MASTER

Framework for adaptive web-based systems

Brouwer, S.R.J.

Award date:
2014

Link to publication
1 Abstract

Recommenders and other adaptive hypermedia are changing the customer’s experience on websites. By adapting the website or web-platform, it is possible to provide the visitor with a personal fit of menus and content.

The adaptation is done on-the-fly, improving the user experience during the session. For example: if a user is browsing the Internet for a small car, the website of a car manufacturer could adapt the content and navigation towards this personal interest by hiding information about large sized cars.

The goal of this research is to develop a global framework based upon standardized open-source techniques, that allows adaptation for experts and novice developers. The global framework consists of basic techniques as a layer to create an easy-to-use adaptive website. The framework is build using PHP as programming language for server-side scripting and JavaScript as client-side scripting. The webserver processes the web-requests and makes sure the PHP-scripts are executed upon request.

As a proof of concept, the adaptation layer is programmed as a hook onto the PHP framework of CodeIgniter (build by Ellislab). The adaptation layer consists of a library of adaptation functions. These functions are linked by the use of specialized tags, placed in the output of the web application. The output is processed by the layer before being sent to the web-client. The adaptation layer can adjust or reshape the output of any requested file. HTML pages can be dynamically updated using background calls to the webserver. In this study, CodeIgniter was used to prove the concept of a global adaptive framework. However, the programming classes used for the adaptation layer can be easily changed to fit another framework or applications.

The adaptation of a webpage requires knowledge about the concept and its user(s). Knowledge about the user needs to be stored and updated. The information about the main concept is outlined in sub-concepts and the relationships between those and variables. Relationships are bound between variables and contain a specific weight. The variable value of a specific concept can be inherited from parent-child and others types of relationships. The user-model, containing the user-data, are the values of the variables with an identifier that binds the user to the value. By the means of queries within different adaptation functions, aggregation is possible by simply ignoring the user-identifier. User-data can be automatically removed after expiring to keep the system stable and light.

Keywords: adaptive, web-based systems, adaptation, hypermedia, content adaptation, presentation adaptation, navigation adaptation
2 Preface

When I started with the web-adaptive systems course, I was fascinated by the idea that it would be possible to change a website ‘on-the-fly’ using a model that focuses on the subject of that website.

The adaptive systems I have been working with were mainly focused on learning environments, keeping track of the knowledge about underlying concepts. This knowledge level would then influence the navigation and content of the webpages. I was surprised not to find any companies or content-management packages that were using an adaptive user model focused on the subject of the website. Also, the framework provided by the University requires specific installation on dedicated servers, and it also proved not to be very stable.

Based upon these perceptions and experiences I decided to build a framework for an Adaptive Web-based System (AWbS). I choose to base this framework on open source techniques like Apache, PHP and MySQL. These techniques are globally used for large commercial web environments. The framework can be extended by experts and configured by the use of a Graphical Web-based User Interface (GWUI).

I started this project by doing research on the current adaptive web-based systems and websites /applications that were built using these systems. To prevent an overload of research information, I limited the research to websites using content management systems (CMS). A CMS provides an easy way to edit the content of a webpage. The results of my research are used as input for determining the structure of the new framework. The framework focuses on interoperability, scalability and performance.

The new framework was subjected to multiple tests measuring performance, scalability and interoperability.

Sander Brouwer
Eindhoven, July 2014
# Contents

1. **Abstract**  
2. **Preface**  
3. **Abbreviations**  
4. **Introduction**  
   4.1 Thesis context  
   4.2 Problem description  
   4.3 Research questions  
   4.4 Methodology  
   4.5 Outline  
5. **Research**  
   5.1 Current web-applications  
   5.2 Adaptive web-based systems  
   5.3 Business perspective  
6. **Analysis**  
   6.1 Goals and structure of websites  
   6.2 Implicit information by request  
   6.3 What to adapt?  
   6.4 Adaptation  
   6.5 Designer and domain-expert  
   6.6 Concept relationship Model  
   6.7 User-models  
   6.8 Usability  
   6.9 Acceptable performance  
7. **Design**  
   7.1 Design decisions  
   7.2 Standalone vs. hook on existing web application  
   7.3 Programming languages  
   7.4 User-, domain- and adaptation-model  
   7.5 Adaptation for novice and expert  
   7.6 Global scripts  
   7.7 Adaptation commands  
   7.8 Proof of concept  
8. **Adaptive Framework**  
   8.1 Basic functionality  
   8.2 Adaptation layers  
   8.3 Adaptation processor  
   8.4 Performance
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9 Evaluation</td>
<td>39</td>
</tr>
<tr>
<td>9.1 Performance</td>
<td>39</td>
</tr>
<tr>
<td>9.2 Limitations</td>
<td>39</td>
</tr>
<tr>
<td>10 Conclusion</td>
<td>40</td>
</tr>
<tr>
<td>11 Future work</td>
<td>41</td>
</tr>
<tr>
<td>11.1 Future developments</td>
<td>41</td>
</tr>
<tr>
<td>11.2 Future research</td>
<td>41</td>
</tr>
<tr>
<td>A Appendices</td>
<td>44</td>
</tr>
<tr>
<td>A.1 CodeIgniter basic installation settings</td>
<td>44</td>
</tr>
<tr>
<td>A.2 CodeIgniter controllers</td>
<td>52</td>
</tr>
<tr>
<td>A.3 CodeIgniter framework hook</td>
<td>75</td>
</tr>
<tr>
<td>A.4 Adaptation class (adapt.class.php)</td>
<td>76</td>
</tr>
<tr>
<td>A.5 Concept relation variable class (crv.class.php)</td>
<td>93</td>
</tr>
</tbody>
</table>
List of Figures

1. Internet Domain Survey Host Count ........................................... 8
2. Graphical representation of the used methodology .......................... 10
4. Five level AHS authoring model .................................................. 14
5. Network Analysis http://www.google.nl ....................................... 18
7. Architecture for generating server-side files .................................. 20
8. Structure ....................................................................................... 27
9. Concept Relation Variable Model .................................................. 36
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Standalone vs. hook</td>
<td>24</td>
</tr>
<tr>
<td>2</td>
<td>Domain and user-model vs. one simplified model</td>
<td>25</td>
</tr>
<tr>
<td>3</td>
<td>CodeIgniter framework vs. major CMS web application</td>
<td>26</td>
</tr>
<tr>
<td>4</td>
<td>Concept Structure Example</td>
<td>36</td>
</tr>
<tr>
<td>5</td>
<td>Servers for measuring the performance</td>
<td>38</td>
</tr>
</tbody>
</table>
3 Abbreviations

AHAM ........ Adaptive Hypermedia Application Model
AHS .......... Adaptive Hypermedia System
AM .......... Adaptation Model
AWBS ........ Adaptive Web-based System
CI .......... CodeIgniter
CMS .......... Content Management System
CPU .......... Central Processing Unit
CSS .......... Cascading Style Sheet
DM .......... Domain Model
GB .......... Gigabyte
HTML ........ HyperText Markup Language
HTTP ........ HyperText Transport Protocol
IIS .......... Internet Information Services
IP .......... Internet Protocol
JS .......... JavaScript
MB .......... Megabyte
MVC .......... Model View Controller
PHP .......... Hypertext Preprocessor
RAM .......... Random-access memory
UM .......... User Model
UML .......... Unified Modeling Language
URI .......... Uniform Resource Identifier
URL .......... Uniform Resource Locator
4 Introduction

This master thesis is the result of my master study business information systems. The thesis is focussed on adaptive web-based systems, through the growth of the world-wide-web and my personal interest in building web-applications. The growth of websites and associated information is clearly noticeable in daily life. Figure 1 shows the growth of registered Internet addresses (domains). The visitors of websites use filters, search-engines and recommenders to find their way through the information. The recommenders are commonly based on user-profiles, and these profiles contain the valuable information about the interests of the visitor.

4.1 Thesis context

The world-wide-web started as a practical project to bring a global information universe into existence using available technology. Sharing information c.q. knowledge by using hypertext, is achieved through HTTP (HyperText Transport Protocol). The $w^3$ data model consists of hypertext links and text. The authors came up with features like: saving information only once, links that allow the topology of the information to evolve, indexes for searching and virtual documents generated by the server in response. A server provides a response based on the structured request of the clients, providing an HTML document.

Later on, different file types were supported in the form of style-sheets, images and documents. Virtual documents are built by using GET and POST requests and server-side scripting. It started by providing search results based on a user query. Every user entering the same query would then get the same result from the server. By caching at the server-side, recalculations could be prevented as long as the cache does not expire. Client-side scripting allows the web-builder to provide the user with more dynamics e.g.: moving objects, alerts and updates without full-page refresh. Some client-side scripts connect to external / remote sources.

By creating user-profiles, relevant information about the visitors can by stored on the server systems. The user-profiles give the system the ability to adapt advertisements and other content on the website to fit the user-profile. In most circumstances, this adaptation is directly bound to a fixed set of rules. There are other ways to adapt the provided web-content to fit the current user by using environment factors sent during the request, e.g. device, screen-width, time (including time-zone), date, IP-address, etc. These factors can be used to adapt the web-content to the user without saving or building a user-profile.

4.2 Problem description

Currently, most websites are designed following a 'one-size-fits-all' philosophy. Each visitor gets the same page-view, except for personal information. The increasing growth of information provided...
online makes it more difficult for visitors to fetch the required / requested data. Web-developers try to improve this experience through the use of search-engines and filters. In most cases, visitor analysis is used to improve the website afterwards. By using the visitor behavior as a model for direct improvements, the experience can be directly adapted to the current visitor.

The main question is:

*It is possible to define a scalable and interoperable framework that supports adaptive web-based systems with an acceptable performance?*

### 4.3 Research questions

The framework should support current web-production environments and rely on stable common techniques to ensure interoperability with currently used web-systems. According to the main question, the following research questions were defined:

1. *Current web-applications using a framework*
   Which type of framework is currently used by websites? Are there possibilities to filter, search or recommend content? The outcome of this question should give an impression if the currently used systems could be adapted to support adaptivity.

2. *Current adaptive web-based systems*
   The adaptive systems that were built and available, their technical demands and limitations. Thereby assessing if one can use a generic or specific adaptive framework.

Next to these research areas, the business perspective will be shortly enlightened to provide an argument for the ‘Why would you want an adaptive framework?’ question.
4.4 Methodology

The following methodology is used to answer the main and research questions defined:

![Graphical representation of the used methodology](image)

**Figure 2**: Graphical representation of the used methodology

4.4.1 Literature research

The literature research will focus on the ‘as-is’ situation, adaptive hypermedia and the use of specific techniques. The ‘as-is’ situation is a research into websites at the moment. The adaptive framework should connect with the currently used server-side scripts. The adaptive hypermedia is a look at related work. Related work can be in the form of articles, but also systems supporting adaptive web-based systems. The research will be the basis for the design decisions used for the framework.

4.4.2 Analysis

The research together with the (main) questions will be analyzed and formatted to create a specific outline for the design phase.

4.4.3 Design

The design phase will outline the adaptive framework in technical and functional requirements used during the build phase. The research questions in 4.3 should lead to a specification of the framework.
4.4.4 Building

To create a proof of concept, the framework will be partially built on the CodeIgniter framework. Also the functionality will be discussed.

4.4.5 Evaluation

The evaluation phase will be used to evaluate the research-project and the framework. The evaluation will also include a part about the limitations caused by the design choices and further research and development steps.

4.4.6 Conclusion

The conclusion of the research-project will be drawn by answering the main-questions.

4.5 Outline

This thesis will further be structured by the methodology outlined in 4.4.

Chapter 5 will describe the research
Chapter 6 will describe the analysis
Chapter 7 will outline the design decisions
Chapter 8 will describe the framework
Chapter 9 will outline additional research, limitations and improvements to be made.
5 Research

This research has been divided into three parts. The first part is about the current web-applications using a framework. The second part is about adaptive web-based-systems that were build. A website can be built from scratch or with the use of an existing framework. This framework is a layer on top of the webservers, providing basic functionality. A commonly used framework is a CMS. CMS’s allows people without expert programming knowledge to change webpages and other parts depending on the functionality of the system.

Finally, the third part of the research will focus on targeting visitors and the role of adaptation towards business purposes. The information available on the world-wide-web continues to grow. Also businesses are expanding their information about the available products and services, where in the past, for example, only contact information was presented. Visitors maybe be exploring, looking for specific topics, requiring support, etc. and still finding their own way through the structure of the website.

5.1 Current web-applications

Web applications use different techniques such as filters, recommenders and search-engines to help the visitors find their way through the content. These techniques can be improved by using implicit and explicit user-preferences. These preferences can be stored in a user-profile. A content-management-system lets non-experts edit the content of webpages via a web application.

5.1.1 Search engines

Search engines are helping users to find their way through the information available on the Internet or on a specific website. These search engines are instructed by the user by entering keywords. These keywords are matched with the indexes of the search-engines, providing the user with a list of URL’s that might match the search, in order of relevance. Typically the same result is returned if another user enters the same key-words into the engine [21]. In most cases, every user of the search engine has a different need. The results of a search engine can be improved by using personal information such as relevance feedback, interest, demographic information or user ratings.

5.1.2 Filters

A visitor might use filters to select specific content. These filters are shown on the graphical user-interface and connected with algorithms processing the values set. The graphical representation differs in the property type. In case of a price of an object, it can be a two-sided slider, of which an example can be found in figure 3. Filters can be boundaries set on specific properties of the subject. This item can be a product or service.

5.1.3 Recommenders

Recommender-systems are another method to prevent a user from searching through the massive amount of information. They are

![Web-content-filters](Tweakers.net (2014))
commonly used in e-commerce solutions e.g. to show user related products or products that fit earlier searches.

5.1.4 Personalization

A user-profile can be used to improve filters, searches and recommendations. This user-profile can be set-up by explicitly asking the user to enter preferences / interests, but also (semi)automatically by tracking and processing behavior of the user. The implicit information can also be extended by the basic information provided in the web-request, such as location based on IP-address, browser, device and time(zone).

Advertisements can be based on stored search history and the use of cookies. The main challenge is describing the search, making the user the expert on the subject he or she is looking for.

5.1.5 Content management systems

CMS’s are server-side systems used to keep websites manageable. The management application is commonly web-based and accessible via the world-wide-web. Multiple CMS’s, like Wordpress and Joomla, give a novice the ability to change the content of webpages easily, include news-feeds, blogs, etc. The website w3techs.com is constantly monitoring the use of CMS’s, in June 2014, 38.4% of all websites where using a CMS. CMS’s commonly provide WYSIWYG (What You See Is What You Get) editors, giving users the ability to easily change the content of the website. Wordpress is used the most (22.4 percent of all websites). Wordpress is an open-source system, started from an idea to create a blogging-system (https://wordpress.org/about/). Adaptability described as, for example, in the article ‘the next big thing’ [14] is not used in above named CM-systems. Most large websites use content management systems with modules and components. These components or modules may have recommendation systems, or they adapt things based upon certain profiles, but this profile is not globally used within the web application. This is, for example, only available in the web-shop component where a recommendation-system is used to recommend products based on the product-pages the user is visiting. Recommendation systems are commonly used by e-commerce platforms. Recommenders are based upon user-models and historical data. The current recommenders focus mainly on one parameter, e.g. what the user likes.

5.1.6 Templates

The use of templates is increasing. Web-designers design all kinds of lay-outs in order to sell them on marketplaces such as ‘Envato themeforest’ [7]. Most of these templates use specialized techniques to support different devices and web-browsers, such as Bootstrap (front-end framework) and jQuery (JavaScript library) [2][4]. [17]

5.1.7 Tracking and cookies

A reference of the visitor is one of the requirements for a server-side script to identify customers. This reference is called a ‘session’. The session-cookie is the file placed on the client’s system during the processing phase of the request containing a unique identifier. The server also includes an expire timeframe for this session, telling the client to remove the value after a certain amount
of time or after closing the web-browser. The server can locally store session-data, or the session-identifier can be used in the database to store user-specific information. This information could lose its reference if the session expires, becoming untraceable during a new-session. Therefore, user-information can be bound to a username or e-mail-address. Cookies can also contain personal information or can be set by other servers loaded through JavaScripts for further analysis of user behavior. Systems like Google Analytics are used by a variety of websites creating a great insight in the web-browsing history of the visitor.

5.1.8 Acceptable performance

The retrieval-time for a webpage is crucial for the impression and motivation of the visitor [20]. Visitors might cancel the retrieval of the web-site, research in 2004 shows that users are willing to wait 2 seconds for a simple information retrieval [19]. The accepted waiting time has become much shorter, users are used to fast loading webpages. 250 milliseconds can change the impression of a visitor [17].

5.2 Adaptive web-based systems

Research has shown that implementing adaptive navigational support and adaptive presentation support can increase the speed of navigation and understanding of the content [11][16]. The adaptive hypermedia systems like AHA! (Adaptive Hypermedia Architecture), are using a reference model called AHAM, which is a concept hierarchy to describe relations between the different sub-concepts, pages and fragments [16]. These properties are described in the domain model. This domain model does not take the goals and/or tasks into account. As described in ‘From Adaptive HyperMedia to the Adaptive Web’ [12], the domain model should be less static. The model shown in figure 4 shows the five level AHS authoring model [13]. The domain-model is the top-layer, respectively connected to the question ‘What?’, covering the concepts c.q. subject of the system. The underlying model should be more dynamic if you make the domain-layer less static. The domain-model used in the most AHS (Adaptive HyperMedia Systems) can be considered as a kind of concept-relationship-model. AHAM supports different types of relationships, like the prerequisites. The UM (User Model) represents properties of the user used by the AM (Adaptation Model). The user-model structure is a predefined setup that matches the adaptation-rules. These rules use variables and their values during the adaptation process. The user-model is commonly stored in files or a database on the webserver. The behavior of the visitor can update the user-model. It can repre-
sent properties like knowledge, interest, preferences, goals and objective, etc. These properties can be combined with specific concepts. The adaptive systems developed are mostly focused on learning environment. Creating user-models that keep track of the knowledge of the user on specific concepts described in the domain-model. The adaptation-rules define the route through the available information.

5.2.1 Generic vs. specific

The adaptive web-based systems built so far can be generic or specific. The Grapple research project was aimed to build a generic adaptive framework while, for example, the ‘adaptive tourist suggestions for a commercial web site’ of the university of Torino was specifically focussing on tourists visiting a city [15]. The adaptive tourist website is using adaptive filtering to prevent overload of information towards the visitor. The design is specifically engineered to support tourists visiting Torino. The system could be adapted to support other cities, but is not usable for a website with another purpose e.g.: selling cameras. Another common specific field for adaptive technologies is education. Technologies like curriculum sequencing, creating individual paths, creating the optimal way for a specific student to go through the material [10]. Different systems were designed for educational purposes e.g.: ELM-ART, Interbook, etc. [10].

5.2.2 Roles for implementation

The currently used Adaptive HyperMedia Systems require three levels for setting-up the application. The first is the technician who sets up the low-level programming. The second is the domain-expert who has the knowledge about the content. The third is the middle man, translating the outcome of the domain-expert to actual pages within the application. In practice, the one creating the structure of the application is the same person creating the pages, e.g. the one writing the course structure is the one writing the course. Since the domain-model needs to be set-up by a domain expert, this expert should be capable to define the structure and actual pages within the application.

5.2.3 Requirements for implementation

The current Adaptive HyperMedia systems require specific server installations, e.g.: Tomcat, Java and server-scripts. The global web-hosting services are using different techniques to prevent changes to the server, mostly not supporting specific server changes. Since specific changes might influence the stability or vulnerability of the server.

5.2.4 Client-server architecture

The adaptive web-based systems use adaptation rules and different models to generate the adaptation. Writing these models and rules requires the knowledge for defining and programming these systems. The adaptation is commonly generated at the server-side to process the rules before sending the response towards the client. In some cases, adaptation can also be executed on the client-side, by using JavaScript, Cascading Style Sheets or asynchronous server requests. The article on ‘A DSL based on CSS for HyperText Adaptation’ [18], is an example of a client-side adaptive processing method. Client-side processing of adaptation limits the necessary server-capacity, since the client is executing the adaptation rules. Besides the fact that it lowers server-capacity it also allows monitoring visitor behavior more closely because it eliminates the need for communication.
User behavior such as mouse-overs, clicks and window-movements can be used directly by the engine that is also running on the same device. The connection causes latency in case of a server-client-architecture. The client-side processing however limits the use of aggregation of user-data and requires all necessary resources at the client-side. Also, more complicated rules requiring queries or a reasonable amount of processing time can create problems at the side of the client. Think of mobile-phones with limited processors or browsers without database support.

5.2.5 Aggregation of user-models

Aggregation of user-data can lead to interesting statistical data for adaptive systems. The aggregation is only possible in case the user-models are stored at the server-side. The aggregation of user-models is not generally implemented in the current adaptive hypermedia systems. The access to another person’s user-profile could lead to privacy issues but could also improve the adaptation by the use of statistical information based upon the weighted averages or other computations over different user-models.

5.3 Business perspective

The number of websites is constantly growing, almost all respectable businesses have a place on the internet. In most cases the website is to provide the visitor with information about the company, products, services, area of operation and contact-details. A visitor looking for a specific product or service is commonly using a search-engine to find the URL of the companies website. The interest of the visitor can be easily tracked from the behavior on the website and the referral details. Conversion can be increased by adapting the webpage towards the interests of the current visitor.
6 Analysis

The growth of data available on the world-wide-web creates the need to focus on the visitor’s perspective. The visitor of the website can access a page directly by typing a URL (Uniform Resource Locator) or indirectly by using a search-engine. Analytical tools are used to optimize the website for the visitor, in most cases on e-commerce platforms to improve the conversions. These tools lead to reports containing statistical data useful for improvements. Marketing companies are constantly monitoring and targeting users.

Still all webpages containing non-personal data are the same for every user, trying to fit one design and context on all users. This generates the one-size-fits-all problem. Each user might have a different purpose with the visit. These purposes lead to specific behavior on the webpages, for instance only looking at specific categories of products or directly visiting the contact-page. Targeting of users is especially important in commercial businesses and web-environments with a large amount of pages and context. The aim for the web-designer and business behind the websites is to shorten the search for the user and focus on the goal.

6.1 Goals and structure of websites

The initial focus should be on the goals of the website, goals could be (examples):

1. Provide information
   This can provide the visitor with information about a certain topic.

2. Sell products or services
   Think of e-commerce platforms providing product information and the possibility to buy products and services.

3. Socialize
   Social-platforms, blogs, forums and other applications that allow users to connect and share information.

4. Personal
   Personal websites containing experiences and interests.

The goal leads to the structure of the content. This structure is commonly expressed in menus, categories and articles. The menus contain menu-items connected to the different main-pages available. Different menus can exist within the different pages. The structure can be seen as a tree with different branches and leafs. Each branch c.q. category has its own properties leading to specific items.

The purpose of providing information can be sharing knowledge, teaching, but also information about businesses, products, news, etc. One of the fields heavily trying to adapt the web-content to the user are e-learning systems. In these systems the user gets a walk-through based on his current knowledge level. This knowledge level could be determined explicitly by asking knowledge questions, but these might be passed by the user. An implicit way of measuring is difficult, you could look at the content of the pages read and/or visited and keep track of the different aspects of that page. In this case the web application is creating a walk-through by changing the menus, content and presentation based on the knowledge level. In case of a business site, you might want to focus on the visitors interests. The point of measurement changes from knowledge towards interest.
For example: You are a personal car-manufacturer with an informative product website containing all models and their information. The initial pages shown the newest models and a menu with links towards all different models sorted by price from high to low. The visitor browses the first two models. This could imply something about the budget available, but could also be the size the visitor is looking for. In this case a marketeer of the car-manufacturer might start asking questions, would show specifications of both vehicles or offer a test drive.

6.2 Implicit information by request

The visitor provides information by requesting a webpage. A web-request generally contains an IP-address (Internet Protocol address), browser and device information and possibly a referral link. The IP-address gives a rough indication of the visitor’s location, useful for geo-targeting. Geo-targeting is commonly used for internet marketing and also for advertisements. The use of proxy-servers, VPN-connections and internet-providers lower the reliability of the location. The browser used is very important for matching techniques used by the developer and supported by the web-client (browser). Some web applications even warn in case of the visitor uses an outdated or incompatible browser. This information contains also the operating system used in the request. The referer header shows the URL the user is coming from, this can be spoofed or not shown.

All this information provided during requests might be of value to provide the user with the correct content. Of course the use of the information should be based on the goal and relationships with the content. In case the operating system is sent in the request, it might become very interesting. For example: your business sells Apple computers and the visitor uses a Microsoft Windows based computer. From a marketing perspective you might want to convince the user of the pros and compatibility compared with a Windows based system.

6.3 What to adapt?

The webpages need to be changed to provide the visitor with a customized fit, based on implicit and explicit user information. These changes can be categorized in the different elements of the page. The main markup language for webpages is HTML (Hyper Text Markup Language). The HTML supports a variety of elements (tags) that lead to a specific output. HTML received by the client can be rendered by server-side scripting. Client-side coding can be sent or referred within the HTML, where JavaScript is the generally supported client-language. The HTML tags such as ‘img’, ‘script’ and ‘link’ can refer to (external) files. The browser creates separate requests for the external files, figure 5 shows an example of a GET request on ‘http://www.google.nl’. The network analysis shows that some images, scripts and style sheets are loaded. All these files are loaded for the final presentation, rendered on the screen of the visitor. The navigation and context are generally provided by the HTML-code itself.

Figure 5: Network Analysis http://www.google.nl
The ‘graphical’ presentation is commonly provided and set by a CSS (Cascading Style Sheet). The navigation is one of the key elements because this is crucial for the practical use of the website, you need navigation to easily visit other pages without entering direct URLs. This leads to the following elements:

- Content
- Navigation
- Presentation

6.4 Adaptation

6.4.1 Content adaptation

Figure 6 shows an example of a webpage structure. The content part is the essence of the page. The content mainly consists of text together with images and other media. Content adaptation can be:

1. Add / change or remove parts
   An example of content adaptation can be to show an introduction text in case of a first visit. This text is removed or different in future visits.

2. Size and visibility of parts
   Instead of removing a part of the content, the part can be made smaller, bigger or transparent etc.

3. Ordering parts
   Depending on the adaptation, the order of different parts can be changed to fit the adaptation and user-profile.

4. Show or hide toggle parts
   In this case the user can still access parts not recommended by a toggle, after clicking on the toggle the part will be stretched. You could also think of ‘read more...’-links.

The decision to remove parts should be prevented if the reliability of the adaptation is insufficient. Since the visitor is then no longer able to access these parts of the content. This could cause losing potential customers or other viable information for the visitor. In this case it would be recommended to use stretch-text with accordion-menus or tabs, to toggle the hide and display feature of this particular part. Content adaptation can be closely related to presentation adaptation, certainly in if you want to adapt text-sizes and visibility. These properties are commonly defined within the style sheet, also providing the presentation style. However the adaptation of things like text-size can also be defined within the ‘style’-tags of the HTML elements.
6.4.2 Navigation adaptation

The navigation adaptation techniques are used to guide the user through the webpages. The navigational-elements can be hyperlinks, buttons, redirects and other elements leading to a change in content or presentation of the current page. The majority of the adaptive navigation techniques consist of:

1. (Hyper)link sorting
   Hyperlink ordering can be used in structured navigational parts such as menus. The menu-items can be ordered by the adaptation engine.

2. (Hyper)link annotation
   Changing the look of the link by adding colors or symbols to for example indicate the recommended links based on the measured interests of the visitor.

3. Add/remove (hyper)link
   Placing or removing links.

6.4.3 Presentation adaptation

The presentation parts of a webpage are commonly defined within the CSS (Cascading Style Sheet). Specialized style sheets and Java-scripts are used to support different devices, including mobile devices with limited screen-sizes. One of the commonly used techniques c.q. tools is called Bootstrap [2]. Bootstrap is focussing on responsive web-design, changing the content for optimal presentation on every screen-size. It also supports all major web-browser to increase compatibility. The style-sheets and Java-scripts can be generated dynamically by server-side scripts instead of using static files. In this case the web-browser will just receive a correctly formatted JavaScript or CSS file. Figure 7 shows the basic architecture of this concept. The use of adaptation within the CSS described in ‘An adaptation language based on CSS’ is not generally support by all major web-browsers at the moment [18]. Therefore it is not ready for a production business solution, but it is certainly an important upcoming technique for client-side adaptation. The CSS based adaptation can also adapt major parts of the content and navigation elements, since the presentation adaptation is also being influenced by the content and navigational adaptation rules. For example: in case of text-sizes, block or hide elements etc. the CSS can easily provide solutions for this. CSS coding such as ‘display’ can easily toggle blocks of the HTML-page.

6.5 Designer and domain-expert

A website is commonly built by a web-developer. The basic programming is done by a developer in the case of a content management system. In most circumstances the web-developer is not the domain expert, requiring a translation from knowledge towards techniques. The currently used AHS depend on technical specialist to define the structure of the website, while the domain expert is setting-up the content of the page. In practice the person with the knowledge about the domain model defines the structure. This requires technical capabilities on the side of the domain expert.
6.6 Concept relationship Model

The concept relationship model is crucial to bring structure into the domain-model. It creates the ability to define relationships between concepts. These relationships can be used in the calculation of inherited variables and thereby increase the functionality of the adaptation model to fetch required data from the domain and user-model.

6.7 User-models

The behavior of users on an adaptive web-application is commonly stored in a user-model. This model is initialized on the first visit and is updated and extended during the rest of the visit. If the same user returns, the system tries to detect a previously used identifier. A previously used identifier is commonly stored on the device of the user in the form of a cookie. The earlier built-up user-model is loaded and used based on the identifier found in the cookie. Adaptation rules based on a fresh or limited user-model can cause unwanted or unreliable results. Also multiple users using the same system can cause trouble for the adaptation. The adaptation engine could use weighted averages or globals to prevent outliers. It is a small step to calculate averages, since commonly all user-models are stored in the same database. The number of expired user-models will grow substantially, because references (cookies) can be deleted on the client-side or a visitor might not use the web application anymore. To keep the system up to speed, user-models can be converted to a global user-model. This global user-model can then be used for global averages or other statistical data.

6.8 Usability

To increase the usability of the framework it should fit the currently used web-applications. The current web applications commonly use extended content management systems supporting plugins, modules, user-management and easy WYSIWYG editors. The framework should be no reinvention of a CMS, it should work together with the different web-systems. In this case the framework is an independent extension to the other framework. It should therefore use standardized techniques such as PHP, MySQL and be written stand-alone, to easily connect to other web-systems. This interoperability can be easily reached by making use of the provided web-system hooks provided by most modern CMS-systems. A hook is a method of patching in different processing parts of a web-system. Every web-system can use different internal processing steps, but all lead to one comparable output before sending the response to the web-client. This comparable HTML output can be processed by the adaptation layer before being sent back.

6.9 Acceptable performance

As the research in section 5.1.8 indicated, a 250 milliseconds increase of retrieval time can change the user-experience. The retrieval performance partially depends on the server-resources, connection speed. It is difficult to measure the server-resources and performance of a shared-hosting environment, since there is insight in the other systems using the server. A virtual private server is virtual, meaning all resources are shared, but partially reserved for your virtual part. The connection speed depends on the data-connection to the server and more importantly the connection of the client. The loading on a public Wi-Fi network could take significantly longer than on a private internet connection. The client needs to process the information received from the web-server, starting with the HTML-page and then loading the linked resources. Usually the client starts
loading the presentation before all images, videos and other linked resources are received. The processing-time of all data received depends on the client-resources.
7 Design

The challenge for the adaptive framework is to create a globally usable, stable environment for most concepts. The concept is closely related with the goal of the website, which can be informing, selling or learning. The framework moves the searching challenge from the user to the designer of the website. The system should be set-up by an expert in the domain. This expert initializes the system with knowledge about the main-concept of the website. This knowledge should help the system translate the search of the user into a sort of walk-through, e.g.: like a sales c.q. expert helping the customer find the product, based on the specification in the form of demands and preferences. These demands and preferences should be submitted by using indirect and direct information from the client. Depending on the type of web application the user-model could not exist or be incomplete to fully support the user. An incomplete or non-existing user-model should complemented by the global user-model.

7.1 Design decisions

The design decisions are based upon requirements for the framework. The following requirements were defined:

1. Ease of use
2. Scalable
3. Usability
4. Interoperability
5. Minimize cost to use

The adaptive framework could be used for any type of application and should use basic web-technologies. The framework can be installed on any operating system that is able to run the basic web-technologies used, these are shown in figure 8.

7.2 Standalone vs. hook on existing web application

One of the major framework design decisions is to build it for standalone or a collaboration purposes. It will work with an existing application in case of collaboration. Table 1 shows the pros and cons. Together with the ‘usability’ and ‘interoperability’ requirement the adaptive framework is designed to collaborate with existing web application. This prevents businesses to completely redesign their current systems. The framework focuses on adaptability and not to be a great content management system. Domain experts need easy to use editors to write the content and build the structure of the webpages. Therefore a CMS is a good combination with an adaptive layer, creating the best of both worlds. In the proof of concept, the adaptation layer will be a ‘display hook’ on the CI framework. This gives the developer functions to easily use different adaptive techniques within the application, but it is also possible to copy the layer to another application.
<table>
<thead>
<tr>
<th></th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standalone</td>
<td>A standalone system can be totally adaptive on every aspect of the system. Adaptation within functions would be possible, lowering the amount of coding.</td>
<td>The currently used web-applications functionality should be implemented in the adaptive framework. This could cause high development cost and learning curves. Also standardized plugins and component used in the major CMS-system might need to be redeveloped for the framework.</td>
</tr>
<tr>
<td>Hook</td>
<td>The adaptation framework can be connected to any other web application such as a CMS, lowering the amount of coding and programming. This also enriches the total environment and would lower the cost of implementation because the current environment needs minimal changes.</td>
<td>Processing-times could be higher if parsing of already formatted output is necessary. Adaptation between the controllers, modules and views, in case of an MVC-architecture would depend on the hook-functionality of the currently used web application.</td>
</tr>
</tbody>
</table>

Table 1: Standalone vs. hook

7.3 Programming languages

7.3.1 Server-side

The major server-side web-based programming languages are: Java, C, C++, Python, ASP.NET and PHP [5]. However, these languages should be accepted by the majority of the web-servers and web-clients used these days. Languages as ASP.NET require a Microsoft webserver, this decreases flexibility and would increase the cost of usage since the majority of the other languages can run on open-source stable web-servers. Java is an open-source platform independent language. The problem with Java are the shared-web-hosting solutions, in this case a server is shared among multiple web-developers. The functionality becomes very limited by security measures taken to keep Java stable and the server secured [1]. PHP is the language supported by all major web-hosting parties in the Netherlands (‘Argeweb’, ‘YourHosting’, ‘Neostrada’, ‘TransIP’, ‘Versio’, ‘mijndomein.nl’ and ‘Domeinwinkel’) [6]. The CMS’s and e-commerce systems discussed earlier commonly use PHP in combination with the MySQL database. For interoperability purposes these techniques are open-source, multiple platform, support by shared web-hosting-servers and commonly used. Java would be an option for larger businesses with dedicated or virtual private servers. For the proof of concept we used PHP and MySQL.

7.3.2 Client-side

The only globally supported client-side scripting language is Javascript. Other languages such as ActionScript, VBscript require specific web-applications. Dart and Typescript are based upon JavaScript and could be used within the framework, however since the majority of the application will on the server-side, these advantages are not required [3]. The jQuery library for JavaScript can be used for adaptation updates without page-refreshes. This will then execute asynchronous requests to the webserver to get adaptation updates or store information in the user-model.
7.4 User-, domain- and adaptation-model

There are three different models used in the AHA! system. The user-, domain- and adaptation-model. The adaptation-model defines the adaptation and uses data from the domain- and user-model. The earlier design decision to build a hook up on different web-applications such as CMS’s ensures that the adaptation-commands will be defined in the web application. The values written towards the user-model commonly refer to specific concepts within the domain-model. For example: you are tracking the visits of a user to specific pages. The user-model would then contain the values for these specific parts of the structure of the domain-model. Table 2 shows the pros and cons. This led to the design-decision to merge the domain- and user-model in the proof of concept. This is further explained in section 8.3.4.

<table>
<thead>
<tr>
<th></th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single model for the</td>
<td>A more simplified architecture. Preventing duplicate definitions of</td>
<td>Limits the possibility to easily implement domain or user specific</td>
</tr>
<tr>
<td>user and domain</td>
<td>concepts and variables. Easy translation from user-data to global</td>
<td>data-structures.</td>
</tr>
<tr>
<td></td>
<td>statistical information.</td>
<td></td>
</tr>
<tr>
<td>Using separate models</td>
<td>Clear distinction between data belonging to the user and the domain.</td>
<td>Could cause duplicate definitions or structures. Increases complexity.</td>
</tr>
<tr>
<td>for the user and</td>
<td>More advanced model specific properties possible.</td>
<td></td>
</tr>
<tr>
<td>domain</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Domain and user-model vs. one simplified model

7.5 Adaptation for novice and expert

Websites are commonly built by web-developers. The content management can be done by a domain-expert in case of a CMS. These experts maintain the structure and content of the webpages. The adaptation of a website is a walk through the structure and content. Therefore creating easy adaptation commands would allow the novice, in this case the domain-expert, to apply basic adaptation rules within the WYSIWYG-editors of the CMS. The CMS will not process these commands, but parse them on towards the adaptation layer. The adaptation layer executes the commands and sends the processed pages towards the user. Experts can change the adaptation layer and use server-side coding to access the concepts, relations and variables.

7.6 Global scripts

Adaptation commands are written into the content of the pages, menus, modules and other parts of the application in front of the adaptation layer. In some cases, the execution of global commands would decrease duplicates and improve the usability. Therefore there should be an option to define global adaptation commands.
7.7 Adaptation commands

Adaptation commands need to be written in the content management systems. This requires that they are not changed or removed by any part of the web application before arriving at the adaptation layer. Some characters are sensitive for html encodings, such as the greater and less than sign. These can be converted to HTML special chars. The adaptation layer then needs to check for the encoded format.

7.8 Proof of concept

A proof of concept should evaluate whether it is possible to define a web adaptation framework. The adaptive framework should be connected to a web application preferably with content management functionality. However for testing and customized scripting a basic framework would increase the flexibility and speed up the development. Table 3 show the design decision between using a major CMS-systems (Wordpress, Joomla) or using a lightweight PHP framework. The focus on these applications is based upon high usage statistics shown in the research section.

There are all kinds PHP frameworks available and all capable for the proof of concept. The comparison focusses on lightweight PHP frameworks with low level server requirements. During the development, the focus must be on the adaptation layer, not on learning the structure of a certain content management system. Therefore a lightweight PHP framework is used for building the adaptive framework.

To create a proof of concept, the adaptation layer is built on the PHP framework called CI (CodeIgniter), developed by Ellislab. CodeIgniter is a light-weight, easy-to-install framework without content management options. This framework creates a Model-View-Controller layout and comes with general libraries for database operations, sessions, xml communication, forms, etc. The framework needs a basic set-up to let a less experienced user (domain expert) define the structure and content of the pages. The adaptation will be defined within the content of the pages using basic commands supporting the different areas of adaptation discussed in section 6.4.

<table>
<thead>
<tr>
<th>PHP Framework</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Light weighted. Simple installation. Easy to make changes and specialized test. Low learning curve to master the framework.</td>
<td>There is no basic CMS or functionality included, requiring manual coding for the novice-part (CMS) of the framework.</td>
<td></td>
</tr>
<tr>
<td>Major content management system</td>
<td>Fits the generally used web-applications. Contains an easy to use user-interface for setting-up the structure and content of the website.</td>
<td>More complex to test and change parts of the CMS, since the enormous amount of files, classes and functions.</td>
</tr>
</tbody>
</table>

Table 3: CodeIgniter framework vs. major CMS web application
7.8.1 Set-up structure

The structure of the framework is shown in figure 8. In the proof of concept, two servers are used. A shared-hosting environment at TransIP (Dutch hosting company) and a dedicated virtual private server at Versio (Dutch hosting company). The dedicated server is running CentOS 6.4 64bit, Apache Webserver, PHP 5.4 and MySQL. The installation of CodeIgniter only consists of extracting a zip-file and a few database and application related settings, shown in appendix A.1. After the basic installation the hook with the adaptive class file was defined, this code is shown in appendix A.3. For the proof of concept a very basic content management is implemented, allowing the definition of pages, menus and users. This is further outlined in the set-up section of the adaptive framework 8.1.

<table>
<thead>
<tr>
<th>Linux Distribution / Windows</th>
<th>Apache Web-Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHP</td>
<td>MySQL (DB Server)</td>
</tr>
<tr>
<td>CodeIgniter Ellislab</td>
<td>Adaptation Layer</td>
</tr>
<tr>
<td>Expert</td>
<td>Adaptation GUI</td>
</tr>
<tr>
<td>Novice</td>
<td></td>
</tr>
</tbody>
</table>

Figure 8: Structure
8 Adaptive Framework

8.1 Basic functionality

The basic functionality is based on the CodeIgniter framework. CodeIgniter uses a model-view-controller architecture. The controller contains the (partial) application-structure and logic. It also creates the connection with the models and the views. A model is responsible for data processing, so it reads, adds, creates, deletes and preprocesses data for the controllers. Within the view, HTML and application-code are merged. The view is the presentation shown to the user. A few controllers, models and views were written for basic functionality, e.g.: page, user, file and admin controller and some support models and views. The page controller manages the basic content pages. The initial 'Home’-page is loaded from the page-controller with a page-parameter 'home’. The CI-framework handles the routing to controllers in a basic URI-structure of /controller/method/id.

8.1.1 Page controller

The page-controller tries to find the requested page within the database table 'pages'. The table contains a page-name, access-level, page-title and page-text. The access-level assigns the different access-level ids defined in the 'acls'-table. The controller checks whether the requested page is found within the database, and the access-level matches the visitor-access-level. A page can be defined within the page-text column in the table or by using a PHP file within the views of the page-controller allowing an expert to code pages easily. The source can be found in appendix A.2.1.

8.1.2 User controller

The access-level of a visitor is initially set to zero (0). The user-controller can set the access-level after a log-in. The login-controller uses the user table within the database. This table contains the email address of the user, the user-password, the assigned access-level, the first-and last name and whether the user is active at the moment. The source can be found in appendix A.2.2.

8.1.3 File controller

Files can be generated dynamically by the file-controller. This controller is accessible via /file/.. The files are written within the views of the file-controller. For example, a dynamic CSS file:

```php
body{
    <?
    if ( date('H') > 18 ) {
        echo 'background: url(../images/bg_dark.jpg');
    }else{
        echo 'background: url(../images/bg_light.jpg');
    }
    ?>
}
```
By implementing the CSS within the header-template, a different background will be loaded depending on the current time of the day. The adaptation layer also uses these techniques for the presentation-adaptation. The source can be found in appendix A.2.3.

8.1.4 Admin controller

The admin-controller loads a small content-management-system that change values of the different tables described earlier. The controller uses a create-read-update-delete plugin called ‘Grocery-CRUD’. This plugin generates forms for read, add, update and delete operations in the database. A novice can use this part of the system to define easy adaptation-commands and a what-you-see-is-what-you-get editor to set-up page-text. The source can be found in appendix A.2.4.

8.1.5 Menu model

The menu-model loads the normal and admin menu. A query gets the menu-item name, link and position from the menu-table. The user-access-level is passed to the get-menu function. The where-part of the query checks the access-level and menu-id. The menu-identifier refers to the normal or admin menu, but also allows additional menus if needed. The source can be found in appendix A.2.5.

8.1.6 Page model

The model for pages contains the following functions:

1. Get-index
   The page-index requests all available page-identifiers and page-tiles.

2. Get-page
   The get-page function requires a page-identifier parameter. The function then retrieves the page-identifier, name, access-level, title and content.

3. Get-page-content
   The page-content function only returns the page-content matching the entered page-identifier.

4. Get-page-meta
   The meta function returns the page-identifier, name, access-level and title of all pages.

The source can be found in appendix A.2.6.

8.1.7 User model

The user-model is specially designed for the user-controller, it contains one function called ‘login’. The login-function requires a user-email and password, then a database search is performed. This search looks for an active-user with the same credentials. If the user is found, then the function returns the user-id, active-status, e-mail address, first name, last name, password and access-level. The source can be found in appendix A.2.7.

8.1.8 Page views

The page-controller checks the existence of a file for the requested page within the page-views folder. If the page-view exists, then this view-file is loaded within the page-controller.
Index
The index-view places the content of the webpage in a content-div HTML element.

Contact
Contact is an example page-view containing a contact-form. The content of the other sample-pages is placed in the page-table.

Additional views
An expert can write additional page-views instead of using the database table content-field. The controller requires a database-record of the page to check the required access-level, but will then prefer a file-view before the database content-field.

The source can be found in appendix A.2.9.

8.1.9 File view
CodeIgniter is used with a rewrite-module, and this module forces files to be loaded through the framework. If this module is not active or compatible with the webserver, then the file-views can be used.

CSS
The CSS-view can be used for defining adaptive CSS.

JS
The CSS-view can be used for defining adaptive JavaScript.

The source can be found in appendix A.2.10.

8.1.10 Login view
The login-form is defined in the login-view. The form is based on Bootstrap form-design techniques.

The source can be found in appendix A.2.8.

8.1.11 Template view
The header and footer view are used in almost all webpages of the framework (proof of concept), since these are loaded by the page controller.

Header
The header-view defines the 'header'-part of an HTML document. The header-part contains meta-tags, CSS-references, Script, page-title, etc. This header-view also defines the first part of the body-tag, including the main-menu and content-header.
8.2 Adaptation layers

The adaptation layer defines three major concepts and one general:

1. General
   (a) Concepts
   (b) Variables
   (c) Relationships

2. Content Adaptation

3. Presentation Adaptation

4. Navigational Adaptation

8.2.1 General layer

The adaptation layer uses concepts. The main concept is the system itself. All other concepts can be children or siblings with a relationship. All variables are globally defined within the variable table of the adaptation database. The variable is bound to an id, concept and inheritance boolean. The id is a unique identifier for the variable. The concept bound to the variable is the main c.q. upper concept of this variable. All child-concepts will automatically have this variable available. The inheritance boolean tells the system to calculate the value of this variable based on relationships.

The system also contains a 'Variable Value' table, and this table is used for storing users variable values during the sessions. A unique session-id is used in order to store values for every client-server session. Each specific value can be stored with a weight, and this can be used in collaboration or calculation of specific variables. Parent-Child relationships are defined by the concept table, but in some cases other types of relations are needed between variables of concepts that do not apply to the whole concept. Within the relationship table, it is possible to bind a parent-concept-var to a child-concept-var with a specific weight. All relationships are taken into account if a variable is enabled for inheritance.

8.2.2 Content adaptation

All information is processed by the adaptation layer while processing a request of a web-client. The 'content adaptation' focuses on the main part of the response, containing the information about the requested page in the form of text and images. For example, the adaptation engine can decide to change, add, remove text or images.
8.2.3 Presentation adaptation

The users might use different types of web-browsers and devices to visit the web application. The user might get a bad user-experience if the interface does not change towards the user’s environment. The presentation adaptation helps defining the different user interfaces by using JavaScript, HTML and CSS in combination with Bootstrap. The presentation adaptation gives the ability to show, hide or change html elements in different user-specific conditions. The CSS and JavaScript files are dynamically generated by the presentation adaptation layer, giving the developer options to add dynamics. JavaScript can do asynchronous requests to the server and adapt the presentation without page refresh.

8.2.4 Navigation adaptation

The navigation adaptation covers all references to other pages, websites or functions referring to other content. These references could be buttons, textual hyperlinks or other objects adaptable within the HTML, CSS and JavaScript techniques. The objects can be reordered, shown, hidden, moved or presented differently based upon history or recommendation. The reordering optimizes the navigation the links and objects. Some references might or might not be required by the user and depending on the adaptation layer be shown or hidden. The location of the navigation could be more imported for a user still finding his/her way, than for a user who already found the required content. The visited-link history gives the user the ability to recognize references already visited.

8.3 Adaptation processor

The adaptation needs to be defined and recognized by the layer for processing. The adaptation layer is defined as a hook in the system, making it parse every single request, from the main HTML-page to an icon-image file. This set-up leads to a very scalable and interoperable system, and adaptation can be defined in every row, input-field or page on the server.

8.3.1 Commands or coding?

The adaptation is easily defined by the use of commands and tags. These tags are recognized and checked by regular expressions, starting with one global expression that fits all others. The global expression used:

```
/\{([^\\s]*)([^\\s]*)\}\{([^\\s]*)([^\\s]*)\}\{([^\\s]*)([^\\s]*)\}\{([^\\s]*)([^\\s]*)\}\{([^\\s]*)([^\\s]*)\}\{([^\\s]*)([^\\s]*)\}\{([^\\s]*)([^\\s]*)\}
```

This expression will be checked and looped until no matches are found anymore. The system processes the matches and replaces them with data or removes them. The expression-match is removed when the system is not able to process it.

For experts, it is possible to influence adaptation variables and functions. They can easily write PHP-code within the pages or files processed by the adaptation engine or even adapt the class file of the adaptation layer.
8.3.2 Overview of basic commands

1. Echo
   The echo-command replaces the command with the value of the requested variable.

2. Top
   The top-command contains multiple functions, all based on the order of variables within concepts.

3. Init
   The init-command defines the value of a variable and allows auto-increment.

4. Set
   The set-command set the value of a variable and overrides the current value if needed.

5. If
   The if-command, consisting of a start- and end-tag, allows the comparison of parameters and variables.

6. Hashtag
   The hashtag covers multiple commands, consisting of a start- and end-tag, this can, for example, be used to process commands and text after a specific amount of time.

These basic functions allow the novice to create simple adaptation functions. Different commands can be combined, and commands can be easily added by an expert.

Echo
The echo-command can write values of different variables including possible inherited relationships. The command looks like:

```
{ variableName }
```

In this case the value of 'VariableName' is requested within the currently active concept. The active concept can be set within the command by placing it in front of the variable:

```
{ conceptA$variableName }
```

Top
The top-command can write the name of the concept, a (global) variable value, the order of a variable within a concept and a menu. It starts with indicating the selection variable and the way of ordering (ascending or descending). The function within the command is specified after defining the way of ordering. For example: This command can be used to show a menu based on relationships within concepts. The concept-command looks like:

```
{ page$visited; max; concept:0/ }
```

This example returns the conceptName with highest visited value (order 0). The lowest value is returned if 'min' is used instead of 'max'.

The conceptVariable-command looks like:
This example returns the variable value of visit of the concept with the highest (order 0) visited value within the page concept. The second example shows the command to request a global variable with the same way of selecting.

The order-command looks like:

\{page$visited; max; order; page/home/\}

This example returns the order the 'page/home'-concept within the page concept based on the visited variable.

The menu-command looks like:

\{page$visited; max; menu; /\} or \{page$visited; max; menu; start: 1; max: 3; linkVar: link; linkNameVar: linkname/\}

This example returns a menu that starts with returning the concept with the highest value of the variable visited. It is also possible to define other variables that should be used for the links and the link-names. The second example shows a menu that skips the first item and stops at the third. Also specific variables for the link and link-name are used. These variables should be set globally and not in the user-model.

Init

The init-command declares a variable, sets the initial value and updates the variable with a specific value if the variable is already set. This increment function requires the variable to be set with a numeric value. The command looks like:

\{$variableName; init; 0; +1/\}

The '+1' value is not mandatory, if you only want to set the variable (if it is not already set) then you can leave this part out of the command. Also in this case the value of 'VariableName' is requested within the currently active concept. The active concept can be set within the command by placing it in front of the variable:

\{conceptA$variableName/\}

Set

The set-command allows to assign or re-assign a value to a variable. The command looks like:

\{$HomeIntro; set; "This is the introduction text"/\}
If
The if-command compares the value of a specific variable with a parameter or other variable. This commands consist out of two tags, one for opening and one for closing the IF-statement. The command looks like:

```html
{$variableName; < 1} .... {/ $variableName}
```

or

```html
{ page/home$visited; == page/contact$visited } .... { / $visited }
```

Hashtag
The hashtag command is designed to support multiple not variable related commands, such as 'fade;in', 'fade;out', etc. The commands looks like:

```html
{ #date; Hi; > 1230 } .... { / # date }
```

This is an example of an If-hashtag function checking comparing the actual time (hour + minutes) with a specific value.

8.3.3 Tags within tags
The commands are checked by regular-expressions, this can cause problems in case of using the same tags within each-other. It is possible to send an identifier with the tags to prevent problems with these constructions. An example is this partial CSS file:

```html
html {
    { #Hi; > 1759%1
        background: url(../images/bg_evening_miriadna.com.jpg) no-repeat center center fixed;
    }{/ #Hi%1
    { #Hi; < 2159%2
        background: url(../images/bg_evening_miriadna.com.jpg) no-repeat center center fixed;
    }{/ #Hi%2
}
```

This style sheet is the actual sheet used in the sample application, toggle the backgrounds at certain times of the day. The ‘% identifier’ creates a unique tag.
8.3.4 Concepts, variables and relations

The user- and domain-model are combined to prevent duplicate declarations of the structure and variables. Figure 9 shows the UML of the concept-, relation-, variable-model designed for use within the proof of concept. The domain-structure can be outlined in concepts, where the first concept will be the main-concept. Other concepts can be bound to the main or sub-concepts, this creates parent-child relationships. The names of the concept can be coupled with the URI (Uniform Resource Identifier). Table 4 shows an example concept structure. The variables are bound to a specific concept, and the conceptId and variableName is a unique combination within the data-structure. The value of a variable can be inherited by its relations with other variables (even from different concepts). The inheritance boolean tells the system whether the value should be calculated by using its relationships. This boolean is necessary because the parent-child relationship is set on the concepts instead of the variables. The relationship table describes the relations between different variables. The weight property indicates a level of importance during numeric calculation. The inheritance will be ignored if the value of a (child)variable is not numeric. Table 4 shows an example including a concept called ‘Home’, in this case ‘home’ might be an accessible page. If you want to calculate a variable value by inheritance then, it should be possible to use the variable value of the parent concept. The ‘parentValue’-boolean tells the system to use the parent value together with the child-relations during calculation. The decision to merge the user- and domain-model require an identifier for domain-values. The variable values are stored in a separate table. The ‘SessionId’ binds the user-session to the personal values. The system should recognize a global domain-specific value for the ‘SessionId’.

<table>
<thead>
<tr>
<th>Id</th>
<th>Parent</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Main</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Home</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Home/Welcome</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>Products</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>Product/Category1</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>Product/Category1/ProductA</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>Contact</td>
</tr>
</tbody>
</table>

Table 4: Concept Structure Example
8.3.5 Computation of variables values

The computation of the requested variable is handled by the concept-relation-variable class. This class-file contains all possible operations within the database structure, all predefined in functions and checked with regular-expressions.

The specific concept and variable are retrieved from the database during a value computation. The value can be requested for an active user-session or globally. The only change in case the global variable value is retrieved, is the sessionId value. This sessionId is then set to '0'. The computation of the value further depends on relationships with other variables.

There are four different computation methods:

1. 0 Default
2. 1 Weighted average
3. 2 Global + default
4. 3 Global + weighted average

The method is set by one simple parameter of the function that retrieves the value. The default computation method is calculated by taking the sum of all relations. The relations and the value of the variable itself are computed by multiplying the value and weight of the variable. In case of the second option, the score of the default option is divided by the sum of the weights leading to a weighted average. The second and third option allow the use of statistics or predefined global variables. Especially useful to prevent extreme outliers is the calculation of the scores, for example, when a new visitor is just clicking around.

8.3.6 Startup scripts

The adaptation layer supports commands to be executed on every page of the system. These commands can be written to the startup-column of the global adaptation settings. An example of a startup-script:

```
{ $visited; init:0;+1/ }
{ $visit; set:1/ }
```

This script initiates the 'visited' and then increases it with one on every page load. This variable is still stored within the active concept, for example, when you visit the 'contact'-page, the startup-script would write the visited variable of the 'contact'-concept.

8.3.7 Global cleanup

The global adaptation settings contain a cleanup boolean that toggles the system to automatically process old user-data. Alphabetical information will be disposed, and numeric values can be set to the global variable of the same concept. In this case, the global value becomes a storage place for the weighted average of this specific variable. The global values can be directly used in the calculation-methods. The global cleanup function also contains an expire-period, indicating the maximum session time in minutes. Zero or undefined values are set to 'unlimited'.

37
8.3.8 Log of all operations

All the operations executed by the adaptation layer are logged during the request and can be sent to the web-browser by toggling the 'log'-boolean within the global adaptation settings table. This log including all database errors, creates, updates, deletes and other adaptation-related messages.

8.4 Performance

<table>
<thead>
<tr>
<th>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>VPS</strong></td>
</tr>
<tr>
<td>1024 MB of RAM</td>
</tr>
<tr>
<td>1 CPU core</td>
</tr>
<tr>
<td>Apache web-server</td>
</tr>
<tr>
<td>CentOS 6.4 64-bits</td>
</tr>
<tr>
<td>Hosted at Versio</td>
</tr>
<tr>
<td><strong>Shared Hosting Environment</strong></td>
</tr>
<tr>
<td>Webhosting Small Package</td>
</tr>
<tr>
<td>Hosted at TransIP</td>
</tr>
<tr>
<td><strong>Dedicated Server</strong></td>
</tr>
<tr>
<td>16 GB of RAM</td>
</tr>
<tr>
<td>1 CPU Core</td>
</tr>
<tr>
<td>IIS (Internet Information Services)</td>
</tr>
<tr>
<td>Windows SBS 2008 64 bit</td>
</tr>
</tbody>
</table>

Table 5: Servers for measuring the performance

The performance depends on the number of computations and server resources. Table 5 shows the servers that were used to measure the performance of the framework. The adaptation layer is loaded after the framework or web-application is done loading. The performance of the adaptation layer can be measured by measuring the computation time. The commands described in 8.3.2 are measured separately. Caching can prevent database requests by the adaptation layer. The adaptation layer caches concept and variable information automatically. The cache is also updated after an update of a variable or concept.
9 Evaluation

9.1 Performance

The average computation time of the framework with two simple adaptation commands is 5 milliseconds. The computation time increased to 30ms, after increasing the number of commands to 30 in the global script. However, next to the server resources, the performance also depends on the size of the webpage. A webpage that contains more information, needs more time to process. The research indicated that 250 milliseconds could influence the visitor’s impression. The performance of the adaptation layer stays within this range and is therefore acceptable.

9.2 Limitations

The proof of concept is built with the use of a light-weighted PHP-framework. The structure of the test-site contained a limited number of items. The adaptation-layer is hooked on the output of the web application, this always causes an increase in loading-time before the response can be sent back to the web-browser (client). The adaptation processor uses regular-expressions to check adaptation commands within the content. Regular expressions have no native support for using the same tags within the same tags. Not supporting tags within tags could limit the adaptability of the webpage. An alternative solution is implemented, using ‘%id’ identifier in the commands to make them unique. Using an identifier is not an optimal solution, it is better to use a parser. A parser can keep track of opening and closing tags. This was not an option for this project because of time constraints.
10 Conclusion

The field of adaptive web-based systems is the future of modern websites. The adaptability creates a custom fit for every visitor, focusing on the interests, knowledge and other user-specific information to improve the experience and stimulate the goal of the website. It is certainly possible to define a framework for adaptable web-based systems. However, adaptivity requires processing of user behavior, causing additional server and/or client load (processing time). The extra processing layer leads to a decrease in performance, especially when the adaptation layer is a hook on the web application. It is essential to optimize processing and database requests to prevent unnecessary waiting times. User-data is stored during every visit. These user-specific records should be archived after expiration and can be converted to globally usable statistical data.

Interoperability and scalability can be easily achieved by the use of standardized, stable web-techniques. Vertical scalability is possible by expanding server-resources. Adding additional resources might be difficult in a shared server environment. A virtual private server is available in minutes in case of rapid growth of the number of users. The adaptation layer built for the proof of concept can be made suitable for horizontal scalability. The first step is to use a separate database-server. The framework is very interoperable through the use of open-source coding techniques. These techniques are supported by the majority of the web-hosts and commonly used in other web-applications such as a content management systems.
11 Future work

11.1 Future developments

The adaptation layer built as a proof of concept could be expanded or redesigned. The redesign part would replace the regular expressions with a parser as indicated in the limitation section. Within this project the layer is connected to CodeIgniter. In future development, other hooks could be written to test the framework with content management systems such as Wordpress or Joomla. The connection with a major CMS would give a good impression of the performance and ease of implementation. In some cases, web applications might not support hooks and have no server-side solutions for connecting the adaptation layer. In that case, the adaptation layer could be connected via the client’s web-browser. The response received from the webserver can be sent to a special adaptation server by the use of JavaScript. The adaptation can be done in multiple places and at multiple times. Some parts can be executed at the client-side or can be processed at a later stage. Better performance could be achieved by classifying the different adaptation rules based on where and when they can or need to be executed.

11.2 Future research

New technologies such as NodeJS, a JavaScript based client-and server-language could affect the design decisions to replace PHP. JavaScript is the client-side language for change HTML elements after page-loads. The combination of a JavaScript based server and client might improve the options to keep track of user behavior between the requests or with less intensive asynchronous connections.

The use of new style sheet (CSS) functionality could also influence design-decisions. The past few years, the CSS functionality has increased to support different styles real-time based on screen-sizes. A server-side script has trouble adapting to behavior only noticed by the browser. This behavior needs to be sent to the server, processed and returned before adaptation can take place. The CSS functionality could be predefined by the adaptation-layer and multiple scenarios could already be loaded with expanding CSS with other browser related information such as ‘reading-time’.


A Appendices

A.1 CodeIgniter basic installation settings

```php
<?php if (!defined('BASEPATH')) exit('No direct script access allowed');

/*
| Base Site URL
| URL to your CodeIgniter root. Typically this will be your base URL,
| WITH a trailing slash:
| http://example.com/
| If this is not set then CodeIgniter will guess the protocol, domain and
| path to your installation.
*/
$config['base_url'] = 'http://adaptive.tue.nl/';

/*
| Index File
| Typically this will be your index.php file, unless you've renamed it to
| something else. If you are using mod_rewrite to remove the page set this
| variable so that it is blank.
*/
$config['index_page'] = '';

/*
| URI PROTOCOL
| This item determines which server global should be used to retrieve the
| URI string. The default setting of 'AUTO' works for most servers.
| If your links do not seem to work, try one of the other delicious flavors:
| 'AUTO'        Default – auto detects
| 'PATH_INFO'   Uses the PATH_INFO
| 'QUERY_STRING' Uses the QUERY_STRING
| 'REQUEST_URI' Uses the REQUEST_URI
| 'ORIG_PATH_INFO' Uses the ORIG_PATH_INFO
*/
$config['uri_protocol'] = 'AUTO';

/*
| URL suffix
```
This option allows you to add a suffix to all URLs generated by CodeIgniter. For more information please see the user guide:

* http://codeigniter.com/user_guide/general/urls.html

```php
$conf['url_suffix'] = ''; /*

Default Language

This determines which set of language files should be used. Make sure there is an available translation if you intend to use something other than english.

*/
$conf['language'] = 'english'; /*

Default Character Set

This determines which character set is used by default in various methods that require a character set to be provided.

*/
$conf['charset'] = 'UTF-8'; /*

Enable/Disable System Hooks

If you would like to use the 'hooks' feature you must enable it by setting this variable to TRUE (boolean). See the user guide for details.

*/
$conf['enable_hooks'] = TRUE; /*

Class Extension Prefix

This item allows you to set the filename/classname prefix when extending native libraries. For more information please see the user guide:

* http://codeigniter.com/user_guide/general/core_classes.html
* http://codeigniter.com/user_guide/general/creating_libraries.html

*/
$conf['subclass_prefix'] = 'AF_';
/*

Allowed URL Characters

This lets you specify with a regular expression which characters are permitted
within your URLs. When someone tries to submit a URL with disallowed
characters they will get a warning message.

As a security measure you are STRONGLY encouraged to restrict URLs to
as few characters as possible. By default only these are allowed: a−z
0−9−%.:_−

Leave blank to allow all characters — but only if you are insane.

DO NOT CHANGE THIS UNLESS YOU FULLY UNDERSTAND THE REPERCUSSIONS!!

*/
$conf['permitted_uri_chars'] = 'a−z 0−9−%.:_−';

/

Enable Query Strings

By default CodeIgniter uses search−engine friendly segment based URLs:
example.com/who/what/where/

By default CodeIgniter enables access to the $_GET array. If for some
reason you would like to disable it, set ‘allow_get_array’ to FALSE.

You can optionally enable standard query string based URLs:
example.com?who=me&what=something&where=here

Options are: TRUE or FALSE (boolean)

The other items let you set the query string ‘words’ that will
invoke your controllers and its functions:
example.com/index.php?c=controller&m=function

Please note that some of the helpers won’t work as expected when
this feature is enabled, since CodeIgniter is designed primarily to
use segment based URLs.

*/
$conf['allow_get_array'] = TRUE;
$conf['enable_query_strings'] = false;
$conf['controller_trigger'] = 'c';
$conf['function_trigger'] = 'm';
$conf['directory_trigger'] = 'd'; // experimental not currently in use

/

Error Logging Threshold
If you have enabled error logging, you can set an error threshold to determine what gets logged. Threshold options are:
You can enable error logging by setting a threshold over zero. The threshold determines what gets logged. Threshold options are:

- 0 = Disables logging. Error logging TURNED OFF
- 1 = Error Messages (including PHP errors)
- 2 = Debug Messages
- 3 = Informational Messages
- 4 = All Messages

For a live site you’ll usually only enable Errors (1) to be logged otherwise your log files will fill up very fast.

```php
$cfg['log_threshold'] = 4;
```

Error Logging Directory Path

Leave this BLANK unless you would like to set something other than the default application/logs/ folder. Use a full server path with trailing slash.

```php
$cfg['log_path'] = '';```

Date Format for Logs

Each item that is logged has an associated date. You can use PHP date codes to set your own date formatting

```php
$cfg['log_dateformat'] = 'Y-m-d H:i:s';
```

Cache Directory Path

Leave this BLANK unless you would like to set something other than the default system/cache/ folder. Use a full server path with trailing slash.

```php
$cfg['cache_path'] = '';
```

Encryption Key
If you use the Encryption class or the Session class you MUST set an encryption key. See the user guide for info.

```php
$conf['encryption_key'] = 'AFVersleuteling20131215s%';
```

### Session Variables

- `sess_cookie_name` = the name you want for the cookie
- `sess_expiration` = the number of SECONDS you want the session to last.
  
  by default sessions last 7200 seconds (two hours). Set to zero for no expiration.
- `sess_expire_on_close` = Whether to cause the session to expire automatically when the browser window is closed
- `sess_encrypt_cookie` = Whether to encrypt the cookie
- `sess_use_database` = Whether to save the session data to a database
- `sess_table_name` = The name of the session database table
- `sess_match_ip` = Whether to match the user's IP address when reading the session data
- `sess_match_useragent` = Whether to match the User Agent when reading the session data
- `sess_time_to_update` = how many seconds between CI refresh Session Information

```php
$conf['sess_cookie_name'] = 'af_session';
$conf['sess_expiration'] = 0;
$conf['sess_expire_on_close'] = FALSE;
$conf['sess_encrypt_cookie'] = TRUE;
$conf['sess_use_database'] = TRUE;
$conf['sess_table_name'] = 'ci_sessions';
$conf['sess_match_ip'] = TRUE;
$conf['sess_match_useragent'] = FALSE;
$conf['sess_time_to_update'] = 7200;
```

### Cookie Related Variables

- `cookie_prefix` = Set a prefix if you need to avoid collisions
- `cookie_domain` = Set to .your-domain.com for site-wide cookies
- `cookie_path` = Typically will be a forward slash
- `cookie_secure` = Cookies will only be set if a secure HTTPS connection exists.

```php
$conf['cookie_prefix'] = "";
$conf['cookie_domain'] = "adaptive.tuer.nl";
$conf['cookie_path'] = "/";
$conf['cookie_secure'] = FALSE;
```
Global XSS Filtering

Determines whether the XSS filter is always active when GET, POST or COOKIE data is encountered

```bash
$config["global_xss_filtering"] = true;
```

Cross Site Request Forgery

Enables a CSRF cookie token to be set. When set to TRUE, token will be checked on a submitted form. If you are accepting user data, it is strongly recommended CSRF protection be enabled.

```bash
$config["csrf_protection"] = false;
$config["csrf_token_name"] = "csrf_af_token";
$config["csrf_cookie_name"] = "csrf_af_name";
$config["csrf_expire"] = 7200;
```

Output Compression

Enables Gzip output compression for faster page loads. When enabled, the output class will test whether your server supports Gzip. Even if it does, however, not all browsers support compression so enable only if you are reasonably sure your visitors can handle it.

VERY IMPORTANT: If you are getting a blank page when compression is enabled it means you are prematurely outputting something to your browser. It could be a line of whitespace at the end of one of your scripts. For compression to work, nothing can be sent before the output buffer is called by the output class. Do not ‘echo’ any values with compression enabled.

```bash
$config["compress_output"] = FALSE;
```

Master Time Reference

Options are ‘local’ or ‘gmt’. This pref tells the system whether to use your server’s local time as the master ‘now’ reference, or convert it to GMT. See the ‘date helper’ page of the user guide for information.
regarding date handling.

`$config['time_reference'] = 'UP1';`

Rewrite PHP Short Tags

If your PHP installation does not have short tag support enabled CI can rewrite the tags on-the-fly, enabling you to utilize that syntax in your view files. Options are TRUE or FALSE (boolean)

`$config['rewrite_short_tags'] = FALSE;`

Reverse Proxy IPs

If your server is behind a reverse proxy, you must whitelist the proxy IP addresses from which CodeIgniter should trust the HTTP_X_FORWARDED_FOR header in order to properly identify the visitor’s IP address. Comma-delimited, e.g. ‘10.0.1.200,10.0.1.201’

`$config['proxy_ips'] = '';`

End of file config.php

Location: ./application/config/config.php
<?php if (!defined('BASEPATH')) exit('No direct script access allowed');

/*

DATABASE CONNECTIVITY SETTINGS

This file will contain the settings needed to access your database.

For complete instructions please consult the 'Database Connection' page of the User Guide.

*/

EXPLANATION OF VARIABLES

[
'hostname'] The hostname of your database server.

['username'] The username used to connect to the database

['password'] The password used to connect to the database

['database'] The name of the database you want to connect to

['dbdriver'] The database type. I.e: mysql. Currently supported: mysql, mysqli, postgres, odbc, mssql, sqlite, oci8

['dbprefix'] You can add an optional prefix, which will be added to the table name when using the Active Record class

['pconnect'] TRUE/FALSE – Whether to use a persistent connection

['db_debug'] TRUE/FALSE – Whether database errors should be displayed.

['cache_on'] TRUE/FALSE – Enables/disables query caching

['cachedir'] The path to the folder where cache files should be stored

['char_set'] The character set used in communicating with the database

['dbcollat'] The character collation used in communicating with the database

NOTE: For MySQL and MySQLi databases, this setting is only used as a backup if your server is running PHP < 5.2.3 or MySQL < 5.0.7 (and in table creation queries made with DB Forge).

There is an incompatibility in PHP with mysql_real_escape_string() which can make your site vulnerable to SQL injection if you are using a multi-byte character set and are running versions lower than these. Sites using Latin-1 or UTF-8 database character set and collation are unaffected.

['swap_pre'] A default table prefix that should be swapped with the dbprefix

['autounit'] Whether or not to automatically initialize the database.

['stricton'] TRUE/FALSE – forces 'Strict Mode' connections – good for ensuring strict SQL while developing

The $active_group variable lets you choose which connection group to make active. By default there is only one group (the 'default' group).

The $active_record variables lets you determine whether or not to load the active record class

$active_group = 'default';
$active_record = TRUE;

$db['default']['hostname'] = 'localhost';
$db['default']['username'] = 'adaptive.tuer.nl';
$db['default']['password'] = '1338mysql@RS';
$db['default']['database'] = 'adaptive.tuer.nl';
$db['default']['dbdriver'] = 'mysql';
A.2 CodeIgniter controllers

A.2.1 Page controller

```php
<?php if (!defined('BASEPATH')) exit('No direct script access allowed');

class Page extends CI_Controller {

  function __construct() {
    parent::__construct();
    $this->load->model('menu_model');
    $this->load->model('pages');
    $this->load->helper('form');
    $this->load->library('af_redirect');
  }

  function view($page='home') {
    $current_session = $this->session->userdata('user');
    if (strlen($page)>0) {
      $data['page_meta'] = $this->pages->get_page_meta($page);
      if (count($data['page_meta'])>0) {
        if (($data['page_meta'][0]->acl_id <= $current_session['user_acl'] ||
          $data['page_meta'][0]->acl_id===0)) {
          // load menu
          $data['menu'] = $this->menu_model->get_menu();
          // title
          $data['title'] = ucfirst($data['page_meta'][0]->page_title);
          // header
          $this->load->view('templates/header', $data);
          // get adaptation data
          $requested_page = uri_string();
          // body
          if (file_exists('application/views/page/'.$page.'.php')) {
            $this->load->view('page/'.$page, $data);
          } else {
          }
        }
      }
    }
  }
}
```
```php
$data['page_content'] = $this->pages->get_page($data['page_meta'][0]['page_id']);

// if there is a page, load the view
$this->load->view('page/index', $data);

// footer
$this->load->view('templates/footer', $data);

// if there is a redirect, return it
$this->af_redirect->to('/errors/403');

// else, if there is a redirect, return it
$this->af_redirect->to('/errors/404');
```

`sources/ci_c_page.php`
A.2.2 User controller

```php
<?php if ( ! defined('BASEPATH')) exit('No direct script access allowed');

class User extends CI_Controller {
    function __construct(){
        parent::__construct();
        $this->load->model('users', '', TRUE);
        $this->load->model('menu_model');
        $this->load->library('form_validation');
        $this->load->helper(array('form'));
    }

    function login($redirect="zoeken"){
        // load menu
        $data['menu'] = $this->menu_model->get_menu();

        // This method will have the credentials validation
        if (isset($_POST['email'])){
            $this->form_validation->set_rules('email', 'Email', 'trim|required|xss_clean');
            if ($this->form_validation->run() == true){
                $this->form_validation->set_rules('password', 'Password', 'trim|required|xss_clean|sha1|callback_check_database');
            } else {
                redirect($redirect, 'refresh');
            }
        } else {
            $data['title'] = "Login"; // Capitalize the first letter
            $data['redirect'] = $redirect;
            $this->load->view('templates/header', $data);
            $this->load->view('user/login_view', $data);
            $this->load->view('templates/footer', $data);
        }
    }

    function logout(){
        session->unset_userdata('user');
        redirect('/', 'refresh');
    }

    function index(){
        $data['title'] = "User"; // Capitalize the first letter
        // $this->load->view('templates/header', $data);
        // $this->load->view('login_view', $data);
        // $this->load->view('templates/footer', $data);
    }

    function check_database($password){
        // Field validation succeeded. Validate against database
        $user_email = $this->input->post('email');
    }
```
// query the database
$result = $this->users->login($user_email, $password);

if ($result)
    {$sess_array = array();
        foreach ($result as $row)
        {$sess_array = array(
            'user_id' => $row->user_id,
            'user_active' => $row->user_active,
            'user_email' => $row->user_email,
            'user_acl' => $row->acl_id,
            'user_fname' => $row->user_fname,
            'user lname' => $row->user_lname
        );
        $this->session->set_userdata('user', $sess_array);
    } return true;
else{
    $this->form_validation->set_message('check_database', 'Invalid username or password');
    return false;
}
}
A.2.3 File controller

```php
<?php if ( ! defined('BASEPATH')) exit('No direct script access allowed');

class File extends CI_Controller {
    function __construct(){
        parent::__construct();
    }
    function view($file){
        if (file_exists('application/views/file/' . $file . '.php')){
            $this->load->view('file/' . $file);
        }
    }
}
?>
```

`sources/ci_files.php`
A.2.4 Admin controller

```php
<?php
if (!defined('BASEPATH')) exit('No direct script access allowed');

class Admin extends CI_Controller {
    var $current_session;
    function __construct() {
        parent::__construct();
        // $this->output->enable_profiler();
        $this->current_session = $this->session->userdata('user');
        if ($this->current_session['user_acl']<2) {
            redirect('user/login/admin', 'refresh');
        }
        // load crud library
        $this->load->library('grocery_CRUD');
        $this->load->model('menu_model');
    }

    public function users() {
        try {
            $crud = new grocery_CRUD();
            $crud->set_table('users');
            $crud->set_subject('Users');
            $crud->display_as('acl_id', 'Access Level');
            $crud->set_relation('acl_id', 'acls', 'acl_label');
            $crud->change_field_type('user_password', 'password');
            $crud->callback_edit_field('user_password', array($this, 'set_password_field_empty'));
            $crud->callback_add_field('user_password', array($this, 'set_password_field_empty'));
            $crud->callback_before_insert(array($this, 'encrypt_password_callback'));
            $crud->callback_before_update(array($this, 'encrypt_password_callback'));
            $output = $crud->render();
            $this->crud_output($output);
        } catch (Exception $e) {
            show_error($e->getMessage() . ' ' . $e->getTraceAsString());
        }
    }

    public function pages() {
        try {
            $crud = new grocery_CRUD();
            $crud->set_table('pages');
            $crud->set_subject('Page');
            $crud->required_fields('page_name', 'acl_id', 'page_title');
            $crud->columns('page_name', 'acl_id', 'page_title', 'page_text');
            // $crud->display_as('acl_id', 'Access Level');
            $crud->set_relation('acl_id', 'acls', 'acl_label');
            $output = $crud->render();
            $this->crud_output($output);
        }
```
public function menus()
{
  try{
    $crud = new grocery_CRUD();
    $crud->set_table('menu');
    $crud->set_subject('Menus');
    $crud->display_as('acl_id','Access Level');
    $crud->set_relation('acl_id','acls','acl_label');
    $crud->set_relation('parent_id','menu','name');
    $crud->set_relation('menu_id','menus','name');
    $output = $crud->render();
    $this->crud_output($output);
  }
  catch( Exception $e){
    show_error($e->getMessage() . ' --- ' . $e->getTraceAsString());
  }
}

public function a_settings()
{
  try{
    $crud = new grocery_CRUD();
    $crud->set_table('a_globals');
    $crud->set_subject('Adaptation Global Script');
    $crud->unset_add();
    $crud->unset_delete();
    $crud->unset_texteditor('startup');
    $output = $crud->render();
    $this->crud_output($output);
  }
  catch( Exception $e){
    show_error($e->getMessage() . ' --- ' . $e->getTraceAsString());
  }
}

public function a_concepts()
{
  try{
    $crud = new grocery_CRUD();
    $crud->set_table('a_concepts');
    $crud->set_subject('Adaptation Concepts');
    $crud->set_relation('concept_parent_id','a_concepts','concept_name');
    $output = $crud->render();
    $this->crud_output($output);
  }
  catch( Exception $e){
    show_error($e->getMessage() . ' --- ' . $e->getTraceAsString());
  }
}
```php
public function a_variables() {
    try {
        $crud = new grocery_CRUD();
        $crud->set_table('a_vars');
        $crud->set_subject('Adaptation Variables');
        $crud->set_relation('concept_id', 'a_concepts', 'concept_name');
        $output = $crud->render();
        $this->crud_output($output);
    } catch (Exception $e) {
        show_error($e->getMessage() . '---'. $e->getTraceAsString());
    }
}
public function a_variable_values() {
    try {
        $crud = new grocery_CRUD();
        $crud->set_table('a_vars_values');
        $crud->set_subject('Adaptation Variables Values');
        $output = $crud->render();
        $this->crud_output($output);
    } catch (Exception $e) {
        show_error($e->getMessage() . '---'. $e->getTraceAsString());
    }
}
public function a_relationships() {
    try {
        $crud = new grocery_CRUD();
        $crud->set_table('a_relationships');
        $crud->set_subject('Adaptation Relationships');
        $crud->set_relation('parent_concept_id', 'a_concepts', 'concept_name');
        $crud->set_relation('child_concept_id', 'a_concepts', 'concept_name');
        $output = $crud->render();
        $this->crud_output($output);
    } catch (Exception $e) {
        show_error($e->getMessage() . '---'. $e->getTraceAsString());
    }
}

//--------------------------------------------------
public function mc_test() {
    try {
        $crud = new grocery_CRUD();
        $crud->set_table('mc_test');
        $crud->set_subject('MC tests');
        $crud->set_relation_n_n('questions', 'mc_test_question', 'mc_question', 'question_id', 'test_id', 'question');
    }
```
$output = $crud->render();
$this->crud_output($output);

} catch (Exception $e) {
    show_error($e->getMessage() . '---'. $e->getTraceAsString());
}

public function mc_question()
{
    try {
        $crud = new grocery_CRUD();
        $crud->set_table('mc_question');
        $crud->set_subject('MC questions');
        $crud->set_relation('test_id', 'mc_test', 'test_name');
        $output = $crud->render();
        $this->crud_output($output);
    } catch (Exception $e) {
        show_error($e->getMessage() . '---'. $e->getTraceAsString());
    }
}

public function mc_answer()
{
    try {
        $crud = new grocery_CRUD();
        $crud->set_table('mc_answer');
        $crud->set_subject('MC answer');
        $crud->set_relation('question_id', 'mc_question', 'question');
        $crud->set_relation('correct_id', 'mc_correct', 'correct_value');
        $output = $crud->render();
        $this->crud_output($output);
    } catch (Exception $e) {
        show_error($e->getMessage() . '---'. $e->getTraceAsString());
    }
}

public function _crud_output($output = null)
{
    $data['crud'] = $output;
    $data['title'] = "Admin - CRUD";
    $data['menu'] = $this->menu_model->get_admin_menu($this->current_session['user_acl']);
    $this->load->view('templates/header_crud', $data);
    $this->load->view('crud.php', $data);
    $this->load->view('templates/footer', $data);
}

function index()
{
    $data['title'] = "Admin"; // Capitalize the first letter
    $data['menu'] = $this->menu_model->get_admin_menu($this->current_session['user_acl']);
    $data['user'] = $this->current_session;
```php
$this->load->view('templates/header_crud', $data);
$this->load->view('admin/index', $data);
$this->load->view('templates/footer', $data);
}

function encrypt_password_callback($post_array) {
    $post_array['user_password'] = sha1($post_array['user_password']);
    return $post_array;
}

function set_password_field_empty() {
    return "<input type='password' name='user_password' value=''/>";
}
```
A.2.5 Menu model

```php
<?php
class menu_model extends CI_Model {
    public function __construct() {
        $this->load->database();
    }
    public function get_menu($acl = 0) {
        $sql = 'SELECT menu.order, menu.id, menu.parent_id, menu.menu_id, menu.name, menu.link, menu.position, menu.acl_id FROM menu WHERE acl_id <= ? AND menu_id=1 ORDER BY menu.order ASC';
        $this->db->cache_on();
        $query = $this->db->query($sql, array($acl));
        return $query->result_array();
    }
    public function get_admin_menu($acl = 0) {
        $sql = 'SELECT menu.order, menu.id, menu.parent_id, menu.menu_id, menu.name, menu.link, menu.position, menu.acl_id FROM menu WHERE acl_id <= ? AND menu_id=2 ORDER BY menu.order ASC';
        $this->db->cache_on();
        $query = $this->db->query($sql, array($acl));
        return $query->result_array();
    }
}
```

sources/ci_m_menu.php
A.2.6 Page model

```php
<?php
Class Pages extends CI_Model
{
    function get_index()
    {
        $this->db->select('page_id, page_title');
        $this->db->from('pages');
        $query = $this->db->get();
        return $query->result();
    }

    function get_page($page_id)
    {
        $this->db->select('page_id, page_name, acl_id, page_title, page_text');
        $this->db->from('pages');
        $this->db->where('page_id', $page_id);
        $query = $this->db->get();
        return $query->result();
    }

    function get_page_content($page_id)
    {
        $this->db->select('page_text');
        $this->db->from('pages');
        $this->db->where('page_id', $page_id);
        $query = $this->db->get();
        return $query->result();
    }

    function get_page_meta($page_name)
    {
        $this->db->select('page_id, page_name, acl_id, page_title');
        $this->db->from('pages');
        $this->db->where('page_name', $page_name);
        $query = $this->db->get();
        return $query->result();
    }
}
?>
```

`sources/ci_m_pages.php`
A.2.7 User model

```php
<?php

Class Users extends CI_Model
{
    function login($user_email, $user_password)
    {
        $this -> db -> select('user_id, user_active, user_email, user_fname, user_lname, user_password, acl_id');
        $this -> db -> from('users');
        $this -> db -> where('user_email', $user_email);
        $this -> db -> where('user_password', $user_password);
        $this -> db -> where('acl_id >', 1);
        $this -> db -> where('user_active', 1);
        $this -> db -> limit(1);
        $query = $this -> db -> get();
        if($query -> num_rows() == 1)
        {
            return $query -> result();
        }
        else
        {
            return false;
        }
    }
}
?>
```

sources/ci_m_users.php
A.2.8 Login view

```php
<?
if(strlen($this->validation_errors()) > 0) {
}

<form class="form-horizontal" action="/user/login/" method="post">
<?php echo $this->security->get_csrf_token_name() . ' ' . $this->security->get_csrf_hash() . ' '; ?>
<fieldset>

<!-- Text input -->
<div class="form-group">
    <label class="col-md-4 control-label" for="email">E-mail</label>
    <div class="col-md-6">
        <input id="email" name="email" type="text" placeholder="E-mail" class="form-control input-md" required=""/>
    </div>
</div>

<!-- Password input -->
<div class="form-group">
    <label class="col-md-4 control-label" for="password">Password</label>
    <div class="col-md-6">
        <input id="password" name="password" type="password" placeholder="Password" class="form-control input-md" required=""/>
    </div>
</div>

<!-- Button -->
<div class="form-group">
    <label class="col-md-4 control-label" for="submit"></label>
    <button id="submit" name="submit" class="btn btn-primary">Login</button>
</div>

</fieldset>
</form>
```

sources/ci_v_login.php
A.2.9 Page views

```php
<?php if ( ! defined('BASEPATH')) exit('No direct script access allowed');?
$page_text = $page_content[0]->page_text;

/sources/ci_v_pages_index.php

<?php if ( ! defined('BASEPATH')) exit('No direct script access allowed');?
if (strlen(validation_errors())>0){
  <div class="alert alert-dismissable alert-warning">
    <?php echo validation_errors(); ?></p><?php }
  $attributes = array('class' => 'form-horizontal');
  echo form_open('page/contact', $attributes);
}

/fieldset>
/field-- Text input-->
<div class="form-group">
  <label class="col-md-4 control-label" for="name">Name</label>
  <input id="name" name="name" type="text" placeholder="Piet Puk" class="form-control input-md" required="">
</div>

/field-- Text input-->
<div class="form-group">
  <label class="col-md-4 control-label" for="email">E-mail</label>
  <input id="email" name="email" type="text" placeholder="piet.puk@domein.nl" class="form-control input-md" required="">
</div>

/field-- Textarea -->
<div class="form-group">
  <label class="col-md-4 control-label" for="message">Message</label>
  <textarea class="form-control" id="message" name="message"></textarea>
</div>

/field-- Button -->
<div class="form-group">
  <button id="send" name="send" class="btn btn-primary">Send</button>
</div>

/fieldset>
</form>
```

/sources/ci_v_pages_contact.php
A.2.10 File view

```php
<?php
header("Content-type: text/css");
?>

div.alert{
    top: 50px;
    /* z-index: 99999; */
    position: absolute;
    width: 100%;
}

html {
    #date:Hi;>1759%1
    {#date:Hi;<2159}
        background: url(../images/bg_evening_miriadna_com.jpg) no-repeat center
        center fixed;
    {/#date}
    {/#date%1}

    {#date:Hi;>2200%2}
    background: url(../images/bg_night_wallpapertoon_com.jpg) no-repeat center
    center fixed;

    {#date}
    {/#date%2}

    {#date:Hi;>0%1}
    {#date:Hi;<700}
        background: url(../images/bg_night_wallpapertoon_com.jpg) no-repeat center
        center fixed;
    {/#date}
    {/#date%1}

    {#date:Hi;>1159%1}
    {#date:Hi;<1800}
        background: url(../images/bg_midday_fc09_devianart_net.jpg) no-repeat center
        center fixed;
    {/#date}
    {/#date%1}

    {#date:Hi;>659%3}
    {#date:Hi;<1200}
        background: url(../images/bg_morning_personal_psu_edu.jpg) no-repeat center
        center fixed;
    {/#date}
    {/#date%3}
        background-size: cover;
    -webkit-background-size: cover;
    -o-background-size: cover;
    background-size: cover;
}

body{
    /*background-color: rgba(255,255,255,.7);*/
    background-color: transparent;
}
```
```html
#main-container {
  border-radius: 5px;
  background-color: rgba(245, 245, 245, 0.8);
}

#footer p a {
  background-color: transparent;
  color: white;
}

.navbar-custom {
  background-color: #e3e3e3;
  border-color: #d2d2d2;
  background-image: -webkit-linear-gradient(top, #f7f7f7, 0%, #e3e3e3 100%);
  background-image: -moz-linear-gradient(top, #e3e3e3 0%, #f7f7f7 100%);
  background-image: linear-gradient(to bottom, #f7f7f7 0%, #e3e3e3 100%);
  background-repeat: repeat-x;
  filter: progid:DXImageTransform.Microsoft.gradient(startColorstr='#f7f7f7', endColorstr='#e3e3e3', GradientType=0);
}

.navbar-custom .navbar-brand {
  color: #3b3b3b;
}

.navbar-custom .navbar-brand:hover, .navbar-custom .navbar-brand:focus {
  color: #222222;
  background-color: transparent;
}

.navbar-custom .navbar-text {
  color: #3b3b3b;
}

.navbar-custom .navbar-nav > li:last-child > a {
  border-right: 1px solid #d2d2d2;
}

.navbar-custom .navbar-nav > li > a {
  color: #3b3b3b;
  border-left: 1px solid #d2d2d2;
}

.navbar-custom .navbar-nav > li > a:hover, .navbar-custom .navbar-nav > li > a:focus {
  color: #000000;
  background-color: transparent;
}

.navbar-custom .navbar-nav > .active > a, .navbar-custom .navbar-nav > .active > a:hover, .navbar-custom .navbar-nav > .active > a:focus {
  color: #000000;
  background-color: #d2d2d2;
  background-image: -webkit-linear-gradient(left, #d2d2d2 0%, #e3e3e3 100%);
  background-image: -moz-linear-gradient(left, #e3e3e3 0%, #d2d2d2 100%);
  background-image: linear-gradient(to right, #d2d2d2 0%, #e3e3e3 100%);
  filter: progid:DXImageTransform.Microsoft.gradient(startColorstr='#d2d2d2', endColorstr='#e3e3e3', GradientType=0);
}
```

```php
<?php
if (!defined('BASEPATH')) exit('No direct script access allowed');
header("content-type: application/javascript");

$(document).delegate('#field-a_global_script', 'keydown', function(e) {
    var keyCode = e.keyCode || e.which;
    if (keyCode == 9) {
        e.preventDefault();
        var start = $(this).get(0).selectionStart;
        var end = $(this).get(0).selectionEnd;

        // set textarea value to: text before caret + tab + text after caret
        $(this).val($(this).val().substring(0, start)
            + "\t"
            + $(this).val().substring(end));

        // put caret at right position again
        $(this).get(0).selectionStart =
        $(this).get(0).selectionEnd = start + 1;
    }
});
```

```
```
A.2.11 Template views

```php
<?
//menu function
function build_menu_parents($menu, $position, &$menu_parents = array()){
    if(isset($menu)){
        //get all parents and unset wrong position
        foreach($menu as $key=>$menu_item){
            if($menu_item['position']==$position){
                if(isset($menu_item['parent_id'])){
                    $menu_parents[$menu_item['parent_id']] = 1;
                }
            } else{
                unset($menu[$key]);
            }
        }
    }
    return $menu;
}
function build_menu_childs(&$menu, $parent_id, $menu_item_index){
    $result = "";
    $childs = array();
    foreach($menu as $key=>$menu_item){
        if($parent_id == $menu_item['parent_id']){
            $childs[$menu_item['order']].$menu_item['id'] = $menu_item;
            unset($menu[$key]);
        }
    }
    ksort($childs);
    foreach($childs as $key=>$child){
        $active = (($_SERVER['REQUEST_URI'] == $child['link']) ? 'class="active" ' : '');
        $result .= "<li id="". $menu_item['menu_id'] ."". $menu_item_index."">". $active."<a href="". $child['link'] ."">". $child['name'] ."".</a></li>";
        $menu_item_index++;}
    return $result;
}
function build_menu($menu, $position){
    //menu item index
    $menu_item_index = (isset($menu_item_index) ? $menu_item_index : 0);
    $result = "";
    $menu_parents = array();
    $menu = build_menu_parents($menu, $position, $menu_parents);
    if(isset($menu)){
        //get all root menu items
        sort($menu);
        $key = 0;
        while(count($menu)){
            if(isset($menu[$key])){
                $menu_item = $menu[$key];
                //get correct position
                if(isset($menu_parents[$menu_item['id']])){
                    $result .= " <li class="dropdown">
```
```php
$active = (($_SERVER['REQUEST_URI'] == $menu_item['link']) ? 'class="active"' : '');
$result .= "<li id="" . 'menu_' . $menu_item['menu_id'] . ">" . $menu_item['name'] . "</li>">
unset($menu_item_index);
}<a href="#" class="dropdown-toggle" data-toggle="dropdown">" .
$menu_item['name'] . "</a>
</ul>
</li>

else if (!$menu_item['parent_id']) {
    // child
    $active = (($_SERVER['REQUEST_URI'] == $menu_item['link']) ? 'class="active"' : '');
    $result .= "<li id="" . 'menu_' . $menu_item['menu_id'] . ">" . $menu_item['name'] . "</li>">
unset($menu_item_index);
}
$key++;}

return $result;

<!--[css Scripts]-->
if(isset($css_extra)) {
    foreach ($css_extra as $value) {
        <link rel="stylesheet" type="text/css" href="<?=$value?>" media="screen"/>
    }
}
// <script src="/js/jquery-ui-1.10.4.custom.js"></script>
<link rel="stylesheet" type="text/css" href="/file.css" media="screen"/>
<title><?php echo $title ?></title>
<script src="/js/jquery-2.1.0.min.js"></script>
<script src="/js/boostrap.min.js"></script>
<script src="/js/file.js"></script>
```
<? //-------------------------------[Extra Scripts]
if(!isset($scripts_extra)){
    foreach($scripts_extra as $value) {
        <script src="?=$value"></script>
    }
}

<script>
}</script>

<style>
}{css/}
</style>
</head>
<body>
<div id="wrap">
<!-- Fixed navbar -->
<div class="navbar navbar-custom navbar-fixed-top" role="navigation">
    <div class="container">
        <div class="navbar-header">
            <button type="button" class="navbar-toggle data-toggle="collapse" data-target="#navbar-collapse">
                <span class="sr-only">Toggle navigation</span>
            </button>
            <a class="navbar-brand" href="/page/home">Adaptive Framework</a>
        </div>
        <ul class="nav navbar-nav navbar-left">
            <? if(isset($menu)) {
                echo build_menu($menu, 'left');
            } ?>
        </ul>
        <ul class="nav navbar-nav navbar-right">
            <? if(isset($menu)) {
                echo build_menu($menu, 'right');
            } ?>
        </ul>
    </div>
</div>
<!-- /.nav-collapse -->
</div>
<div id="main-container">
<h1><?=$title?></h1>
<![if !isset($breadcrumbs)]><p></p><![endif]>

$breadcrumbs = explode('/',$_SERVER['REQUEST_URI']);
$breadcrumbs_prev = "";
$bc_result = "";
foreach ($breadcrumbs as $key=>$item)
{
    if (strlen($item))
    {
        $active = (((count($breadcrumbs)-1) == $key) ? ' class="active" ' : '')
        ;
        if ($item == 'page')
        {
            $item = 'home';
            $bc_result .= "<li><a $active href='./">ucfirst($item)."</a></li>";
            $breadcrumbs_prev .= " /page";
        } elseif ($item != 'home')
        {
            $bc_result .= "<li><a $active href='$breadcrumbs_prev/".$item.'">ucfirst($item)."</a></li>";
            $breadcrumbs_prev .= " /".$item;
        }
    }
}
if (strlen($bc_result) && $_SERVER[ 'REQUEST_URI' ] != '/page/home')
{
    echo "<ol class="breadcrumb">";
    echo $bc_result;
    echo "</ol>";
}
?>

sources/ci_v_template_header.php

</div>
</div>
<div id="footer">
  <div class="container">
    <p class="muted credit">
    <?php
    if (isset($menu))
    {
        foreach ($menu as $menu_item)
        {
            // left?
            if ($menu_item['position'] == 'footer')
            {
                ?><a href="<$=menu_item['link']?"?"<$=menu_item['name']?>>"<?=
                &nbsp;&nbsp;&nbsp;
                ?><?]
            }
        }
    }?
    </p>
  </div>
</div>
</body>
</html>

sources/ci_v_template_footer.php
A.3 Codelgniter framework hook

```php
<?php
if (!defined('BASEPATH')) exit('No direct script access allowed');

/*****************************************************************
| Adaptation Layer
| Author: Brouwer, S.R.J.
|*****************************************************************/

// Adaptation Class extends the framework to process output.
class Adaptation extends CI_Controller {
    // output of complete request
    public $output;

    function __construct() {
        parent::__construct();
        // connect to framework
        $this->CI =& get_instance();
        // get current page output
        $this->output = $this->CI->output->get_output();
    }

    function Display() {
        require_once('adapt.class.php');
        $adapt = new Adapt($this->output);
        $result = $adapt->run();
        echo $result;
    }

} /* End of file adaptation.php */
?>
```

`sources/ci_hook.php`
A.4 Adaptation class (adapt.class.php)

```php
<?php

// Define public variables
// Database settings
public $a_db_hostname= 'localhost';
public $a_db_username= 'adaptive.tuer.nl';
public $a_db_password= '1338 mysql@RS';
public $a_db_name= 'adaptive.tuer.nl';
public $a_db;

// Homepage set
public $home_page = 'page/home';

// [BEGIN php client session information]
public $sessionId;
public $ip_addr;
public $requested_page;
public $requested_page_array;
// [END php client session information]
public $a_globals;

// Output from framework
public $output;

// crv class
public $crv;

// Public benchmark vars
public $a_benchmark;
public $start_time;
public $end_time;

// Cache object
public $data_object_cache = array();
public $data_var_calc_arr = array();

function __construct($output){
    // Start benchmark
    $this->start_time = microtime(TRUE);
    // Get current session
    session_start();
    $this->session_id = session_id();
    $this->ip_addr = $_SERVER['REMOTE_ADDR'];
    $this->requested_page = uri_string();
    $this->requested_page = ((strlen($this->requested_page)==0) ? $this->home_page : $this->requested_page);
    $this->requested_page_array = explode("/", $this->requested_page);
    // If no page, set to /
    // Set output
    $this->output = $output;

    // Database settings
    // Test Database connection
}
```
```php
$db = new mysqli($this->db->hostname, $this->db->username, $this->db->password, $this->db->name);

if ($this->db->connect_errno > 0) {
    $this->output = '<div style="z-index: 9999; position: fixed; width: 100%; bottom: 0; padding: 10px; background: rgb(212, 210, 210);">Unable to connect to database [' . $this->db->connect_error . ']. Please check database settings in the adaptation Class</div>' . $this->output;
    exit;
}

$db_arr = array(
    'hostname' => $this->db->hostname,
    'username' => $this->db->username,
    'password' => $this->db->password,
    'name' => $this->db->name
);

// connect with relation model concept class
require_once('crv.class.php');
$crv = new Crv($db_arr, $this->sessionId);

// get page adaptation data, only apply to page and file URI
$adapt = array('page', '/', 'file', 'apage');
if (in_array($this->requested_page_array[0], $adapt)) {
    if ($this->requested_page_array[0] == 'page') {
        $this->do_adaptation($this->a_globals['startup']);
    }
    $result = $this->do_adaptation($this->output);
    if ($this->a_globals['log']) {
        $result .= '<div style="border: 1px dashed #da0000; background: white; font-family: verdana; font-size: 12px;">' . $this->crv->a_log. '</div>" . $result;
    }
}

function run() {
    // benchmark post result
    $this->end_time = microtime(TRUE);
    if ($this->a_globals['benchmark']) {
        $this->a_benchmark_write("Adaptation loading time:");
        $loading_time = $this->end_time - $this->start_time;
        $result = '<div style="width:100%;border:1px solid red; padding:5px; background:white;">" . $this->a_benchmark . "</div>";
    } elseif ($this->requested_page_array[0] == 'au') {
        $result = $POST['term'];
    } else {
        $result = $this->output;
    }
    return $result;
}
```
function a_benchmark_write($msg, $subject=' '){
    // $callers=debug_backtrace();
    // $subject = $callers[1][1]@''@function';

    // already something written
    if(strlen($this->a_benchmark)){
        $this->a_benchmark .= '&amp;' . date('YmdHims') . ' ' . $subject . ' - ' . $msg;
    } else{
        $this->a_benchmark = date('YmdHims') . ' ' . $subject . ' - ' . $msg;
    }
    return;
}

function get_a_globals(){
    $query = $this->a_db->query("SELECT * FROM a_globals");
    $row = $query->fetch_assoc();
    return $row;
}

function update_adaption_page_data($data_object,$concept, $global=0){
    $data_arr = (array)$data_object;
    $concept_arr = $this->crv->get_a_concept($concept);

    if(!$concept_arr===false){
        foreach($data_arr as $var_name=>$value){
            // get cache
            if(isset($this->data_object_cache[$concept]->$var_name)){
                $data_object_cache = (object)$this->data_object_cache[$concept];
                //check if var needs to be updated
                if($data_object_cache->$var_name != $value){
                    $result = $this->crv->set_a_var_value($var_name, $concept_arr['concept_id'], mysqli_real_escape_string($this->a_db, $value), $global);
                } else{
                    $data_object_cache = (object)$this->data_object_cache[$concept];
                    //set var
                    $result = $this->crv->set_a_var_value($var_name, $concept_arr['concept_id'], mysqli_real_escape_string($this->a_db, $value), $global);
                }
                // update cache
                $data_object_cache->$var_name = $value;
                $this->data_object_cache[$concept] = (array)$data_object_cache;
            }
        }
    }
    return;
}

function remove_quotes($input){
    $input = html_entity_decode($input, ENT_QUOTES);
    $search_arr = array(""", "", htmlentities("''").htmlentities("''"));
$output = str_replace($search_arr, ',', $input);
    return $output;
}

// clean visitor analysis
if ($this->a_globals['clean_toglobal']){
    $query = $this->a_db->query("SELECT var_id, AVG(var_value) as 'var_value', count(*) as 'weight' FROM a_vars_values as a, a_globals as b
        WHERE (TIMESTAMPDIFF(SECOND, a.time, NOW()) > b.clean_expire) AND concat('', var_value + 1) = var_value AND session_id <> '0' GROUP BY var_id");
    if ($query){
        // get all global vars
        $query_global_var_values = $this->a_db->query("SELECT * FROM a_vars_values WHERE session_id='0' AND concat('', var_value + 1) = var_value");
        $global_var_ids = array();
        $global_var_values = array();
        while ($row = $query_global_var_values->fetch_assoc()){  
            $global_var_ids[] = $row['var_id'];
            $global_var_values[$row['var_id']]['value'] = $row['var_value'];
            $global_var_values[$row['var_id']]['weight'] = $row['global_weight'];
        }
        $result = array();
        while ($row = $query->fetch_assoc()){  
            if ('in_array($row['var_id'], $global_var_ids){
                // calculate new value
                $weight = $row['weight'] + $global_var_values[$row['var_id']]['weight'];
                $value = (($row['weight'] - $row['var_value']) + ($global_var_values[$row['var_id']]['value'] - $global_var_values[$row['var_id']]['weight']) / $weight);
                $this->a_db->query("UPDATE a_vars_values SET var_value = ", $value, ", global_weight = ", $weight, " WHERE var_id = ", $row['var_id'], ", session_id='0'";
            } else{
                $this->a_db->query("INSERT INTO a_vars_values (var_id, session_id, var_value, global_weight) VALUES (", $row['var_id'], ", 0, ", $row['var_value'], ", ", $row['weight'], ",")");
            }
        }
    }
    $this->a_db->query("DELETE FROM a_vars_values WHERE (TIMESTAMPDIFF(SECOND, a_vars_values.time, NOW()) > "$this->a_globals['clean_expire']") AND session_id <> '0'";
    $this->a_db->query("UPDATE a_globals SET clean_last = now()";
}

function get_adaptation_page_data($concept, $global=0, $option=0, & $var_calc_arr)
{  
    // retrieve concept
*/
// cache?
if (isset($this->data_object_cache[$concept])) {
    $data_object = (object)$this->data_object_cache[$concept];
    $var_calc_arr = $this->data_var_calc_arr[$concept];
} else {
    $get_concept = $this->crv->get_a_concept($concept);
    // concept not found // create concept?
    if (!$get_concept) {
        // auto_create_concepts
        // ------------------------
        $this->crv->create_a_concept(0, $concept);
        $conceptVars = array();
    } else {
        // retrieve concept
        $conceptVars = $this->crv->get_vars_by_concept($concept, $global, $option);
    }
}

// create data object
$data_object = new stdClass();
foreach ($conceptVars as $key=>$var) {
    $data_object->{$var['var_name']} = $var['var_value'];
    $var_calc_arr[$var['var_name']] = $var['var_calc_arr'];
}

// cache
$this->data_object_cache[$concept] = (array)$data_object;
$this->data_var_calc_arr[$concept] = $var_calc_arr;
}
return $data_object;

// ________________ [lookup or get from data object] ______
function get_adaptation_var($input, $data_object, &$output=' ', $global=0, $option=0, &$arr_var_val = array()) {
    $result=false;
    $a_var = '/([0-9]+)\:\;\{[0-9a-zA-Z]+\} +\S+/si';
    if (preg_match($a_var, $input, $var_matches)) {
        // get val of var
        $global = (strlen($var_matches[1]) ? $var_matches[1] : $global);
        $option = (strlen($var_matches[2]) ? $var_matches[2] : $option);
        $concept = $var_matches[3];
        $var = $var_matches[4];

        // weighted average round functionality
        if (($option == 4) || ($option == 5)) {
            $option = $option - 4;
            $round = true;
        } else {
            $round = false;
        }

        if (strlen($concept) || (($global+$option)>0)) {
            $val = ($this->crv->get_a_var_val($var, $concept, $global, $option, $arr_var_val));
            $val = $val['value'];
        } else {
            // cache?
            if (isset($this->data_object_cache[$concept])) {
                $data_object = (object)$this->data_object_cache[$concept];
                $var_calc_arr = $this->data_var_calc_arr[$concept];
            } else {
                $get_concept = $this->crv->get_a_concept($concept);
                // concept not found // create concept?
                if (!$get_concept) {
                    // auto_create_concepts
                    // ------------------------
                    $this->crv->create_a_concept(0, $concept);
                    $conceptVars = array();
                } else {
                    // retrieve concept
                    $conceptVars = $this->crv->get_vars_by_concept($concept, $global, $option);
                }
            }
        }
    }

if (isset($data_object->{$var})) {
    $val = $data_object->{$var};
} else {
    $val = '';
}
$this->crv->a_log_write("Check: ".htmlspecialchars($a->var));

//round?
if ($round) {
    $output = round($val, 2);
} else {
    $output = $val;
}
$e = true;
return $result;

function do_adaptation ($input, $for_page=false) {
    //output
    $output = $input;
    $css = "";
    $js = "";

    //div index
    $div_index = 0;

    //matches
    $a->global = '/\(\[-\]\$\s.*\)\(\$\|\#\)([0-9a-zA-Z]+)\(\{[0 ,1]\{[0-9]*\}(\[\] )\(\{ [ ] [ ] [ ] )\s?\)\(\$\|\#\)([0-9a-zA-Z]+)\(\{[ ] [ ] [ ]\)\s?\)\(\$\|\#\)([0-9a-zA-Z]+)\(\{[ ] [ ] [ ]\)

    $a->if = '/\(\{[0-9\-a-zA-Z]+\)\s?\)\(\$\|\#\)([0-9a-zA-Z]+)\(\{[ ] [ ] [ ]\)\s?\)\(\$\|\#\)([0-9a-zA-Z]+)\(\{[ ] [ ] [ ]\)

    $a->hash_tag = '/\(\[(0-9a-zA-Z]+\)\s?\)\(\$\|\#\)([0-9a-zA-Z]+)\(\{[ ] [ ] [ ]\)\s?\)\(\$\|\#\)([0-9a-zA-Z]+)\(\{[ ] [ ] [ ]\)

    $a->hash_tag_short = '/\(\[(0-9a-zA-Z]+\)\s?\)\(\$\|\#\)([0-9a-zA-Z]+)\(\{[ ] [ ] [ ]\)\s?\)\(\$\|\#\)([0-9a-zA-Z]+)\(\{[ ] [ ] [ ]\)

    $a->init = '/\(\{[ ] [ ] [ ]\)\(\{ [ ] [ ] [ ]\)\s?\)\(\$\|\#\)([0-9a-zA-Z]+)\(\{[ ] [ ] [ ]\)\s?\)\(\$\|\#\)([0-9a-zA-Z]+)

    $a->var = '/\(\{[ ] [ ] [ ]\)\(\{ [ ] [ ] [ ]\)\s?\)\(\$\|\#\)([0-9a-zA-Z]+)\(\{[ ] [ ] [ ]\)\s?\)\(\$\|\#\)([0-9a-zA-Z]+)

    $a->expression = '/\(#expression \{[ ] [ ] [ ]\)\s?\)\(\$\|\#\)([0-9a-zA-Z]+)\(\{[ ] [ ] [ ]\)\s?\)\(\$\|\#\)([0-9a-zA-Z]+)

    $a->top = '/\(\{[ ] [ ] [ ]\)\(\{ [ ] [ ] [ ]\)\s?\)\(\$\|\#\)([0-9a-zA-Z]+)\(\{[ ] [ ] [ ]\)\s?\)\(\$\|\#\)([0-9a-zA-Z]+)

    //error variable
    $errors = "";

    //global preg_match
    $g_index=0;
    //prevent duplicate lookups on page
    $last_a_data_page = "";
    while (preg_match_all($a->global, $output, $g_matches, PREG_PATTERN_ORDER))
}
{ //start benchmark adaptation page data loading $start_time = microtime(TRUE);
  if ((isset($g_matches[1][$_index])) || (isset($g_matches[9][$_index]))){
    $a_data_page = ((strlen($g_matches[1][$_index] . $g_matches[9][$_index]) >0) ? $g_matches[1][$_index] . $g_matches[9][$_index] : $this->requested_page);
  } else{
    $a_data_page = $this->requested_page;
  } if($last_a_data_page != $a_data_page){
    $var_calc_arr = array();
    $data_object = $this->get_adaptation_page_data($a_data_page,0,0,
      $var_calc_arr);
    $last_a_data_page = $a_data_page;
  }
  //end benchmark page data loading $end_time = microtime(TRUE);
  $this->a_benchmark_write("Data object loading time:" . ($end_time - $start_time));

  switch(true){
    case preg_match_all($a_init, $part, $matches, PREG_PATTERN_ORDER) ? true : false):
      //start benchmark $start_time = microtime(TRUE);
      //loop all adaptations $index = 0;
      foreach($matches[1] as $a_func){
        //get all functions and process
        //already set?
        if(isset($data_object->$a_func) && is_numeric($data_object->$a_func) && isset($matches[3][$_index])){
          //numeric ? or float ?
          $increment_operator = substr($matches[3][$_index],0,1);
          $increment_val = substr($matches[3][$_index],1); if(is_numeric($data_object->$a_func) && (is_numeric($increment_val))){
            eval("$result = ".$data_object->$a_func.$increment_operator.$increment_val."); $data_object->$a_func = $result;
          } else($this->get_adaptation_var($increment_val,$data_object,$increment_val)){
            //get val of var
            if(is_numeric($increment_val)){
              eval("$result = ".$data_object->$a_func.$increment_operator.$increment_val."); $data_object->$a_func = $result;
            } else if(isset($data_object->$a_func)){$init_val = $matches[2][$_index];
              if(is_numeric($init_val)){
$data_object->$a_func = (int)$init_val;
} else if (($this)->get_adaptation_var($matches[3][$index], $data_object, $init_value))&($isset ($data_object->$increment_val))
    // get val of var
    if (is_numeric($init_value))
        $data_object->$a_func = $init_val;
    } else{
        $init_val = $this->remove_quotes($init_val);
        $data_object->$a_func = $init_val;
    }
}

// remove init
$part = str_replace($matches[0][index], '', $part);

// write back
$this->update_adaptation_page_data($data_object, $a_data_page);

// end benchmark
$end_time = microtime(TRUE);
$this->a_benchmark_write("Init loading time:", ($end_time - $start_time));
    // $index++;
    //break;
    break;
    case ( preg_match_all($a_set, $part, $matches, PREG_PATTERN_ORDER)) ? true : false):
        // start benchmark
        $start_time = microtime(TRUE);
        // loop all adaptations
        // get_adaptation_var($input, $data_object, $output);
        $index = 0;
        foreach($matches[1] as $a_func){
            $set_val = $matches[2][index];
            if (is_numeric($set_val))
                $data_object->$a_func = $set_val;
        } elseif (($this)->get_adaptation_var($matches[2][index], $data_object, $set_value))
            // get val of var
            if ($set_value)
                $data_object->$a_func = $set_value;
        else{
            $data_object->$a_func = $this->remove_quotes($set_val);
        }

        // remove set
        $part = str_replace($matches[0][index], '', $part);

        // write back
        $this->update_adaptation_page_data($data_object, $a_data_page);

        // end benchmark
        $end_time = microtime(TRUE);
        $this->a_benchmark_write("Set loading time:", ($end_time - $start_time));

    }
break;       // $index++;  
}  

case preg_match_all($a_if, $part, $matches, PREG_PATTERN_ORDER) 
? true : false):
    // start benchmark
    $start_time = microtime(TRUE);
    
    // loop all adaptations
    $index = 0;
    foreach($matches[2] as $a_func){
        // get all functions and process
        $concept = strlen($matches[1][$index]) ? $matches[1][$index] : $a_data_page;
        $var = $matches[2][$index];
    
        // get left if part
        if($this->get_adaptation_var($concept, '$'. $var, $data_object, $a_if_var)){
            $left_if = $a_if_var;
        } else{
            $part = str_replace($matches[0][$index], '', $a_if_var);
            $this->crv->a_log_write('"If error variable: <i>' . htmlentities($concept) . '</i><br/>"");
            break;
        }

        // get right if
        $right_if = $matches[4][$index];
        if($this->get_adaptation_var($right_if, $data_object, $a_if_var)){
            $right_if = $a_if_var;
        }

        if(!is_numeric($left_if)){
            $left_if = "". addslashes($left_if)."";
        }

        $right_if = html_entity_decode($right_if, ENT_QUOTES);
        eval(' $result = (("".$left_if.html_entity_decode($matches[3][$index]) ."".$right_if.' ) ? true : false); ');
        if($result){
            $part = str_replace($matches[0][$index], html_entity_decode(htmlentities($matches[6][$index]))).$part);
        } else{
            $part = str_replace($matches[0][$index], '$'. $part);
        }
        
        $index++;  
    }  
    
    // end benchmark
    $end_time = microtime(TRUE);
    $this->a_benchmark_write("If loading time:" . ($end_time - $start_time));
    break;  
}
break;
case (preg_match_all($a_echo, $part, $matches, PREG_PATTERN_ORDER) ? true : false):
    // start benchmark
    $start_time = microtime(TRUE);

    // loop all adaptations
    $index = 0;
    foreach($matches[2] as $a_func){
        $concept = (strlen($matches[1][0]) ? $matches[1][0] :
            $a_data_page);
        $result = '';
        if($this->get_adaptation_var($matches[1][0], '$', $matches[2][0], $index), $data_object, $get_result)){
            $result = $get_result;
        } else{
            $part = str_replace($matches[0][0], $result, $part);
        }
        if(strlen($result)){
            $part = str_replace($matches[0][0], '''', $part);
        } else{
            $this->crv->a_log_write('[' . $htmlentities($matches[2][0]) . ']<br />' );
        }
        break;
    }
    // end benchmark
    $end_time = microtime(TRUE);
    $this->a_benchmark_write("Echo loading time: ".( $end_time - $start_time ));

    break;
case (preg_match_all($a_top, $part, $matches,PREG_PATTERN_ORDER ) ? true : false):
    // start benchmark
    $start_time = microtime(TRUE);

    // loop all adaptations
    $index = 0;

    foreach($matches[2] as $a_func){
        // get all functions and process
        $variable = $matches[2][0];
        $sort = $matches[3][0];

        $func = $matches[4][0];
        $options = explode(';', $matches[5][0]);
        $a_top_arr = $var_calc_arr[$variable];

        // filter duplicates
        $filter_arr = array();
        for($i=0;$i<count($a_top_arr);$i++){
            if(in_array($a_top_arr[$i]['concept_name'], $filter_arr)){
                unset($a_top_arr[$i]);
            } else{
                $filter_arr[] = $a_top_arr[$i]['concept_name'];
            }
        }
        $a_top_arr[$i]['var_name'] = $filter_arr;
    }
$return = "";
switch ($sort) {
    case 'max':
        rsort($a_top_arr);
        break;
    case 'min':
        sort($a_top_arr);
        break;
}
$options = explode(";", $matches[5][0]);
switch ($func) {
    case 'concept':
        if (is_numeric($options[0])) {
            $return = $a_top_arr[$options[0]]["concept name"];  
        } break;
    case 'conceptGlobalVar':
        if (is_numeric($options[0])) {
            if (isset($options[1])) {
                $value = $this->crv->get_a_var_val($options[1], $a_top_arr[$options[0]]["concept name"], 1, 0);
                $return = $value["value"];  
            } 
        } break;
    case 'conceptVar':
        if (is_numeric($options[0])) {
            if (isset($options[1])) {
                $value = $this->crv->get_a_var_val($options[1], $a_top_arr[$options[0]]["concept name"], 0, 0);
                $return = $value["value"];  
            } 
        } break;
    case 'order':
        foreach ($a_top_arr as $key=>$item){
            if (isset($options[0])) {
                if ($item["concept name"] == $options[0]) {
                    $return = $key;
                }  
            } break;
    case 'menu':
        /*
         * start-item
         * max-item
         * link name var
         */


$linkVar = "/"

// defaults
$start = 0;
$max = count($a_top_arr);

foreach($options as $key=>$option){
  if(strlen($option)){
    $option_arr = explode(':', $option);
    if(strlen($option_arr[0]) && strlen($option_arr[1])){
      switch($option_arr[0]){
        case 'start':
          $start = $option_arr[1];
          break;
        case 'max':
          $max = $option_arr[1];
          break;
        case 'linkVar':
          $linkVar = $option_arr[1];
          break;
        case 'linkNameVar':
          $linkNameVar = $option_arr[1];
          break;
      }
    }
  }
}

// switch options
$return = "<ul>";
for($i=$start; $i<$max; $i++){
  $key = $i;
  $item = $a_top_arr[$i];

  //------------------------[linkVar]------------------------
  if(isset($linkVar)){
    $value = $this->crv->get_a_var_val($option_arr[1], $item['concept_name'], 1, 0);
    $link = (!$is_numeric($value['value']) ? $value['value'] : "$item['concept_name']");
  } else{
    $link = "../".$item['concept_name'];
  }
  //------------------------[linkNameVar]------------------------
  if(isset($linkNameVar)){
    $value = $this->crv->get_a_var_val($option_arr[1], $item['concept_name'], 1, 0);
    $linkName = (!$is_numeric($value['value']) ? $value['value'] : $item['concept_name']);
  } else{
    $linkName = $item['concept_name'];
  }
  $return .= "<li><a href='$link'>".$linkName."</a></li>";
}

$return = "</ul>";
```php
$return .= "</ul>";
break;
case 'navbar':
break;

$part = str_replace($matches[0][$index], $return, $part);

// end benchmark
$start_time = microtime(TRUE);
$this->a_benchmark_write("Top loading time: ". ($end_time - $start_time));
break;
}

$part = str_replace($matches[0][0], $result, $part);
break;

$part = str_replace($matches[0][0], $result, $part);
break;

$part = str_replace($matches[0][0], $result, $part);
break;

$f this->preg_match_all($a_expression, $part, $matches, PREG_PATTERN_ORDER) ? true : false):
$result = "";
$error = "";
try{
$eval_code = html_entity_decode(html_entity_decode($matches[1][0], ENT_QUOTES));
ob_start();
eval('\$result=' . $eval_code . ');
$error = ob_get_contents();
ob_end_clean();
} catch (Exception $e) {
$f this->crv->a_log_write("Expression: Caught exception: ", $e->getMessage() . "\n");
}
if(strlen($error)){
$f this->crv->a_log_write("Expression: ", $error);
}

$part = str_replace($matches[0][0], $result, $part);
break;

if (strlen($matches[2][$index])){
switch($matches[1][$index]){
case 'interval':
if (strlen($matches[2][$index])){

$matches[4][$index] = (is_numeric($matches[4][$index])) ? (int)($matches[4][$index]) : 0;
$matches[2][$index] = (explode("; ", $matches[2][$index]));
$apage = $matches[2][$index][0];
$replaceid = (isset($matches[2][$index][1]) ? $matches[2][$index][1] : 'console');
//ajax js update if
$js .= "
```
```
```php
$(document).ready(function() {
    setInterval("ajaxd_\$.div_index.\'()\', \$.matches[4][\$index]\');
});

function ajaxd_\$.div_index.\'() {
    $.ajax({
        type: "POST",
        url: ".\/../apage/\$.page.\'",
        data: {
            term: "1"
        },
        success: function(data){
            \$('#\$.replaceid.\'').html(data);
        }
    });
}\
$part = str_replace($matches[0][\$index]."">", $part);
break;
    case 'read':
        if (strlen($matches[2][\$index])){
            $matches[4][\$index] = (is_numeric($matches[4][\$index]) ? (int)($matches[4][\$index]) : 0);
            $matches[2][\$index] = explode(";", $matches[2][\$index]);
            $page = $matches[2][\$index][0];
            $replaceid = (isset($matches[2][\$index][1]) ? $matches[2][\$index][1] : 'console');
            //ajax js update if
            $js .="
                $(document).ready(function() {
                    setTimeout("ajaxd_\$.div_index.\'()\', \$.matches[4][\$index]\');
                });
            
            function ajaxd_\$.div_index.\'() {
                $.ajax({
                    type: "POST",
                    url: ".\/../apage/\$.page.\'",
                    data: {
                        term: "1"
                    },
                    success: function(data){
                        \$('#\$.replaceid.\'').html(data);
                    }
                });
            }\
            $part = str_replace($matches[0][\$index]."">", $part);
        break;
        case 'progressbar':
            if ($this->get_adaptation_var($matches[4][\$index], $data_object, $a_hashtag_var)){
                $matches[4][\$index] = $a_hashtag_var;
            }
            $result = '";
```
$matches[4][$index] = (is_numeric($matches[4][$index]) ? (float)($matches[4][$index]) : 0);
$matches[2][$index] = (is_numeric($matches[2][$index]) ? (float)($matches[2][$index]) : 100);
$percentage = round(((($matches[4][$index]) / $matches[2][$index]) * 100), 2);
$result = '<div class="progress">
    <div class="progress-bar" role="progressbar" aria-valuenow="' . $percentage . '">
        <percentage>.%'</div>
</div>
';
$part = str_replace($matches[0][$index], $result, $part);
break;
default:
    $this->crv->a_log_write("Check: ", htmlentities($matches[0][$index]));
    $part = str_replace($matches[0][$index], '', $part);
break;

// end benchmark
$end_time = microtime(TRUE);
$this->a_benchmark_write("Hashtag short loading time: ", ($end_time - $start_time));
break;
case (preg_match_all($a_hashtag, $part, $matches, PREG_PATTERN_ORDER) ? true : false):
    // start benchmark
    $start_time = microtime(TRUE);
    // loop all adaptations
    $index = 0;
    foreach ($matches[1] as $a_func){
        // get all functions and process
        switch ($matches[1][$index]){
            case "date":
                if ($this->get_adaptation_var($matches[4][$index], $data_object, $a_hashtag_var)){
                    $matches[4][$index] = $a_hashtag_var;
                }
                $result = ''; $matches[4][$index] = (is_numeric($matches[4][$index]) ? (int)($matches[4][$index]) : 0);
                if (strlen(html_entity_decode($matches[3][$index])))};
eval('\$result = ((date(\' . $matches[2][\$index] \'. \'. html_entity_decode($matches[3][\$index]). $matches[4][\$index] \'. \') ? '' : \'. htmllentities($matches[6][\$index]):\' \': \');
\$part = str_replace($matches[0][\$index], $result, $part);

\$js = '"
break;
case "fade":
switch($matches[2][\$index]){
    case 'in':
        if($this->get_adaptation_var($matches[4][\$index], $data_object, $a_hashtag_var)){
            $matches[4][\$index] = $a_hashtag_var;
        }

        $matches[4][\$index] = (is_numeric($matches[4][\$index]) ? (int)($matches[4][\$index]) : 0);
        \$div_id = "a_fadein_" . \$div_index;
        $part = str_replace($matches[0][\$index], "<span id='\$div_id' style='display:none;'>" . $matches[6][\$index] . "</span>" , $part);

        \$js .= "
        jQuery(document).ready(function () {
            setTimeout(\"jQuery('#\$div_id').show();\", " . $matches[4][\$index] . ");
            ");
            break;
        case "out":
            if($this->get_adaptation_var($matches[4][\$index], $data_object, $a_hashtag_var)){
                $matches[4][\$index] = $a_hashtag_var;
            }

            $matches[4][\$index] = (is_numeric($matches[4][\$index]) ? (int)$matches[4][\$index] : 0);
            \$div_id = "a_fadeout_" . \$div_index;
            $part = str_replace($matches[0][\$index], "<span id='\$div_id' style='display:none;'>" . $matches[6][\$index] . "</span>" , $part);

            \$js .= "
            jQuery(document).ready(function () {
                setTimeout(\"jQuery('#\$div_id').hide();\", " . $matches[4][\$index] . ");
                ");
                $css .= ""
                break;
            } break;

        default:
            $part = str_replace($matches[0][\$index], "", $part);
            break;
        }
    \$index++;
    \$div_index++;  
    break;
}  
                     // end benchmark  
$end_time = microtime(TRUE);  
$this->benchmark_write("Hashtag loading time:" .($end_time -  
$start_time));  
                    
break;  
default:  
$part = str_replace($g_matches[0][0],"",$part);  
$this->crv->a_log_write("Check: " .htmlentities($g_matches  
[0][0]));  
break;  
}  
$output = str_replace($g_matches[0][0],$part,$output);  
}  
                     // css and js  
$output = str_replace("{js/}" ,$js ,$output);  
$output = str_replace("{css/}" ,$css ,$output);  
return $output;  
}  
?>
A.5 Concept relation variable class (crv.class.php)

```php
<?php
/*
Name: crv.class.php
Author: Brouwer, S.R.J.

Concept Relationship Variable Class
*/
class Crv{
    // define public variables
    // database settings
    public $a_db_hostname;
    public $a_db_username;
    public $a_db_password;
    public $a_db_name;
    public $a_db;
    // session id
    public $sessionId;
    // output of complete request
    public $output;
    //-----------------(concept, relations, var RegEx)------------------
    public $a_regex = "/[a-zA-Z0-9\-]+/";
    public $a_regex_concept_id = "/[0-9]+";
    public $a_regex_concept_parent_id = "/[0-9]+/";
    public $a_regex_concept_name = "/[a-zA-Z]/[0-9a-zA-Z]+/";
    public $a_regex_var_id = "/[0-9]+/";
    public $a_regex_var_name = "/[a-zA-Z]+/[0-9a-zA-Z]+/";
    public $a_regex_var_inheritance = "/[0-9]{1}/";
    public $a_regex_var_parent_id = "/[1-9][0-9]+/";
    public $a_regex_var_weight = "/[0-9]\.-+/";
    public $a_regex_var_value = ".*/";
    public $a_regex_var_option = "/[0-3]{1}/";
    //--------[log]-----
    public $a_log;

    function __construct($db_arr, $sessionId){
        // cache queries
        ini_set('mysqlnd_qc.enable_qc', 1);
        ini_set('mysqlnd_qc.cache_by_default', 1);

        // set global sessionID
        $this->sessionId = $sessionId;

        // set database parameters
        $this->a_db_hostname = $db_arr['hostname'];
        $this->a_db_username = $db_arr['username'];
        $this->a_db_password = $db_arr['password'];
        $this->a_db_name = $db_arr['name'];
        $this->a_db = new mysqli($this->a_db_hostname, $this->a_db_username, $this->a_db_password, $this->a_db_name);

        //-----------------[Logging]-----------------------
        function a_log_write($msg, $subject=''){   
```

// callers = debug_backtrace();
// $subject = $callers[1]["function "];

// already something written
if (strlen($this->a_log)){
    $this->a_log .= '<br />'.$subject.' - '.$msg;
} else{
    $this->a_log .= date("YmdHims").' '.$subject.' - '.$msg;
}
return;

//-------------------------------[Concepts]

function get_a_concept($concept){
    if (is_numeric($concept)){
        $query = $this->a_db->query("SELECT * FROM a_concepts WHERE concept_id=".$concept);
    } else{
        $query = $this->a_db->query("SELECT * FROM a_concepts WHERE concept_name =".".$concept." ");
    }
    $result = false;
    if ($row = $query->fetch_assoc()){
        $result = $row;
    } else{
        $this->a_log->write('Not found ');
    }
    return $result;
}

function get_a_concepts(){
    $query = $this->a_db->query("SELECT * FROM a_concepts");
    $result = array();
    while ($row = $query->fetch_assoc()){
        $result[] = $row;
    }
    return $result;
}

function create_a_concept($concept_parent_id=0, $concept_name){
    $result = false;
    if ((preg_match($this->a_regex_concept_name, $concept_name)) \&\& (preg_match($this->a_regex_concept_parent_id, $concept_parent_id))){
        if ($query = $this->a_db->query("INSERT INTO a_concepts (concept_parent_id , concept_name) VALUES (".".$concept_parent_id." , ".".$concept_name." ")")){
            $result = true;
            $this->a_log->write('Insert succeeded , id='.$this->a_db->insert_id);
        } else{
            $this->a_log->write('Insert failed , '.$this->a_db->error);
        }
    } else{
        $this->a_log->write('Invalid input ');
    }
    return $result;
}

function update_a_concept($search, $concept_parent_id=0, $concept_name){
    $result = false;
}
if (preg_match($this->a_regex, $search)) {
    $concept_search = $this->get_a_concept($search);
    if(isset($concept_search['concept_id'])){}
    if((preg_match($this->a_regex_concept_name, $concept_name)) && (preg_match($this->a_regex_concept_parent_id, $concept_parent_id))) {
        if($query = $this->a_db->query("UPDATE a_concepts SET concept_parent_id = ", $concept_parent_id , concept_name = " . $concept_name. " WHERE concept_id = " . $concept_search['concept_id'] )) {
            $result = true;
            $this->a_log_write("Update succeeded ");
        } else{
            $this->a_log_write("Update failed , ". $this->a_db->error);
        } else{
            $this->a_log_write("Invalid input ");
        } else{
            $this->a_log_write("Not found, unable to update ");
        } else{
            $this->a_log_write("Invalid search ");
        } return $result;
    }
}

function remove_a_concept($search){
    $result = false;
    if (preg_match($this->a_regex, $search)) {
        $concept_search = $this->get_a_concept($search);
        if(isset($concept_search['concept_id'])){}
        if($query = $this->a_db->query("DELETE FROM a_concepts WHERE concept_id = " . $concept_search['concept_id'] )) {
            $result = true;
            $this->a_log_write("Delete succeeded ");
        } else{
            $this->a_log_write("Delete failed , ". $this->a_db->error);
        } else{
            $this->a_log_write("Not found, unable to delete ");
        } else{
            $this->a_log_write("Invalid search ");
        } return $result;
    }
}

function childs_a_concept($concept_id){
    $query = $this->a_db->query("SELECT * FROM a_concepts WHERE concept_parent_id=" . $concept_id . " AND concept_parent_id <>concept_id ");
    $result = array();
    while($row = $query->fetch_assoc()){
        $result[] = $row;
    }
    return $result;
}

function get_a_relationship($relationship){
// -------------------------------[Relationships]
}
```php
$queue = $this->a_db->query("SELECT * FROM a_relationships WHERE relationship_id='" . $relationship . "'");
$result = array();
if ($row = $queue->fetch_assoc()){
    $result = $row;
} else{
    $result = false;
    $this->a_log_write('Not found');
}
return $result;

function childs_a_relationships($parent_concept_id, $parent_var_name){
    $query = $this->a_db->query("SELECT * FROM a_relationships WHERE parent_concept_id='" . $parent_concept_id . "' AND parent_var_name='" . $parent_var_name . "'");
    while ($row = $query->fetch_assoc()){ "$result [] = $row;
}
    return $result;
}

function get_a_relationships(){
    $query = $this->a_db->query("SELECT * FROM a_relationships");
    $result = array();
    while ($row = $query->fetch_assoc()){ "$result [] = $row;
}
    return $result;
}

function create_a_relationship($parent_concept_id, $parent_var_name, $child_concept_id, $child_var_name, $relationship_weight=1)
{
    $result = false;
    if ( (preg_match($this->a_regex_concept_id, $parent_concept_id) && (preg_match($this->a_regex_concept_id, $child_concept_id) && ($parent_concept_id <> $child_concept_id) ) && (preg_match($this->a_regex_var_name, $parent_var_name)) && (preg_match($this->a_regex_var_name, $child_var_name)) )
        && (preg_match($this->a_regex_relationship_weight, $relationship_weight))
        && (preg_match($this->a_regex_var_id, $child_var_name)))
    {
        if ($query = $this->a_db->query("INSERT INTO a_relationships (parent_concept_id, parent_var_name, child_concept_id, child_var_name, relationship_weight) VALUES ('" . $parent_concept_id . "', " . $parent_var_name . ", " . $child_concept_id . "', " . $child_var_name . ", '" . $relationship_weight . "')")
        ){
            $result = true;
            $this->a_log_write('Insert succeeded, id=' . $this->a_db->insert_id);
        } else{
            $this->a_log_write('Insert failed, ' . $this->a_db->error);
        }
    } else{
        $this->a_log_write('Invalid input');
    }
    return $result;
}
```
function update_a_relationship($search, $parent_concept_id, $parent_var_name, $child_concept_id, $child_var_name, $relationship_weight = 1) {
    $result = false;
    if (preg_match($this->a_regex, $search)) {
        $relationship_search = $this->get_a_relationship($search);
        if (isset($relationship_search['relationship_id'])) {
            if ((preg_match($this->a_regex_concept_id, $parent_concept_id)) &&
                (preg_match($this->a_regex_concept_id, $child_concept_id)) &&
                ($parent_concept_id == $child_concept_id) ||
                (preg_match($this->a_regex_var_name, $parent_var_name)) &&
                (preg_match($this->a_regex_var_name, $child_var_name)) &&
                ($parent_var_name == $child_var_name) &&
                ($relationship_weight == $relationship_weight)) {
                if ($query = $this->a_db->query("UPDATE a_relationships SET
                    parent_concept_id = ":$parent_concept_id," . $parent_var_name = ":" .
                    child_concept_id = ":$child_concept_id," . $child_var_name = ":" .
                    relationship_weight = ":$relationship_weight."
                    WHERE relationship_id = ":$relationship_search['relationship_id']")
                    $result = true;
                $this->a_log_write('Update succeeded');
            } else {
                $this->a_log_write('Update failed', ":$this->a_db->error);
            }
        } else {
            $this->a_log_write('Invalid input');
        } else {
            $this->a_log_write('Not found, unable to update');
        }
        $this->a_log_write('Invalid search');
    }
    return $result;
}

function remove_a_relationship($search) {
    $result = false;
    if (preg_match($this->a_regex_relationship_id, $search)) {
        $relationship_search = $this->get_a_relationship($search);
        if (isset($relationship_search['relationship_id'])) {
            if ($query = $this->a_db->query("DELETE FROM a_relationships WHERE
                relationship_id = ":$relationship_search['relationship_id']")
                $result = true;
            $this->a_log_write('Delete succeeded');
        } else {
            $this->a_log_write('Delete failed', ":$this->a_db->error);
        } else {
            $this->a_log_write('Not found, unable to delete');
        }
        $this->a_log_write('Invalid search');
    }
    return $result;
}
function get_a_vars()

    $query = $this->a_db->query("SELECT * FROM a_vars");
    $result = array();
    while ($row = $query->fetch_assoc()) {
        $result[] = $row;
    }
    return $result;
}

function get_a_vars_by_concept($concept_name, $global=0, $option=0) {

    /* Retrieve all vars of a specific concept with their values */

    // always send empty array
    $result = false;
    // get concept
    $concept = $this->get_a_concept($concept_name);

    if ($concept) {
        // conceptId retrieved — Get all vars
        $query = $this->a_db->query("SELECT * FROM a_vars WHERE concept_id='" . $concept["concept_id"] . "'");
        // set vars array
        $vars = array();
        // loop all returned rows
        while ($row = $query->fetch_assoc()) {
            $vars[] = $row;
        }

        // get all values of the specific rows and set in the result array
        foreach ($vars as &$_var) {
            // if inheritance, calculate value else return
            $value = $this->get_a_var($_var["var_name"], $concept["concept_name"], $global, $option, $_arr_var_val);
            $var["var_calc_val"] = $arr_var_val;
            $var["var_value"] = $value["value"];  
            $var["var_weight"] = $value["weight"];  
        }
        $result = $vars;
    }
    return $result;
}

function create_a_var($var_name, $var_inheritance=0, $concept_id, $var_weight=1, $global=0) {
    $result = false;

    if ((preg_match($this->a_regex_concept_id, $concept_id))
        && (preg_match($this->a_regex_var_name, $var_name))
        && (preg_match($this->a_regex_var_inheritance, $var_inheritance))
        && (preg_match($this->a_regex_var_weight, $var_weight))){

        if ($query = $this->a_db->query("INSERT INTO a_vars (concept_id, var_name, var_inheritance, var_weight) VALUES (" . $concept_id . ", " . $var_name . ", " . $var_inheritance . ", " . $var_weight . ")")
            $this->a_log_write("Insert succeeded, id='" . $this->a_db->insert_id);
$result = true;
}
} else {
    $this->a_log_write('Insert failed , ' . $this->a_db->error);
}
} else {
    $this->a_log_write('Invalid input');
}
return $result;
}

function update_a_var($var_id, $var_name, $var_inheritance=0, $concept_id, $var_weight=1) {
    $result = false;
    if ((preg_match($this->a_regex_concept_id, $concept_id)) && (preg_match($this->a_regex_var_name, $var_name)) && (preg_match($this->a_regex_var_inheritance, $var_inheritance)) && (preg_match($this->a_regex_var_weight, $var_weight))) {
        if ($query = $this->a_db->query("UPDATE a_vars SET concept_id = ", $concept_id, ", var_name='" . $var_name . ", var_weight=" . $var_weight . ", var_inheritance=" . $var_inheritance . ", WHERE var_id=" . $var_id)) {
            if ($this->a_db->affected_rows > 0) {
                $result = true;
                $this->a_log_write('Update succeeded');
            } else {
                $this->a_log_write('Update failed or not necessary');
            } else {
                $this->a_log_write('Update failed , ' . $this->a_db->error);
            }
        } else {
            $this->a_log_write('Invalid input');
        }
    return $result;
}

function set_a_var_value($var_name, $concept_id, $var_value, $global=0, $var_weight=1, $var_inheritance=0) {
    $result = false;
    //check var value exists
    $var_search = $this->get_a_var($var_name, $concept_id, $global);
    if (!$var_search) {
        //insert
        $result = $this->create_a_var($var_name, $var_inheritance, $concept_id, $var_weight);
        if ($result) {
            //again
            $result = $this->set_a_var_value($var_name, $concept_id, $var_value, $global, $var_weight, $var_inheritance);
        } else {
            //set var value
            $result = $this->set_a_var_val($var_search['var_id'], $var_value, $global);
        }
    }
    return $result;
}

function get_a_var_by_id($var_id){
```php
$result = false;
if (preg_match($this->a_regex_var_id, $var_id)) {
    $query = $this->a_db->query("SELECT * FROM a_vars WHERE var_id=" . $var_id);
    if ($row = $query->fetch_assoc()) {
        $result = $row;
    } else {
        $this->a_log_write('Not found');
    }
} else {
    $this->a_log_write('Invalid input');
}
return $result;
}

function get_a_var($var_name, $concept_id) {
    $result = false;
    if ((preg_match($this->a_regex_concept_id, $concept_id)) && (preg_match($this->a_regex_var_name, $var_name))) {
        $query = $this->a_db->query("SELECT * FROM a_vars WHERE var_name='" . $var_name . "' AND concept_id='" . $concept_id);
        if ($row = $query->fetch_assoc()) {
            $result = $row;
        } else {
            $this->a_log_write('Not found');
        }
    } else {
        $this->a_log_write('Invalid input');
    }
    return $result;
}

function remove_a_var($var_id) {
    $result = false;
    if (preg_match($this->a_regex_var_id, $var_id)) {
        if ($query = $this->a_db->query("DELETE FROM a_vars WHERE var_id=" . $var_id)) {
            if ($this->a_db->affected_rows > 0) {
                $result = true;
                $this->a_log_write('Delete succeed');
            } else {
                $this->a_log_write('Delete failed');
            }
        } else {
            $this->a_log_write('Delete failed, ' . $this->a_db->error);
        }
    } else {
        $this->a_log_write('Invalid input');
    }
    return $result;
}

// ------------------------------- REMOVE var values 
function remove_a_var_val($var_id, $global) {
    $result = false;
    if (preg_match($this->a_regex_var_id, $var_id)) && (preg_match($this->a_regex_var_global, $global)) {
        $query_session = ($global ? 0 : $this->session_id);
    }
```

if ($query = $this->a_db->query("DELETE FROM a_vars_values WHERE session_id=" . $query_session . " AND var_id=" . $var_id)) {
    if ($this->a_db->affected_rows > 0) {
        $result = true;
        $this->a_log_write('Delete succeeded');
    } else {
        $this->a_log_write('Delete failed');
    }
    $this->a_log_write('Delete failed', '. $this->a_db->error);
} else {
    $this->a_log_write('Invalid input');
}
return $result;

function set_a_var_val($var_id, $var_value, $global=0) {
    $result = false;
    if ( (preg_match($this->a_regex_var_id, $var_id)) && (preg_match($this->a_regex_var_value, $var_value)) && (preg_match($this->a_regex_var_global, $global)) ) {
        $query_session = ($global ? 0 : $this->session_id);
        if ($query = $this->a_db->query("SELECT var_value FROM a_vars_values WHERE session_id=" . $query_session . " AND var_id=" . $var_id)) {
            if ($row = $query->fetch_assoc()) {
                // update
                $result = $this->update_a_var_val($var_id, $var_value, $global);
            } else {
                // add
                $result = $this->add_a_var_val($var_id, $var_value, $global);
            }
            else {
                $this->a_log_write('Get var value failed', '. $this->a_db->error');
            }
        } else {
            $this->a_log_write('Invalid input');
        }
    return $result;
}

function add_a_var_val($var_id, $var_value, $global=0) {
    $result = false;
    if ( (preg_match($this->a_regex_var_id, $var_id)) && (preg_match($this->a_regex_var_value, $var_value)) && (preg_match($this->a_regex_var_global, $global)) ) {
        $query_session = ($global ? 0 : $this->session_id);
        if ($query = $this->a_db->query("INSERT INTO a_vars_values (var_id, session_id, var_value) VALUES (" . $var_id . ", " . $query_session . ", " . $var_value . ")")) {
            $this->a_log_write('Insert succeeded', id=' . $this->a_db->insert_id);
            $result = true;
        } else {
            $this->a_log_write('Insert var value failed', '. $this->a_db->error');
        }
    return $result;
}
```php
// UPDATE var values
function update_var_values($var_id, $var_value, $global = 0) {
    $result = false;
    if ((preg_match($this->regex_var_id, $var_id)) && (preg_match($this->regex_var_value, $var_value)) && (preg_match($this->regex_var_global, $global))) {
        $query_session = ($global ? 0 : $this->session_id);
        if ($query = $this->db->query("UPDATE a_vars_values SET var_value='\". $var_value. '\' WHERE session_id='\". $query_session. '\' AND var_id='\". $var_id) {
            $this->a_log_write('Update succeeded');
            $result = true;
        } else {
            $this->a_log_write('Update var value failed, '. $this->db->error);
        }
    } else {
        $this->a_log_write('Invalid input');
    }
    return $result;
}

// GET var values
function get_var_values_by_id($var_id, $global = 0) {
    $result = false;
    if ((preg_match($this->regex_var_id, $var_id)) && (preg_match($this->regex_var_global, $global))) {
        $query_session = ($global ? 0 : $this->session_id);
        if ($query = $this->db->query("SELECT var_value FROM a_vars_values WHERE session_id='\". $query_session. '\' AND var_id='\". $var_id) {
            if ($row = $query->fetch_assoc()) {
                $result = $row;
            } else {
                $this->a_log_write('Get var value not found');
            }
        } else {
            $this->a_log_write('Get var value failed, '. $this->db->error);
        }
    } else {
        $this->a_log_write('Invalid input');
    }
    return $result['var_value'];
}

function get_var_values($var_name, $concept_name, $global = 0, $option = 0, & $arr_var_val = array()) {
    $result = array();
    $result['value'] = 0;
    $result['weight'] = 0;
    /* option
    0 = default; value including possible relations and weight
    1 = Weighted Average
    2 = from average; take global + specific score
    3 = from average; take global + specific score weighted
    */
```
489  // check input
490  if ((preg_match($this->a_regex_concept_name, $concept_name)) &&
491      (preg_match($this->a_regex_var_name, $var_name)) &&
492      (preg_match($this->a_regex_var_global, $global)) &&
493      (preg_match($this->a_regex_var_option, $option))) {
494      // get concept
495      $concept = $this->get_a_concept($concept_name);
496      if (!$concept) {
497          $this->a_log_write('Get concept failed');
498          return $result;
499      }
500      // get var
501      $var = $this->get_a_var($var_name, $concept['concept_id']);
502      if (!$var) {
503          $this->a_log_write('No (child) var within concept');
504          return $result;
505      }
506      // inherited
507      if ($var['var_inheritance']) {
508          $value_arr = array();
509          $weight_arr = array();
510          // initiate vars
511          $value_arr[] = 0;
512          $weight_arr[] = 0;
513          // if parent has value and weight
514          if ($var['var_parent_value']){
515              $parent_value = $this->get_a_var_val_by_id($var['var_id'], $global);
516              $value_arr[] = $parent_value;
517              $weight_arr[] = $var['var_weight'];
518          }
519      }
520      // get parent childs
521      $childConcepts = $this->childs_a_concept($concept['concept_id']);
522      foreach ($childConcepts as $childConcept) {
523          // get child variable value if exists
524          $child_var = $this->get_a_var_val($var_name, $childConcept['concept_name'], $global, $option, $arr_var_val);
525          if (!$child_var['weight']==0) {
526              if (is_numeric($child_var['value'])){
527                  $value_arr[] = $child_var['value'];
528                  $weight_arr[] = $child_var['weight'];
529              } else {
530                  $this->a_log_write('Child value not numeric: '. $childConcept['concept_name'], $global, $option, $arr_var_val, $child_var);
531              }
532          } else {
533              $this->a_log_write('Child value not numeric: '. $childConcept['concept_name'], $global, $option, $arr_var_val, $child_var);
534          }
535      }
```php
// get relationships
foreach ($this->childs->a_relationships($concept['concept_id']).
  $var['var_name']) as $childRelationship{
  if ($childRelationship['child_concept_id']==$concept['concept_id']){
    // get concept of child relationship
    $childRelationshipConcept = $this->get_a_concept($childRelationship['child_concept_id']);
    if ($childRelationshipConcept){
      $childRelationshipVar = $this->get_a_var($childRelationship['child_concept_id'], 0);
      if ($childRelationshipVar){
        // get child variable value if exists and belongs to same concept
        if ($childRelationshipVar['concept_id']==
          $childRelationship['child_concept_id']){
          $child_var = $this->get_a_var_val($childRelationshipVar['var_name'], $childRelationshipConcept['concept_name'],
            $global, $option, $arr_var_val);
          if (!$child_var['weight']==0){
            if (is_numeric($child_var['value'])){  
              $value_arr[] = $child_var['value'];
              $weight_arr[] = $child_var['weight'] * $childRelationship['relationship_weight'];
            } else{
              $log_child_var = (isset($child_var['var_name']) ? $child_var['var_name'] : '');
              $this->a_log_write('Child value not numeric: '. $childRelationshipConcept['concept_name'] .'. ', $log_child_var);
            }
          } else{
            $this->a_log_write('Child with zero weight: '. $childRelationshipConcept['concept_name'] .'. ', $child_var['var_name']);
          }
        } else{
          $this->a_log_write('Child concept and var dont match/belong: '. $childRelationship['child_concept_id'] .'. ' . $childRelationship['child_var_name']);
        }
      } else{
        $this->a_log_write('Child concept not found: '. $childRelationship['child_concept_id'] .'. ' . $childRelationship['child_var_name']);
      }
    } else{
      $this->a_log_write('Child var not found: '. $childRelationship['child_concept_id'] .'. ' . $childRelationship['child_var_name']);
    }
  } else{
    $this->a_log_write('Child with zero weight or does not exist: '. $childConcept['concept_name']);
  }
}
```
child_var_name}));
}
else{
    $this->a_log_write('Loop on relationship:'. $concept['concept_id']);
}

//calculate result depending on option
/*
  2 = global * weight + case 0
  3 = global * weight + case 1
*/
if(( $option==2)||($option==3)){
    $global_value = $this->get_a_var_val($var_name, $concept_name, 1, 0);
    $value_arr[] = $global_value['value'];
    $weight_arr[] = $global_value['weight'];
    $option = $option-2;
}

switch($option){
    case 0:
        //default
        foreach($value_arr as $key=>$value){
            $result['value'] = $result['value'] + ($value * $weight_arr[$key]);
        }
        //end value
        $result['value'] = $result['value'];
        //weight of parent var
        $result['weight'] = $var['var_weight'];
        break;
    case 1:
        //wa
        //sum of weight
        $weight_sum = 0;
        foreach($value_arr as $key=>$value){
            $result['value'] = $result['value'] + ($value * $weight_arr[$key]);
        }
        //end value
        $weight_sum = ($weight_sum == 0 ? 1 : $weight_sum);
        $result['value'] = $result['value'] / $weight_sum;
        //weight of parent var
        $result['weight'] = $var['var_weight'];
        break;
}

//add to value array if parent has it's own value
if($var['var_parent_value']){
    $arr_var_val[] = array(
        'weight' => $result['weight'],
        'value' => $result['value'],
        'var_name' => $var_name,
        'concept_name' => $concept_name
    );
}
else {
    $result['value'] = $this->get_a_var_val_by_id($var['var_id'],
        $global);
    $result['weight'] = $var['var_weight'];
    // set array list of variable
    $arr_var_val[] = array(
        'weight' => $var['var_weight'],
        'value' => $result['value'],
        'var_name' => $var_name,
        'concept_name' => $concept_name
    );
} else {
    $this->log_write('Invalid input');
}
return $result;
?>

sources/crv.class.php