"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.wi1.uni-hohenheim.de/2005/WUMML wumml.xsd">

  <data name="sessions" language="wumml" xlink:href="input.xml"/>

  <task data-name="sessions" result-name="sequences">
    <sequences type="connected" output="sequences">
      <constraints>
        <constraint type="template" condition="a * b"/>
        <constraint type="selection" condition="$b.$support / $a.$support > 0.1"/>
        <constraint type="selection"
                    condition="$a//attribute[name='url'] = 'http://www.uni-hohenheim.de'"/>
        <constraint type="projection" condition="attribute[name='url']"/>
      </constraints>

      <identification itemset="/set/element" timestamp="attribute[@name='date']"
                      select="attribute[@name='url']"/>
      <grouping select="attribute[@name='session-id']"/>
    </sequences>
  </task>

  <visualization data-name="sequences" language="wumml" output="text"/>
</session>
```

Figure 8: Sample WUMML query for sequence analysis

```
<set>
  ...
  <element>
    <attribute>
      <sequence>
        <element>
          <attribute name="url">http://www.uni-hohenheim.de</attribute>
        </element>
        <link/>
        <element>
          <attribute name="url">
            http://www.uni-hohenheim.de/studium/index.htm</attribute>
        </element>
        <link/>
        <element>
          <attribute name="url">
            http://www.uni-hohenheim.de/studium/hinweise_vvz.htm</attribute>
        </element>
      </sequence>
    </attribute>
  </element>
  ...
</set>
```

Figure 9: Output of WUMML query for sequence analysis

Conditions for generating sequences are defined with *<constraints>*. According to our question the first constraint expresses the pattern, that applies to a web page *a*, whose visit results in a view of page *b* after an arbitrary number of unconsidered page views. Corresponding to the confidence in association analysis the second constraint determines the conditional probability for a page view of *b* given *a*. We are only interested in sequences for which the probability of visiting page *b* after page *a* is higher than 10 percent. The third *<constraint>* element constricts page views that can be bound to *a*. The last constraint with type *projection* limits the extent of the output. It specifies that the output should only contain the *url* attribute of the page views.
An excerpt of a possible result of the query is displayed in figure 9.

5 Conclusion and future work

The introduced WUM framework forms a consistent and comprehensive method for a structured execution of WUM. It is an integrative, open and thus extensible technique that is capable of integrating any WUM method and data source.
The core is formed by the Web Usage Mining Markup Language (WUMML), which enables formulating WUM analysis as queries. WUMML constitutes a new interface between users and WUM software tools and plays a similar role like SQL for database management systems. WUMML allows definition and repeated execution of arbitrary complex analysis schemes. The language is realised completely in XML and uses common XML standards to enable easy processing and integration into existing WUM software.
WUMML supports flexible formulation of queries for answering the considered questions and offers the user to satisfy his or her specific information demand individually and systematically. Up to now comparable query languages existed only rudimentary for selected WUM methods in delimited applications.

Further work is necessary for practical realisation of the framework by means of developing a WUMML interpreter, which may be integrated into existing WUM software tools. We are looking for a mining tool with adequate functionality to adapt it to the requirements of our WUM framework and to develop a first software prototype that fully implements this WUM framework.
To offer WUM as a service a WUM software tool with comprehensive functionality has to be used, but such a tool might not be available. For professional purposes it is not sufficient to use several tools during one WUM analysis, even though the WUM framework and WUMML would support it. To overcome the problem of insufficient functionality of single WUM tools the functional modules characterised in our WUM framework could be realised as web services. In this way, a WUM tool dynamically obtains the functionality to process a WUMML query from the web.

References

[CMS97] Cooley, R.; Mobasher, B.; Srivastava, J.: Web Mining: Information and Pattern Discovery on the World Wide Web. In: Proceedings of the 9th IEEE International Conference on Tools with Artificial Intelligence (ICTAI), Newport Beach, USA, November 1997; pp. 558-567.

[HMW02] Hippner, H.; Merzenich, M.; Wilde, K.D.: Grundlagen des Web Mining – Prozess, Methoden und praktischer Einsatz. In (Hippner, H.; Merzenich, M.; Wilde, K.D. Eds.): Handbuch Web Mining im Marketing. Konzepte, Systeme, Fallstudien. Braunschweig, Wiesbaden, 2002; pp. 3-31.

[Ko01] Kohavi, R.: Mining E-Commerce Data: The Good, the Bad, and the Ugly. In: Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), San Francisco, USA, August 2001; pp. 8-13.

[MPC96] Meo, R.; Psaila, G.; Ceri, S.: A New SQL-like Operator for Mining Association Rules. In: Proceedings of the 22th International Conference on Very Large Data Bases (VLDB), Bombay, India, September 1996; pp. 122-133.

[Me03] Meo, R.: Optimization of a Language for Data Mining. In: Proceedings of the 2003 ACM Symposium on Applied Computing, Melbourne, USA, March 2003; pp. 437-444.

[PUK01] Punin, J.R.; Krishnamoorthy, M.S.; Zaki, M.J.: LOGML: Log Markup Language for Web Usage Mining. In: WEBKDD 2001: Mining Web Log Data Across All Customers Touch Points, Third International Workshop, San Francisco, USA, August 2001; pp. 88-112.

[SF98] Spiliopoulou, M.; Faulstich, L.C.: WUM: A Web Utilization Miner. In: International Workshop on the Web and Databases (WebDB), Valencia, Spain, March 1998; pp. 184-203.