RECONSTRUCTION OF THE WEB APPLICATION HYPERTEXT MODEL USING WEB LOGS

Igor Rožanc  
Faculty of Computer and Information Science  
University of Ljubljana  
Trzaška 25  
1000 Ljubljana, Slovenia  
email: igor.rozanc@fri.uni-lj.si

Marko Poženel  
Faculty of Computer and Information Science  
University of Ljubljana  
Trzaška 25  
1000 Ljubljana, Slovenia  
email: marko.pozenel@fri.uni-lj.si

ABSTRACT
An approach for automatic reconstruction of a part of a web application description, namely the hypertext model, using web logs is presented in the paper. The approach includes an advanced analysis of a large amount of the web logs data and the construction of the Application Transition Graph (ATG) as the formal description of hypertext model. Although the proposed reconstruction is limited on the navigational structure of the web application, the defined process does not depend on any particular technology and can be performed in an automatic way. The obtained results are (1) a complete model of those parts of a web application that are actually in use and (2) a very good starting point for obtaining a complete model including the unused parts as well. In either case, the approach significantly reduces the complexity and costs of reverse engineering. As a case study the approach is applied on the web student information system eStudent from the University of Ljubljana, Slovenia.

KEY WORDS
Reverse engineering, web application, hypertext model, web log, ATG

1 Introduction
A correct and complete understanding of the requirements is the most critical step in the development of any kind of application. When a new application is to replace an existing application, the knowledge built into the existing application the latter becomes a valuable source of the domain knowledge. In the optimal case the existing technical documentation of the existing application contains a complete definition of formal requirements and a complete technical specifications. Unfortunately this is a very rare situation: mostly the only up-to-date source is the code of the existing application, usually upgraded with some information collected during the application lifetime. Thus, the knowledge required has to be extracted by a careful analysis of the assets available, and presented in a suitable formal description (i.e MDD compliant model [1]). This process is called reverse engineering [2, 3].

Apart from some obvious technology and case dependent issues of reverse engineering, e.g., the exact definition of the suitable approach, its level of the automatisation and the amount of resources including staff, time, and cost, some more research oriented questions arise as well. First, which sources are to be used to produce certain results needed? Which part of analysis can be performed if the sources are limited? What is the optimal formal description of the extracted knowledge? And finally, how to determine the optimal balance between all available sources, the expected results and the required efforts for the reverse engineering approach?

In this paper we address those issues for reverse engineering of the web applications. The ideal result of reverse engineering is a formal definition somehow equivalent to the model of the application from the design phase [2]. In case of a web application, it has to cover diverse views of the application. Obviously, the complete web application description has a lot of details and thus its extraction tends to be a highly complex task using all sources available. Usually it is performed by a number of diverse time consuming steps combining the static and dynamic approaches [3]. The static approach analyses the source code of the application, i.e., the complete source code parsing, while the dynamic approach executes the application and analyses the execution effects, i.e., systematic application execution using diverse inputs. Some of the steps might be completely automated or at least machine supported by commercial or custom developed tools. However, the expensive expert manual assistance might be the only solution for many other steps [4, 5]. Most such approaches [3, 4, 5, 6] share a number of problems: they require a lot of time and expensive tools, and they can be performed by experienced stuff only. Frequently those approaches may be even non-applicable as they demand the access to unavailable sources, i.e., the source code of the application.

To overcome these problems, this paper presents an alternative approach for automatic reconstruction of a part of the knowledge built in the web application from the limited sources. More precisely, it produces the hypertext model of a web applications from web logs. The approach is based on the advanced analysis of a large amount of the web logs, something that can be performed efficiently in an automatic way. Additionally, the approach is (1) cheap...
as no additional resources are needed, (2) universal as it is not source dependant and is thus applicable for diverse web technologies, and (3) highly usable as it produces the standard web application description. On the other hand the information incorporated in web application logs is limited — it mostly contains the navigational details and some content information in special cases only. Nevertheless, if the results of approach do not meet the expectations, they can be used as a first step of a more profound (and far more expensive) reverse engineering.

A short demonstration of the approach is presented on a case study based on the web information system eˇStudent at the University of Ljubljana, Slovenia [7]. This IS a large Oracle Portal\(^1\) web application which has recently been reengineered to a new web application.

The rest of the paper starts with Section 2 which contains a definition of a suitable web application description with an emphasis on the hypertext model details. Section 3 gives a presentation of the web logs details and the processing necessary to obtain clean session data for further processing. The core of the paper is Section 4: it describes the steps necessary to generate the hypertext model using a large amount of the web log data. The approach performance is shortly presented in the case study of the eˇStudent web application in Section 5. The paper is concluded with discussion and conclusion. For the purpose of this paper, figures obtained from the generated diagrams have been manually translated to English.

2 The Formal Web Application Description

A complete web application description has to cover diverse views of the application, namely its content, navigation and presentation details. Muller [8] has advised the description should consist of (1) a business model, (2) a hypertext model, and (3) a presentation model. The business model describes the functionality of the web application using diverse diagrams to define the business logic details. The hypertext model describes the navigational structure of the web application, namely how web pages are built and linked together. The details of the graphic appearance of a web application are presented in the presentation model. The three-model web application description is the approach supported by most methods [9, 10, 11], however different authors might use different model names.

Clearly, the extraction of all three models requires all available sources and a heavy duty reverse engineering. By limiting the sources to the web log data we obtain the insight into some navigational details only. In special cases some content information might be derived from the names of visited web pages while the presentation description is clearly not part of the web log data. Thus we focus on proper presentation of the hypertext model only.

Several formal descriptions including FSM [12] and WebML [13], are available for the presentation of the complex navigation through a dynamic web application. However, the approach presented by Offutt and Wu [14] was selected as the most appropriate as it presents the extracted navigational knowledge with simple graphs. Additionally, the approach is aimed towards common dynamic web application, thus covering diverse web technologies currently used.

The Offutt and Wu approach uses the Atomic Section Model (ASM), a well known model developed for testing web applications primarily but used elsewhere as well. Each dynamic HTML page is represented by an ASM (spatially named Component Interaction Model - CIM), which defines the page’s detailed internal structure and transitions, i.e., calls to other dynamic pages. The CIMs of all dynamic HTML pages are combined together into a large Application Transition Graph (ATG), which offers a global preview of the entire web application structure and navigation. Thus, the ATG of a web application is the formal representation of the hypertext model.

The CIM of a dynamic HTML page is a quadruple \(\text{CIM} = \langle S, A, CE, T \rangle\) with

- a set of start pages \(S\) from which the page is referenced,
- a set of atomic sections \(A\) the page is made of,
- a component (regular) expression \(CE\) describing the page structure, and
- a set of transitions \(T\) pointing from and to (other) HTML pages.

An atomic section (AS) is a basic block of code, where all lines of code in the block must always be executed in an uninterrupted sequence. The component expression is presented as a regular expression and it defines all possible combinations of diverse ASs to dynamically construct a valid HTML page. The set of transitions defines a number of calls to dynamic pages, described with corresponding parameters and classified into diverse predefined groups.

Usually the first component of a CIM tends to be a single start page, and the sets \(A\) and \(T\) contain fairly small number of elements resulting in simple CIM graphs which are suitable for quick insight into a single dynamic web page structure. An example of a CIM for the web page from the eˇStudent web application is presented in Figure 1.

The production of all CIMs for the dynamic pages enables the composition of the ATG. Formally, the ATG of a dynamic web application is a quadruple \(\text{ATG} = \langle \Gamma, \Theta, \Sigma, \alpha \rangle\) with

- a set \(\Gamma\) of dynamic HTML pages (CIMs),
- a set \(\Theta\) containing all transitions of all CIMs,
- a set \(\Sigma\) of variables defining possible states of the presentation layer, and

\(^1\)The company, product and service names used in this paper are for identification purposes only — all trademarks and registered trademarks are the property of their respective holders.
• a set \(\alpha\) of all diverse starting pages (usually one).

The detailed ATG presentation can become highly complex when it is designed as a number of CIMs glued together. However, the ATG is usually presented at a higher level of abstraction as a directed graph with a set of vertices \(\Gamma\) and a set of edges \(\Theta\). This presentation is optimal for a quick insight into the complete web application navigational structure.

3 The Web Logs

3.1 The web logs definitions

Web servers save all interactions with clients in web server logs. These are primarily collected for debugging purposes although nowadays they are commonly used as a source of data for data mining and user behaviour analysis as well.

The level of the saved details depends on web server setting and the web log format the web server uses. The web log file formats are defined in W3C organisation standard so they can be analysed by diverse web analysis programs. Frequently web servers use the default format named Common Log Format (CLF) for logging the interaction with clients. It contains seven basic data attributes: IP address, host id, user id, date and time, requested source, HTTP status code and number of bytes sent.

The W3C format called Extended CLF (ECLF) contains two more attributes: page referrer and user agent used. The first denotes the web page linked to the requested source while user agent identifies web browser and the operating system used [15]. ECLF format offers much more possibilities for a quality analysis of the web log data, but even ECLF may not be sufficient and thus many web servers log other data elements (like cookies or server IP) as well. Unfortunately, the number of data elements in log file is limited due to HTTP protocol which does not offer much information in the first place [16]. Table 1 presents the comparison between CLF and ECLF log file formats.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>CLF</th>
<th>ECLF</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>host</td>
<td>●</td>
<td>●</td>
<td>IP address / domain</td>
</tr>
<tr>
<td>ident</td>
<td>●</td>
<td>●</td>
<td>User ID / '-'</td>
</tr>
<tr>
<td>authuser</td>
<td>●</td>
<td>●</td>
<td>User ID / blank</td>
</tr>
<tr>
<td>date and time</td>
<td>●</td>
<td>●</td>
<td>Date &amp; time</td>
</tr>
<tr>
<td>request</td>
<td>●</td>
<td>●</td>
<td>Source at web server</td>
</tr>
<tr>
<td>HTTP status</td>
<td>●</td>
<td>●</td>
<td>HTTP status code</td>
</tr>
<tr>
<td>bytes sent</td>
<td>●</td>
<td>●</td>
<td>Bytes sent to client</td>
</tr>
<tr>
<td>referrer</td>
<td>●</td>
<td>●</td>
<td>URL of request</td>
</tr>
<tr>
<td>user-agent</td>
<td>●</td>
<td></td>
<td>Browser &amp; OS</td>
</tr>
<tr>
<td>filename</td>
<td>Fit name</td>
<td></td>
<td>File name</td>
</tr>
<tr>
<td>cookie</td>
<td></td>
<td>●</td>
<td>Value of cookie</td>
</tr>
</tbody>
</table>

Table 1. The CLF and ECLF log file format [15].

The log files of web servers are the most common source of the clickstream data [17]. A clickstream is called a sequence of clicks a single user makes while browsing through a website. The complete clickstream defines the user session representing one visit of the user to the web site. Spiliopoulou et al [18] define a session as a sequence of user actions from the moment he entered the site until he left. Thus, the user session starts with the first user request and ends with the last one. If the time span between two consecutive page requests of a certain user exceeds a certain predefined value, the session is divided into two sessions.

The clickstream data in a web log file contain a large number of mixed sessions for the diverse (or even the same) user. Because raw log data are noisy and contain lots of unnecessary data it has to be properly preprocessed in order to get useful data. Thus we have to cleanse data, identify web users, identify sessions and determine the correct order of requests inside the user session. This process is called sessionization [19] and it is shown in Figure 2.

3.2 The information in the web logs

The drawback of the web logs for the data analysis is that (1) only limited data is available and (2) the quality of raw data may be low [17]. The first problem sets the limits of such data use, i.e., navigational description of the web application, while the latter gives insight into the complexity a suitable process implementation. In our approach, the second problem is addressed by the preprocessing step.

Using user sessions extracted from web logs we can
reconstruct a complete set of all different (dynamic) web pages and transitions between them. These web pages and transitions are sufficient to construct the ATG of the web application, however without the CIM details. Additionally the names of the retrieved web pages names may present some content information.

Additionally, the data in web logs can be incomplete, noisy, and sometimes without visible transitions between pages. One of the biggest problems for the processing of web log data is that user sessions are not always explicitly connected. Kohavi [17] denotes nine deficiencies of the web logs:

1. the web log data do not identify sessions and users;
2. it is not connected with transaction data;
3. lack of important events;
4. the web logs do not store web form data;
5. the missing semantic information of what URLs means;
6. the web logs lack information for web applications that dynamically generate the page content;
7. the web logs are large ASCII flat files;
8. contain lots of doubled and redundant information (i.e requests for images);
9. the web log lack important information like screen size.

The extraction process has to overcome (or at least diminish the effect of) these problems. The quality of web log data can be raised with the quality preprocessing phase before the data analysis.

Despite above disadvantages web logs represent important source of data for analysing user and web site behaviour. One of the most important advantages of the web logs is the amount of data as they represent complete live usage data for all users. Additionally, the text format of web logs files does not require any additional software on client side.

### 3.3 The preprocessing

Due to the incomplete and noisy log data proper session reconstruction can be a challenge. Many authors were researching the problem of user session reconstruction [20, 19, 21, 22] and developed various heuristics. Brendt et al [20] claim that the quality of clickstream data is affected by (1) proper user identification, (2) ability to detect the end of user session and (3) faithful reconstruction of the activities in the user session. Each of the issues has to be resolved in order to get faithfully reconstructed sessions. Badly reconstructed sessions can affect the quality and they can result in misleading conclusions.

Users can be identified according to ip, cookie or with the help of other heuristics [20, 16]. The technique used depends also on web log format used. User session ends when user leaves the web site, which is detected by a longer timeout after his last request. Different authors propose different timeout values [23, 24], but usually 30 minutes timeout is used [19]. If user makes a request after that time, a new user session is detected. Faithful reconstruction of the activities in a user session can be a challenge due to the appearance of interleaved sessions as well. Those are usually performed by advanced users. An interleaved session is a session made by a single user simultaneously browsing the same web site in two or more browser tabs, windows or different browsers. In this case we have two different user sessions, but only one longer session is detected. Because the interleaved sessions have a negative impact on the quality of the retrieved results, a solution has to be found.

Identification of interleaved user sessions is not straightforward and it can not be performed without some kind of contextual help. Our solution is to perform the session separation process using knowledge of past user behaviour and state space search [22]. Each request is appointed to the most likely session using the information on users’ previous behaviour. The result is the set of reliably reconstructed user sessions. Any further presentation of the details of the separation of the interleaved sessions is beyond the scope of this paper.
4 The Approach Definition

The complete approach can be summarized as a sequence of the following five steps:

1. The access to the web log data: the web log files are transferred from the web server to the database;
2. The preprocessing of the web log data: the sessionization process and the identification of interleaved user sessions are performed;
3. The identification of the hypertext model data: the sessions data are processed to determine the dynamic pages and the transitions between them;
4. The optimization of the hypertext model data: the results are summarized into a correct definition of ATG elements representing the hypertext model;
5. The presentation of the hypertext model: the ATG graph and a list of separate web pages names are presented to the analyst.

Table 2 presents the inputs and results of each step.

STEP 1 The web log data are saved in several large textual files at the web server and these must be copied to a separate working directory first. The large amount of the log file data may cause problems during the analysis. To obtain the better performance of the consecutive steps we parse the files, clean the data and insert the cleaned data into a database. In the cleaning phase the web spider records, duplicated entries (due to the specifics of application server) and the entries not related to specific web application are removed. Usually a smaller part of the data is considered correct and thus usable for further processing.

STEP 2 The preprocessing of the data is performed in two steps both supported by a custom made tool we provided. First, the sessionization phase identify the users according to IP and use this information to reconstruct the user session records in the chronological order. The sessions data is separated and saved into a separate table. Then, the interleaved sessions are identified and faithfully separated following the process described in section 3 to obtain the clear session data. The final result is not entirely correct, but still exact enough to be used in further steps.

STEP 3 The further processing of the session data, i.e., the sequences of the dynamic pages visited by single user, results in a complete set of start pages, dynamic web pages ever used in web application, and a set of transitions between them. It is important to observe the quantity of redundant data as most pages and transitions are repeated over and over. However the tool we provided is able to distinguish the sets of diverse starting pages, (dynamic) web pages and most probable transitions. The tool defines a union of those dynamic pages and transitions by iterating through all sessions and counting the total number of occurrences for each element separately. Once all sessions are processed, the transitions repeated more than a predefined number of times are considered correct only. This is necessary as session data might still contain errors. Such probabilistic approach gives good results in practice, but a bigger problem may be the processing of such large amount of data. Of course, all elements are saved into the database.

STEP 4 Next we apply a heuristic to further determine certain inconsistencies in the data. First we reconstruct sequences of web pages which are always visited in the same order. These web pages may represent special web pages dedicated to login, logout or reporting procedures. Such pages may be (or may be not) mapped into one CIM to simplify the final result.

Additionally, certain transitions may lead to a web page opened in a new window, something that may be understood as a separate session with a wrong first page. Two situations may arise: the dynamic page in new window represents a temporary page without further transitions, i.e., help window, or a separate session containing diverse content had been actually started. A heuristic approach based on user previous behaviour is used to distinguish between both types of situations as well.

Finally, the cleansed data, i.e., starting web pages, web pages equivalent to CIMs, and transitions, is saved separately into the database table. At this point even optimised data may still have certain inconsistencies and we plan to address these issues in our future work. Unfortunately, no additional information on the hypertext model, i.e., CIM model details, can be retrieved from the data available.

STEP 5 The last step is the graphical representation of the ATG graph. This can be achieved using the DOT tool or other similar approach. To build the graph the retrieved data on dynamic pages and transitions is processed. Each CIM is represented by as vertex with a name only, as further information on CIM details can not be understood from web logs data. Additionally, a list of all different dynamic pages is exported as well.

5 Case Study: eStudent Web Application

The approach was demonstrated in practice on the web student information system named eStudent (developed and initially deployed at the Faculty of Computer and Information Science of the University of Ljubljana, Slovenia) [7]. It is a quite large three-tier web application built using the Oracle (DB and Portal) technology (more than 220,000 lines of code in total). The development of eStudent started in 2001 and by 2003 the initial release has been used by most faculties of the University of Ljubljana supporting approximately 20,000 users - students, professors and staff. Its main functions are providing electronic support for student enrolment, management of examination records and
grades, and keeping the alumni records. The developer team consisted of developers who knew the requirements in practice. They used an agile methodology of development which proved efficient [25] for the development, however resulted in poor technical documentation.

During its development and maintenance the eŠtudent accumulated a huge amount of the domain knowledge and proved to be reliable web application. Nevertheless, due to the introduction of the imminent new Bologna study programs in 2009 the requirements for the student information system changed significantly, i.e., the extensive selective courses support was needed. Additionally, the technical solution the eŠtudent was built on became outdated. After a few unsuccessful attempts a new version of the information system is currently being built and the “old” eŠtudent is about to be retired.

Due to a poor technical documentation the idea of extracting domain knowledge from eŠtudent became popular [26]. A quite complex static method to extract domain knowledge into an MDD compliant model using source code was presented and suitable tools were proposed [5]. Although the method requires a considerable amount oftime, it does deliver the results. Thus the approach presented in this paper was developed to find an alternative way to partially solve the problem.

We developed a prototype on a case study for the eŠtudent to demonstrate the issues addressed in the previous section. The web logs accumulated over a span of twelve years (2001–2013) of active use at two middle size faculties were used and thus several hundred million logs in approximately 25 web log files were checked. Between 500.000 to 700.000 diverse eŠtudent user sessions were successfully extracted from web logs data per year at the end of STEP 2. The processing of web logs per one year took approximately an hour, and it was executed entirely automatically.

For the demonstration a small amount of collected sessions were processed further, just enough to prove the correctness of the approach. Each session was processed separately to collect ATG elements data. Even a small number of sessions provide quite good results, however to be entirely correct a large number of resolved sessions from different time intervals should be used. An example of a simple session is shown in Table 3 where most of the unnecessary details were omitted. In the last column the repeated calls to the same web page (refresh or resubmit) is given.

After STEP 3 the data were further optimised in STEP 4 to be graphically present the ATG. The development of this part took a lot of effort and has not yet been fully completed. To demonstrate the results for the session in Table 3 the ATG graph in Figure 3 was created manually. In the figure the special web pages (the web page sequences for login, logout and report preparation) are coloured white and yellow, while grey nodes represent the core application dynamic web pages. Of course, to recognise them as special, several sessions had to be analysed first as described in STEP 4.

After the partial extraction process of the eŠtudent it was discovered that the entire ATG graph contains one start web page (login), more than 280 different dynamic

<table>
<thead>
<tr>
<th>STEP</th>
<th>Description</th>
<th>Input</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Access</td>
<td>web log files</td>
<td>web logs in DB</td>
</tr>
<tr>
<td>2</td>
<td>Preprocessing</td>
<td>web logs in DB</td>
<td>sessions in DB</td>
</tr>
<tr>
<td>3</td>
<td>Identification</td>
<td>sessions in DB</td>
<td>web pages, transitions in DB</td>
</tr>
<tr>
<td>4</td>
<td>Optimisation</td>
<td>web pages, transitions in DB</td>
<td>ATG elements in DB</td>
</tr>
<tr>
<td>5</td>
<td>Presentation</td>
<td>ATG elements in DB</td>
<td>graphical ATG, web pages list</td>
</tr>
</tbody>
</table>

Table 2. The inputs and results of separate steps

<table>
<thead>
<tr>
<th>Seq</th>
<th>Time</th>
<th>Page accessed</th>
<th>Refresh</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6:55:05</td>
<td>login</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>6:55:05</td>
<td>ls_login</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>6:55:12</td>
<td>check_password</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>6:55:13</td>
<td>process_signon</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>6:55:14</td>
<td>DYN_SCHEME_SELECTION</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>6:55:15</td>
<td>DYN_INIT_PARAMETERS</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>6:55:16</td>
<td>DYN DISPLAY PAYMENTS</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>6:55:21</td>
<td>DYN EXAM CERTIFICATE</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>6:55:27</td>
<td>DYN_ENTRY_OF_DECISION</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>6:55:31</td>
<td>DYN OPEN EXAM REGS</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>6:55:37</td>
<td>DYN OPEN EXAM REGS</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>6:55:45</td>
<td>GET REPORT</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>6:57:51</td>
<td>GET REPORT</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>7:00:23</td>
<td>DYN DISPLAY PAYMENTS</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>7:00:26</td>
<td>DYN EXAM REGISTER</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>7:00:37</td>
<td>DYN EXAM REGISTER</td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>7:00:41</td>
<td>DYN EXAM REGISTER</td>
<td>1</td>
</tr>
<tr>
<td>18</td>
<td>7:00:52</td>
<td>DYN EXAM REGISTER</td>
<td>1</td>
</tr>
<tr>
<td>19</td>
<td>7:00:58</td>
<td>GET REPORT</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>7:04:50</td>
<td>DYN GRADES DATA</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>7:04:56</td>
<td>DYN GRADES DATA</td>
<td>1</td>
</tr>
<tr>
<td>22</td>
<td>7:05:05</td>
<td>DYN GRADES DATA</td>
<td>1</td>
</tr>
<tr>
<td>23</td>
<td>7:05:16</td>
<td>DYN GRADES DATA</td>
<td>1</td>
</tr>
<tr>
<td>24</td>
<td>7:05:26</td>
<td>DYN GRADES DATA</td>
<td>1</td>
</tr>
<tr>
<td>25</td>
<td>7:05:41</td>
<td>DYN HELP</td>
<td>0</td>
</tr>
<tr>
<td>26</td>
<td>7:06:48</td>
<td>DYN GRADES DATA</td>
<td>1</td>
</tr>
<tr>
<td>27</td>
<td>7:07:01</td>
<td>GET REPORT</td>
<td>0</td>
</tr>
<tr>
<td>28</td>
<td>7:19:27</td>
<td>DYN LOGOUT</td>
<td>0</td>
</tr>
<tr>
<td>29</td>
<td>7:19:32</td>
<td>ls_logout</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 3. An example of a user session.
Figure 3. The simplified ATG corresponding to the session data presented in Figure 3. White and yellow nodes represent special web pages.

web pages (per faculty) and a quite complex set of transactions. Each dynamic page is used to construct a number of actual dynamic web pages using different data (i.e. DYN_GRADES_DATA is used every time a teacher grades a student). This ATG data proved to be too complex to present it graphically in the paper. However, in our case a system for naming the dynamic pages was used and this proved to be very helpful for the right positioning of the elements in graph.

6 Conclusion

In the future the approach has to be tested on various web application logs as well. Nevertheless, the approach proved to be successful as we are able to obtain a complete (in sense of covering all web pages) and presumably correct ATG. Most importantly we can perform it (mostly) automatically, thus using a number of custom developed tools to perform the described steps. Some minor manual manipulation to provide correct inputs for the tools is used as well. However, two aspects must also be considered:

- the information contained in the web logs is limited and we were able to obtain as much as possible using the presented steps, and
- the described processing and the final result must be compared to alternative approaches (in our case to the approach presented by Rožanc and Slivnik [5]) which are much more staff and time demanding.

The presented approach evolved gradually and it is still being refined. Many improvements were introduced as the results proved to be incomplete. A lot of effort was invested in STEP 2 (the correct sessionization and the reconstruction of the interleaved sessions) while STEP 3 (the optimal transactions detection) and STEP 4 (the ATG elements optimisation) are still to be improved. A suitable solution for STEP 5 has to be discovered too.

Additionally some interesting research issues emerged as well. The number of web log data (and consequently the extracted session data) is very large. Is it necessary to correctly analyse the entire amount of data? If the analysis is to be performed on partial data only, how to determine the proper subset of data (in the sense of correct sampling)? And, is it possible to combine the web log data with some other data available to upgrade the process and obtain better results, i.e., CIM definition? These issues are left for future work.

References


