Adaptive System Framework

A Way to a Simple Development of Adaptive Hypermedia Systems

Balík Martin and Jelínek Ivan
Department of Computer Science and Engineering
Faculty of Electrical Engineering, Czech Technical University
Prague, Czech Republic
e-mail: {balikm1, jelinek}@fel.cvut.cz

***Abstract*—Adaptive hypermedia systems (AHS) are complex systems that require an expensive and time-consuming design and development process. Complex solutions are usually realized as reusable frameworks and program libraries. However, there is currently no widely acceptable solution for building AHS. Based on our research, we are developing a framework that could significantly make the design and development of AHS easier. First, we formalized the adaptive system architecture, and then, we defined basic structures for storing required data. Further, we designed the adaptation and integration modules and developed reusable adaptive web user interface components. Such a framework is considered to become a foundation stone for various types of AHS.**

***Keywords*—adaptive hypermedia; personalisation; framework; software development**

I. INTRODUCTION

The purpose of adaptive hypermedia systems (AHS) is to adapt content, presentation and navigation of hypermedia to satisfy user's needs and preferences. The fundamental principle of AHS is to observe user's behavior, to build a user model reflecting all user's characteristics, e.g., knowledge, preferences, or event history, and to customize the pages presented to the user based on these characteristics. The aim of our research is to provide a reference model of AHS and its implementation that would contribute to the facilitation of an AHS development process.

The Generic Ontological Model for Adaptive Web Environments (GOMAWE) [1] forms a theoretical basis for an application framework that should rapidly simplify and speed up the AHS development. This is achieved by reusable ready-to-use software components provided by the framework implementation and extensibility of the framework for further use cases and novel technologies.

The Adaptive System Framework (ASF) was built to help a software developer create adaptive web applications. ASF provides the most typical AHS components serving as building blocks for further development. ASF is based on the theoretical model and satisfies the following important requirements. To be generally applicable, the framework has to be split into components with independent responsibilities. To follow generally accepted solutions to common application problems, design patterns [2] should be extensively used. The implementation of the framework should be based on well-known and widely used application frameworks. In contrast to other frameworks focusing on the users' collaboration [3] or adaptation process modeling [4], our framework aims at formalization of adaptive system architecture. Further, it focuses on targeting the problem of a storage layer abstraction, foundations of the data structures needed for a user modeling and providing a basic set of adaptation-oriented user interface components.

The paper is structured as follows. Section 1 deals with the description of AHS and the tasks to solve. In Section 2, a current state of the art of the discussed topic is being reviewed. In Section 3, an ASF framework is described in detail focusing on individual layers. In Section 4, both an evaluation method and application of the framework in a prototype implementation is discussed. Finally, the paper concludes by summarizing results of the research and indicates the directions of the future work; see Section 5.

II. RELATED RESEARCH

Hypermedia adaptation has become a topic of a researchers' interest for almost 15 years. The researchers have been trying to formalize an adaptive system architecture, behavior and to line up with the existing web development standards. Since self-adaptive systems are complex, they require a special approach in the process of designing and developing such projects.

In the early stage of the AHS research, the model AHAM with its implementation AHA! [5] was the first widely used architecture of the adaptive systems. The AHAM domain model can be represented by single ontology, since it deals with the concepts and their mutual relations. However, it was not designed to deal with multiple ontologies. The AHA! system is built on the basis of outdated technologies, and the web user interface is not in compliance with modern web standards. Therefore, we offer the improvements which lie in building a novel framework based on up-to-date web technologies.

In the following text, we present a layout of already existing adaptation frameworks and libraries.

GRAPPLE (Generic Responsive Adaptive Personalized Learning Environment) [6] has been developed at the Eindhoven University of Technology, as a part of the FP7 project [7]. This system is focused on adaptive learning. It is integrated with existing learning systems (e.g., Blackboard [8] or Moodle [9]). The most important contribution of this project lies in integrating the adaptive delivery of the teaching materials for the course into a supported learning process.

 ISBN: 978-1-61208-274-5

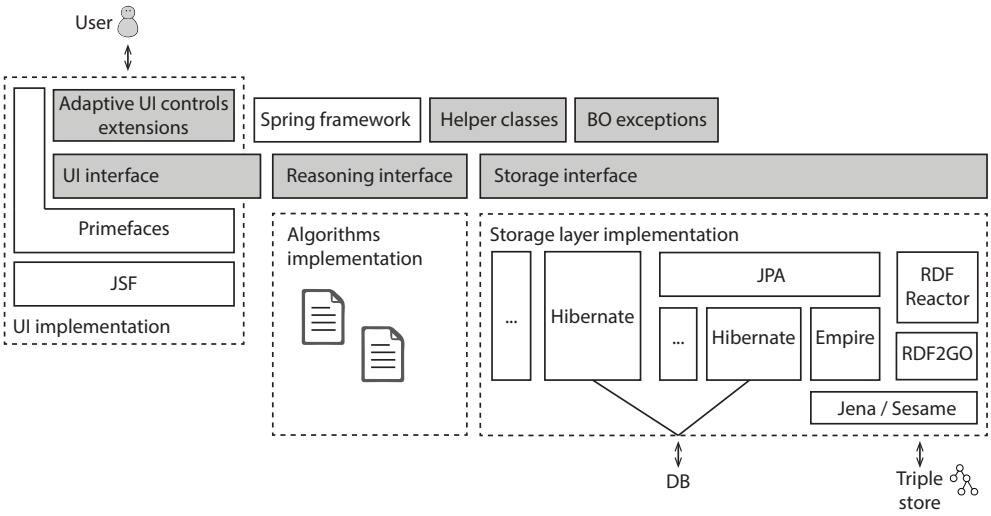

Figure 1. Adaptive system framework architecture

Another project, specifically focused on adaptive learning, is the Adaptive eLearning Platform [10]. This system is the implementation of the Virtual Apparatus Framework, a content development paradigm modeled after the process of developing a teaching lab activity. The approach used in this project is tightly connected to the learning process and is based more on pedagogical principles than on software engineering.

A further promising project is also HyperAdapt [11]. In this project, a specialized approach utilizing an aspect-oriented programming is used. The authors place the adaptivity into separate modules called adaptation aspects. The aspects are not applied on a model level, but on XML documents.

One of the solutions intended to extend legacy web applications with adaptive behavior is the Adaptive Server Framework [12]. Compared to our solution, this project is focused on server-side components only. The design principle is to separate the implementation of adaptive behavior from the server application business logic. The coupling of components is ensured by a message-based communication.

A similar solution is the Rainbow project [13]. This project uses an architecture-based approach. The system adaptation is predefined by the architecture style of the system. The commonly used design principle is the principle of a modular architecture.

Another solution partially inspiring our design of GOMAWE is the MUSE semantic framework [14]. The framework is built on multidimensional ontological planes. The intersection between the planes allows the representation of semantic rules. A similar principle is used in GOMAWE, where a multidimensional matrix of rules is used to infer the information not explicitly stored in the user model.

In comparison with other solutions, our ASF project aims at supporting not only adaptive learning, but also adaptive hypermedia systems in general. The purpose of our project is to provide a reusable solution that could be used by the developers of adaptive applications. We formally described adaptive hypermedia within the GOMAWE model and designed the framework that will be described in detail in the following sections. The Adaptive System Framework is a solution for a simpler and rapid development of AHS.

III. Framework architecture

The ASF architecture is generic. It defines an interface of various layers of the potential system, and therefore its implementation can be realized in multiple programming languages. Fig. 1 represents the description of ASF architecture based on our GOMAWE model. The ASF architecture consists of the layers replacing the GOMAWE Storage, Reasoning and UI interface layers. The main highlighted components of the core framework library define the fundamentals of the architecture. The default implementation is built on the selected persistence frameworks supporting relational database and triple stores. In our experiments the storage layer was based on various frameworks, e.g., JPA, Hibernate, Empire, or RDF Reactor. However, our framework can be extended by any other implementation of the storage interface. The extensions are indicated by the '...' symbol in the diagram; see Fig. 1. The same situation is in case of algorithms performing the adaptation above the storage layer. Some algorithms are part of the framework, others can be added by the developer as an implementation of the reasoning interface.

The user interface is based on Java Server Faces (JSF) and is supported by a Primefaces components suite [15]. Our goal is to extend basic web components by the adaptation-specific extensions; see Fig. 1. We think that there are several adaptation techniques that the framework can provide "out of the box" allowing the developers to apply the adaptation without any need of additional work.

A. Data storage

One of the most important parts of every adaptive system is the user model. This is the repository, where information about the user is stored. This information is used in the adaptation process to filter information and personalize the presentation according to the user's preferences.

 ISBN: 978-1-61208-274-5

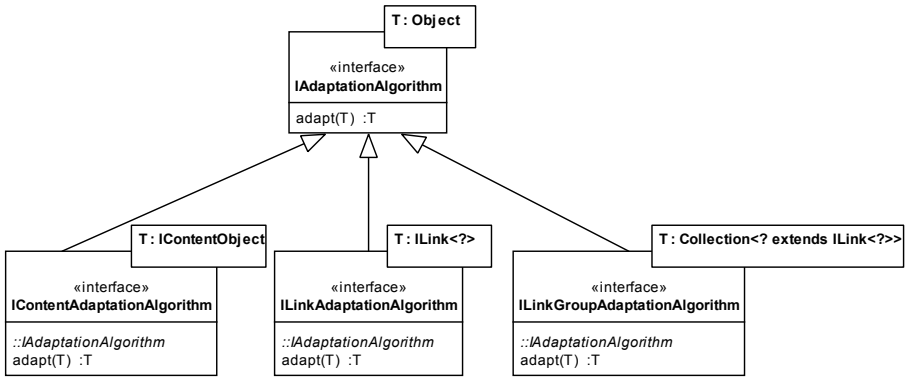

Figure 2. Adaptation algorithms classification

We divided the user data into two parts – the user profile and user model. The user profile contains explicit user's preferences. This data is corresponding to the "settings page" and is stored as key-value pairs. The key is usually a constant string defined by the developer of the application. The user model, on the other hand, stores the data observed while the application monitors the user. The information is always associated with a domain object and represents the user's relation to the object, e.g, user's knowledge of the topic, user's preferences, or their past experience. The user model corresponds to the overlay over the application domain model.

The access to the user model and user's profile is supported by an adaptation manager. The adaptation manager implementation is based both on the Singleton Design Pattern providing a manager instance and the Factory Method Design Pattern used for creating domain-specific model instances for an individual user.

In our design, the user model has a special architecture. Each attribute is assigned to one dimension. Dimensions can be custom-defined for each adaptive system. The dimension forms a group of related attributes. It can be visualized as a multidimensional matrix.

The multidimensional user model is formally described as follows:

Definition 1 (Multidimensional User Model)**.** The Multidimensional User Model is a tuple $MUM = (D, A, V)$

$$r : A \to V | \forall a \in A : r_{(a)} \in V_a \wedge r_{(a)} \in D, \qquad (1)$$

where D is a finite set of dimensions, A is a finite set of attributes, each associated with a particular dimension, V is a set of attribute values and V_a is the domain of attribute a.

Another important part of the adaptive system storage is the rule repository. The rules can represent conditions defined by the author of the content of the application, e.g., by a teacher who prepares an adaptive course, or they can be generated by specialized adaptation algorithms. The rules assume that the user-model characteristics are associated with predefined dimensions. The dimensions can be used to filter the rules while the rules are being evaluated. This contributes to better performance and helps the designer of an adaptive algorithm to maintain the rules easier.

Both the domain and the user model can be represented by simple concepts and their relations. However, our solution was designed to use multiple lightweight ontologies. The use of ontologies was motivated by the requirements of the data semantics, data exchange and integration among applications, and as well, by the need to infer the information implicitly stored in the user model.

The adaptive process is executed above the storage layer and acts as a mediator between the raw data and the user.

B. Adaptive behavior

One of the goals of the framework optimization is to make components of an adaptive system reusable and generally applicable. To achieve this requirement, we defined a general interface over any algorithm that will be used to perform the adaptation (Fig. 2).

The adaptation algorithms are further divided according to the "adaptation techniques taxonomy" (Fig. 3). A simplified version of the taxonomy is based on the taxonomy defined in [16]. Less important techniques currently not implemented by the framework were excluded from the original taxonomy. A content adaptation, link adaptation and link-group adaptation algorithms are specified in the framework.

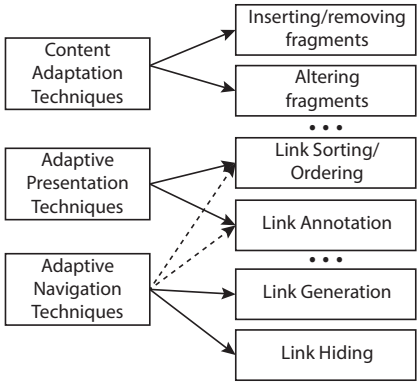

Figure 3. Simplified adaptation techniques taxonomy

A content adaptation algorithm is an algorithm which is used to transparently transform the content of the domain concepts based on the user model. It can be used to substitute the elements of the domain concepts. For example, we can recognize various extents based on the user's knowledge stereotypes (beginner, advanced, expert).

A link adaptation algorithm is intended for customizing a single link. On the other hand, the link-group adaptation algorithm assumes a collection of the links to serve as both input and result. The links can be adapted by sorting. The input collection is processed, and the order of the links is modified. A different approach is used for a link generation. In this case, the result is not dependent on the adaptation function input, since the data is retrieved from a repository. Other link adaptation strategies include a direct guidance, link annotation or adaptive link hiding.

There are two main types of the link adaptation algorithms in the framework:

- adaptation based on the current context, where the existing links can be sorted, filtered, etc.
- link generation, where the algorithm is responsible only for retrieving the input data. A default value can be provided to the adaptation function. It can be used in case when the user disables the adaptation, or if there is not sufficient input data to generate the links automatically.

In GOMAWE, the adaptation was designed as an extendable set of black box components that perform the adaptation based on the information stored in the user model. A framework implementation (Fig. 4) is realized as a Strategy Design Pattern [2]. The intent of the Strategy Design Pattern is, first, to define a family of algorithms, second, encapsulate each of them, and third, make them interchangeable. It enables the algorithm to vary independently of the clients that use it.

We can define an elementary adaptation as an adaptation function.

Definition 2 (Adaptation Function)**.** An Adaptation Function AF is a transformation between default and adapted hypermedia elements. A hypermedia element is considered as a portion of HTML code that is a part of the web page.

$$AF : e_d \rightarrow e_a, \tag{2}$$

where e_d is the default element, and e_a is the adapted element.

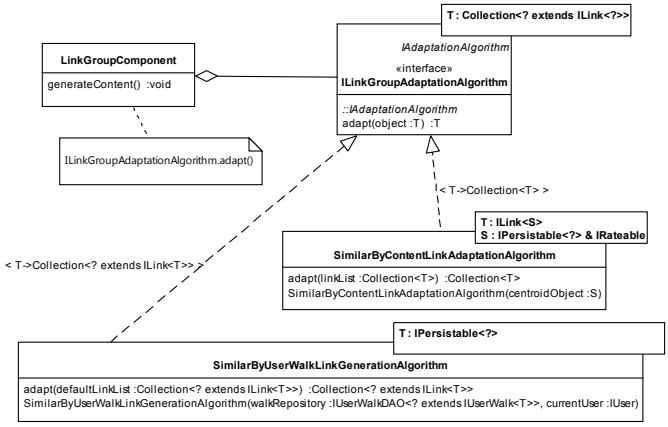

Figure 4. Link-group adaptation algorithms

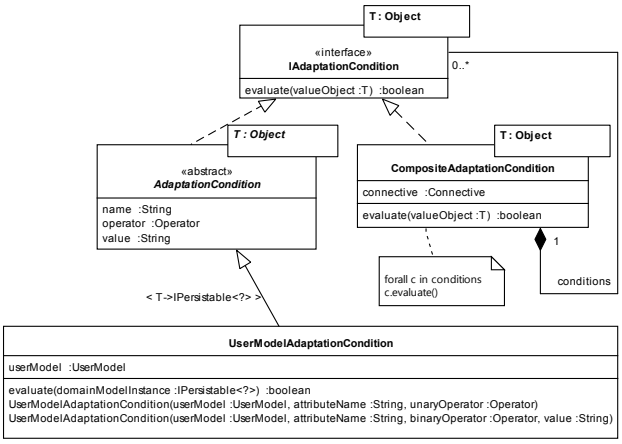

Figure 5. Condition class hierarchy

A generic group adaptation function usually takes a collection of default or initial values as an input, and returns an adapted collection of items of the same type. The particular algorithm is encapsulated inside the black box which is implemented as a class. Algorithm instances can be optionally parametrized before the actual adaptation is performed.

Another extension of the adaptation component will lie in the meta-adaptation support, where best-suited algorithms will be adaptively selected. For this purpose, the Strategy Design Pattern will be extended to the Adaptive Strategy Design Pattern [17]. The Adaptive Strategy Design Pattern defines a self-adaptive strategy. A single strategy referencing the best available concrete strategy is exposed to the client and the client is required only to provide an access to the environment information that can be used to choose the best strategy.

The data filtering in the user model is based on the conditions (Fig. 5). Condition classes follow the Composite Design Pattern [2]. The task of the Composite Design Pattern is to compose objects into tree-like structures to represent part-whole hierarchies. The Composite allows clients to treat individual objects and the compositions of objects uniformly. The conditions have multiple applications in the framework. The same hierarchy is used for evaluating the rules. The purpose of the condition is determined by a particular implementation of the abstract *AdaptationCondition* class.

The rules are defined on the intersections of the user model dimensions. The dimensions are used to limit the information space and to contribute to better evaluation performance. There are two ways of creating the rules in the data storage. The rules can be defined directly by the content designer, or they can be a product of an adaptation algorithm. The combination of these techniques can lead to interesting adaptive behavior. This will be the objective of our future research.

In the following sub-section, we will show an example of an adaptation algorithm that can be integrated into the system.

 ISBN: 978-1-61208-274-5

$document \leftarrow currentWalk$.get($currentWalk$.size$-1$)
for all userWalk in walks **do**
  $newIndex \leftarrow currentWalk$.getPosition($document$)
  $oldIndex \leftarrow userWalk$.getPosition($document$)
  {compare a similarity of documents preceding the current one in the current and a stored walk}
  while $newIndex \geq 0 \wedge oldIndex \geq 0$ **do**
    if $currentWalkDoc \neq userWalkDoc$ **then**
      break
    else
      increment quality and decrement indexes
    end if
  end while
  if $quality > 0$ **then**
    put into document-quality map
  end if
end for
sort document quality map by document quality
return set of documents sorted by quality

Figure 6. Algorithm: Get possible subsequent documents by comparing the history of the user walk transitions

Example (Walking the document space)**.**
To clerify a link adaptation, we will show how the adaptive link generation based on the similarity of navigating paths within the document space is supported by the framework.

Let us have a finite set D of the documents d. A document walk is an ordered set W, where $W \subseteq P$. The document walk is always associated with the user and represents a navigation sequence of the user throughout the document set in a single session. Finding a similar sequence allows us to predict the next most suitable document for a current user based on other users' behavior (Fig. 6).

The desired algorithm is classified as a link-group adaptation algorithm based on ASF (Fig. 2). In our case, it is a link generation algorithm.

Fig. 7 presents a sequence diagram describing typical steps of the adaptation algorithm sequencing. In our specific case, the user walk algorithm, first, requests the default values from the user model (in case of a link generation, this step is not required), and, second, it loads other user walks from the repository. Based on this data, a set of recommended subsequent documents is returned to the content generator.

C. User interface

The most user-oriented layer of the framework comprises the user interface components that are customized for the web-page adaptation. The components are based on the JSF and the Primefaces component suite. The components utilize JavaScript and AJAX to provide rich user experience. An added value to the commonly used web components is the tight coupling with adaptive behavior, user model and the adaptation engine.

An adaptive text output can be taken as an example of a simple component. In a common web component framework, we can find a text output component generating a text to the web page. The text content selection or customization must be done by the developer. In our framework, we want to provide intelligent web components tightly bound with the adaptation engine. The adaptive text component is able to provide various content adaptation techniques. The functionality of the component should be based on the configuration, selected algorithm and on the provided data storage for the data binding. Any of these parameters can be changed later, without any significant modifications to the web page logic and code.

IV. PROTOTYPE VALIDATION

The Adaptive System Framework was applied in the development of a learning course. We chose an adaptive learning environment since the adaptation is very often applied in this area. The university environment provides many opportunities to evaluate such an application in the courses and seminars. Our adaptive learning application was used in a C language programming course.

The implementation of the storage layer was based on JPA API and Hibernate implementation. MySQL Server database was used as a data storage. The choice of the

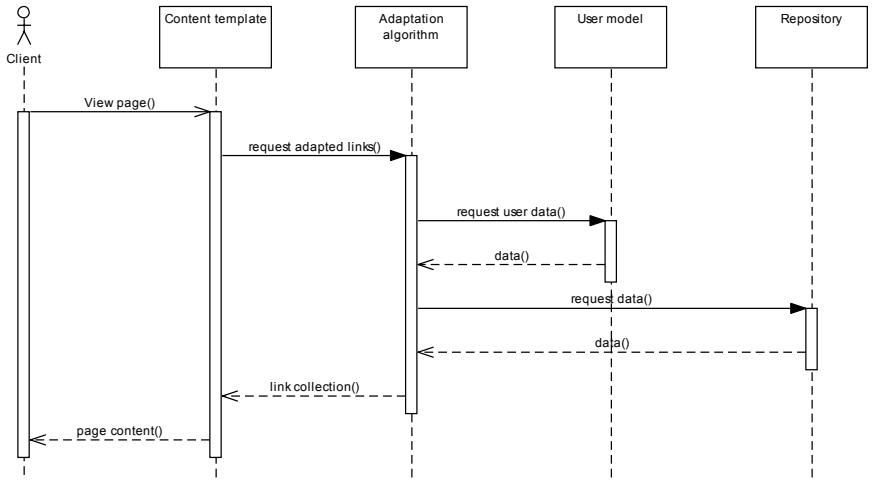

Figure 7. Adaptation algorithm usage

storage limited some benefits of the framework. However, as a prototype, it was sufficient enough to validate the framework architecture. At present, we are working on the future extensions of the adaptive application, where the ontologies representing an important feature of our design will be used. In the next version of the learning course, the data will be saved in a triple store and the integration features will be evaluated.

The centralized user model management is beneficial for the application development. Using the adaptation manager, the user profile and user model properties can be accessed from any component of the application.

The design of the application core based on the ASF framework consists of the following important steps:

1) Definition of the domain objects – in case of the learning application called learning objects and their relations
2) Definition of the user profile and user model attributes
3) Design of the adaptive algorithms for the desired behavior
4) Configuration of data sources
5) Binding the data results either to the application logic or directly to the adaptive UI components

The application was tested by 35 students, and the content was limited to one topic of one week of the semester. From the results of the log analysis, we could get interesting observations regarding the feedback from the students. In this phase, the feedback was limited to a preference screen only. While using an online course, 5 students (14%) tried to change the adaptation setting and 5 of them tried to reset the result statistics. More students were interested in personal settings, particularly, the visual theme of the application. 12 students (34%) changed the visual theme. From these results, we can conclude that a default setting of the adaptation is very important and that the adaptation based on automatic observations of the users should be extensively applied. An explicit feedback can be expected after the users become more friendly with the system and start customizing the system to be more comfortable to use.

V. Conclusions and future work

The Adaptive System Framework can be regarded as a possible solution to an effective development of adaptive hypermedia systems. The proposed framework defines fundamental components and is based on a formal model. It provides various possibilities of its implementations and their further extensions. It leads to a simplified process of the development and, at the same time, it does not limit the developer in customized extensions. The default framework implementation is based on both modern frameworks used for the development of the web applications and state-of-the-art technologies of the Semantic Web.

We have verified the framework by implementing a prototype of e-learning web application. In the future, we would like to extend the framework with more reusable adaptable JSF components based on the well-known adaptation techniques. Extending a set of implemented algorithms will enable use new methods of the personalization. The results will be thoroughly verified by the application of an adaptive web-based learning in real-class scenarios.

Acknowledgment

The results of our research form a part of the scientific work of a special research group WEBING [18]. The work was supported by the grant of the Grant Agency of the Czech Technical University in Prague.

References

[1] M. Balík and I. Jelínek, "Towards Semantic Web-based Adaptive Hypermedia Model," in ESWC Ph.D. Symposium, Tenerife, Spain, 2008, pp. 1–5.

[2] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design patterns: elements of reusable object-oriented software. Addison-Wesley Longman Publishing Co., Inc., 1995.

[3] M. Šimko, M. Barla, and M. Bieliková, "ALEF : A Framework for Adaptive Web-Based Learning 2.0," Ifip International Federation For Information Processing, vol. 324, 2010, pp. 367–378.

[4] E. Knutov, P. De Bra, and M. Pechenizkiy, "Generic Adaptation Framework: A Process-Oriented Perspective," Journal Of Digital Information, vol. 12, no. 1, 2011.

[5] H. Wu, E. de Kort, and P. De Bra, "Design issues for general-purpose adaptive hypermedia systems," in Proceedings of the twelfth ACM conference on Hypertext and Hypermedia - HYPERTEXT '01. New York, New York, USA: ACM Press, 2001, pp. 141–150.

[6] P. De Bra, D. Smits, K. V. D. Sluijs, A. I. Cristea, and M. Hendrix, "GRAPPLE: Personalization and adaptation in learning management systems," in Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications (ED-MEDIA 2010), Toronto, Canada, 2010, pp. 3029–3038.

[7] GRAPPLE Project Website. [Accessed: Apr. 24, 2013]. [Online]. Available: http://www.grapple-project.org/

[8] Blackboard Learning System Website. [Accessed: May 6, 2013]. [Online]. Available: http://www.blackboard.com/

[9] Moodle Learning System Website. [Accessed: May 6, 2013]. [Online]. Available: https://moodle.org/

[10] D. Ben-Naim, "A Software Architecture that Promotes Pedagogical Ownership in Intelligent Tutoring Systems," Ph.D. dissertation, University of New South Wales, Sydney, Australia, 2010.

[11] M. Niederhausen, S. Karol, U. Aß mann, and K. Meiß ner, "HyperAdapt: Enabling Aspects for XML," in Web Engineering, 9th International Conference, ICWE 2009, ser. Lecture Notes in Computer Science, M. Gaedke, M. Grossniklaus, and O. Díaz, Eds. San Sebastián: Springer, 2009, pp. 461–464.

[12] I. Gorton, Y. Liu, and N. Trivedi, "An extensible and lightweight architecture for adaptive server," Softw., Pract. Exper., vol. 38, no. 8, 2008, pp. 853–883.

[13] S. Cheng, "Rainbow: Cost-Effective Software Architecture-Based Self-Adaptation," Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, 2008.

[14] F. Carmagnola, F. Cena, C. Gena, and I. Torre, "MUSE: A Multidimensional Semantic Environment for Adaptive Hypermedia Systems," in Proceedings of Lernen, Wissensentdeckung und Adaptivität (LWA), M. Bauer, B. Brandherm, J. Fürnkranz, G. Grieser, A. Hotho, A. Jedlitschka, and A. Kröner, Eds. Saarbrücken, Germany: DFKI, 2005, pp. 14–19.

[15] PrimeFaces Component Suite. [Accessed: Jan. 31, 2013]. [Online]. Available: http://primefaces.org

[16] E. Knutov, P. De Bra, and M. Pechenizkiy, "AH 12 years later: a comprehensive survey of adaptive hypermedia methods and techniques," New Review of Hypermedia and Multimedia, vol. 15, no. 1, Apr. 2009, pp. 5–38.

[17] O. Aubert and A. Beugnard, "Adaptive Strategy Design Pattern," in Proceedings of The Second Asian Pacific Pattern Languages of Programming Conference (KoalaPLoP 2001), The Country Place, Melbourne, 2001, pp. 1–12.

[18] Webing Research Group Website. [Accessed: May 6, 2013]. [Online]. Available: http://webing.felk.cvut.cz

 ISBN: 978-1-61208-274-5