Adaptivity in GRAPPLE: Adaptation in Any Way You Like

Paul De Bra, David Smits, Mykola Pechenizkiy, Ekaterina Vasilyeva
GRAPPLE Project,
Eindhoven University of Technology (TU/e)
Eindhoven, The Netherlands
grapple@win.tue.nl

Abstract: GRAPPLE is an EU funded IST FP7 project that brings together a group of researchers into adaptive learning technology and environments and developers of learning management systems (LMSs), in order to offer adaptive learning as a standard feature of future LMSs. This paper presents the adaptation engine used in GRAPPLE, and explains why we consider it to be a truly general-purpose adaptive learning environment (ALE). In particular the paper describes how the core *adaptation component* (or *engine*) can be configured to perform any kind of adaptation to any type of XML document, and how it is designed to communicate and work with other systems, in particular with (different) LMSs.

Introduction

After some initial isolated research efforts the field of adaptive technology-enhanced learning (or adaptive TEL) really took off with the publication in 1996 of the seminal paper on adaptive hypermedia by Peter Brusilovsky (Brusilovsky 1996), which was updated in 2001 (Brusilovsky, 2001). A large number of research papers appeared in journals and at conferences such as WebNet/ELearn, Intelligent Tutoring Systems, ED-MEDIA, User Modeling, and later also the dedicated Adaptive Hypermedia conference series. A number of research prototypes of adaptive TEL systems were built, and mostly used only for demonstration purposes, or sometimes also for adaptive course delivery in the authors' institutes. One noteworthy exception is the AHA! system (De Bra et al, 2006), a general-purpose open source adaptive hypermedia platform that has been used by researchers and educators from all over the world[1]. However, to date the use of adaptive technology in learning applications remains very limited. The GRAPPLE project is aimed at changing that by bringing adaptive TEL to the masses. This is done by integrating an *adaptive TEL environment* (henceforth abbreviated to ALE, for *adaptive learning environment*) with major *learning management systems* (or LMSs) using a *service-oriented (web) framework approach*.

GRAPPLE bundles the expertise of researchers from 15 universities, research institutes and companies, including the creators of the adaptive systems AHA! (De Bra et al, 2006), KBS-Hyperbook (Henze et al, 1999), RATH (Hockemyer et al, 1998), APeLS (Conlan et al, 2002) and WINDS (Kravcik et al, 2004), of user modeling languages and services, including UserML (Heckmann et al, 2003), (Heckmann et al, 2005a), GUMO (Heckmann et al, 2005b) and the work of (Van der Sluijs et al, 2006), of experts in learning standards (e.g. the Open University Nederland and Atos Origin Spain), of developers of and contributors to LMSs including Moodle, Claroline and Sakai, and of developers of industrial TEL applications (Atos, Guinti Labs and IMC Information Multimedia Communication AG). The goal of GRAPPLE is to have the ALE become a "standard" component of the LMSs so that the thousands of institutes (world wide) using these LMSs automatically have access to the ALE.

In this paper we first describe why combining an ALE with an LMS is a natural combination of two tools with complementary functionality. (This description is taken from a general overview of GRAPPLE that was published at ED-MEDIA (De Bra et al, 2008) as a poster.) We then describe the ALE used in GRAPPLE, which is a complete redesign of the AHA! system. The redesign specifically aims to turn AHA! into a modular architecture with explicit communication between components, enabling it to interface to other modules of an LMS. We also show how system designers can completely tailor the adaptive functionality of the adaptation engine to fit their needs.

[1] Some noteworthy examples are the AlcoZone alcohol tutorial from Virginia Tech (Bhosale, 2006), an automata theory course from Korea University (Lee et al, 2005) and a programming course from the Slovak University of Technology (Bieliková et al, 2005). We are also aware of on-going work in Brazil, Colombia, and South Africa.

The "marriage" between an ALE and LMS

Many institutes in higher education, but also (large) knowledge-intensive companies, use a learning management system to *manage* the learning process. This management consists of both *administration* of the processes and their outcome and of *facilitating* the learning itself by means of course selection, delivery and evaluation tools. The functions of an LMS include (but there are many more):

- registering (and later perhaps deleting) users, and authorization (login, access restrictions)
- enrollment in courses (or other types of learning modules)
- workflow (task management, notifying learners of assignments that are due, assignment of new tasks after assessment of completed tasks, notifying tutors of completed assignments to be graded, etc.)
- distribution (or delivery) of learning material
- assessment (including multiple-choice tests, but also upload of assignment work for off-line assessment by a tutor)
- portfolio management (certifications, registration of completed courses or course programs)
- etc.

One would expect that the *distribution of learning material* would be very well supported by all LMSs and widely used. However, in a lot of cases this part of the LMS is only used (in practice) to serve documents (complete course texts as a single Word or pdf file, Powerpoint slides, etc.) that are not well integrated with the functionality of the LMS and do not enable fine-grained tracking of the learner's progress. This is where the ALE comes in. It performs the following functions:

- presenting a course text as a website (pages with links, allowing fine-grained tracking of the learner's progress)
- adaptive guidance through link generation, sorting, hiding or annotation
- adaptive page content to automatically compensate for missing prerequisite knowledge
- possibly other adaptive tools like adaptive tests, collaborative filtering (of links), etc.

In order to successfully combine an LMS and an ALE the following types of integration facilities are needed:

- single sign on: when a learner logs on in the LMS, goes to a course sub-site, and then to a course page (s)he must be automatically be logged on in the ALE (and registered if this was not yet the case);
- user model/profile exchange and communication: the ALE must have access to the information the LMS stores about the user (e.g. results of multiple-choice tests, but also previously attended courses or skills and knowledge obtained elsewhere but registered in the LMS); potentially the LMS may need user information from the ALE as well, to record what the user has studied, at a high level of abstraction (whereas the ALE keeps a fine-grained user model).

When an LMS and ALE are properly integrated the learner should not be aware that some of the used services are offered by the LMS and others by the ALE. This can be achieved by having the ALE operate as one of the LMS's tools, presenting its output in one or more frames within the total presentation form offered by the LMS. GRAPPLE aims at realizing such seamless integration between its ALE and different LMSs. It remains to be seen how seamless the integration will be in the end. In an early experiment we have realized (in a collaboration with the University of Linz, Austria) a truly seamless integration between the AHA! system and Sakai. This experiment has shown that such ALE-LMS integration is possible.

The GRAPPLE ALE architecture: AHA! version 4

Figure 1 shows the architecture of the GRAPPLE ALE part (which we will simply refer to as AHA! version 4 or AHA! for short). A prototype of this system is already in (production) use at the Eindhoven University of Technology, serving an adaptive course text on the topic of "hypermedia structures and systems".[2] (The prototype is not yet connected to an LMS.) In this paper we focus our attention on the parts that make *the adaptation in* AHA! highly customizable: the *processors* and *modules*. The remainder of the architecture is highly customizable as well: the choice of *event bus* implementation, *user model service* and *-cache*, *domain model service* and *-cache*, *login manager* and *concept manager* can all be replaced by indicating in a configuration file that a different implementation should be used.

[2] This course can be found at http://wwwis.win.tue.nl/2ID65/ and is open to visitors. You can create your own login or browse the course anonymously, with complete adaptive functionality available.

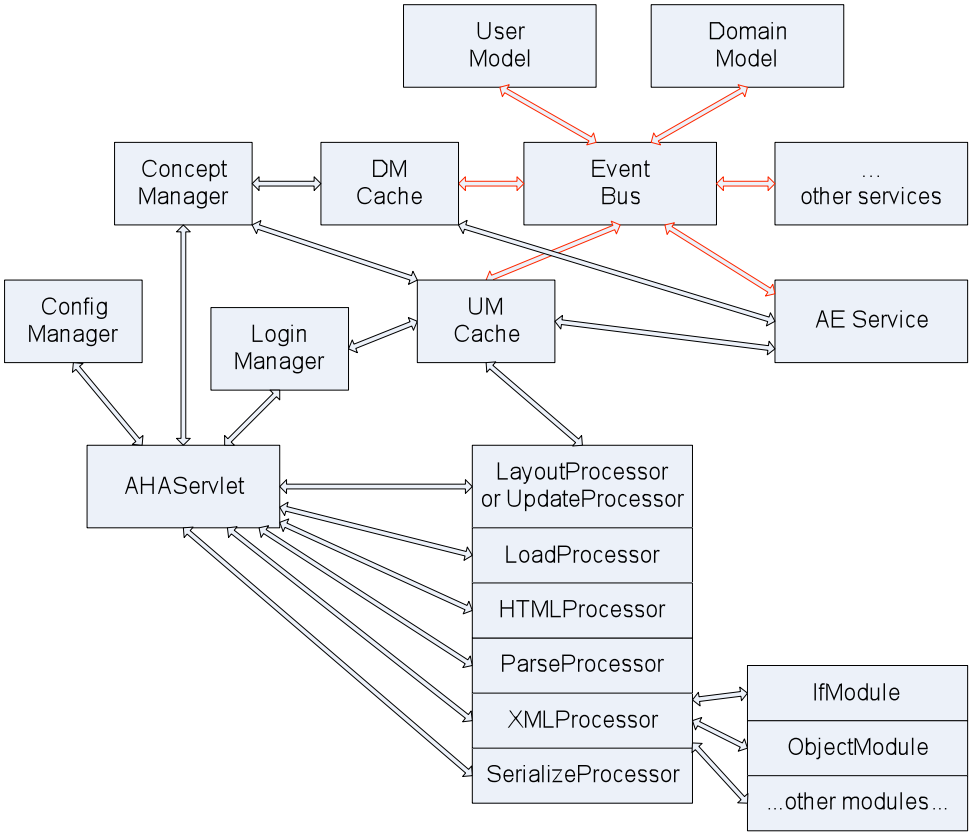

Figure 1: AHA! version 4: the GRAPPLE ALE architecture.

In GRAPPLE an application (or course) is described at a high level using a *Conceptual Adaptation Model* (or CAM). The language used to describe a CAM is still being designed, but initial ideas are described in (Hendrix et al, 2008). In the absence of a "real" CAM (and authoring tools to create it) the current AHA! domain model component is an import module for (already slightly extended) AHA! version 3 structures. The adaptation as seen by the engine (AHA! version 4) consists of *event-condition-action rules* that define user model updates, and *expressions* over the domain model (DM) and user model (UM) to select content, choose link adaptation elements, choose layout, processors and modules to use, etc. Although the *conditions* and *actions* typically evaluate expressions and perform assignments, AHA! allows the expressions and assignments to contain arbitrary Java code, thereby making them very powerful adaptation tools.

AHA! Processors

When AHA! receives a request for a concept, the "Config Manager" merges a hierarchy of configuration files to decide which processors and modules to use for processing the request, and possibly with which configuration options. We look here at two of these processors: the LayoutProcessor and the XMLProcessor.

- When a concept is requested the configuration determines whether to create a *layout* for it or not, and if so, which LayoutProcessor to use for it. (Layout for (X)HTML uses "frames", but for other document types this may be different.) In our hypermedia course for instance we use a condition:

```
<layoutprocessor merge="replace" expr="${c2ID65.introductorypart.done}">
   <code>nl.tue.aha.ae.processor.LayoutProcessor</code>
   <update>true</update>
</layoutprocessor>
```

  This indicates that the layout processor is only used after the "introductory part" is "done". (This is a Boolean attribute of a concept in UM.)
  The layout processor (when it is used will use a configuration (not shown here) to decide which frameset to generate. Each frame is assigned a *view* to be presented in the frame. AHA! has completely generated view

types, for instance presenting a partial table of contents (or fish-eye view) of an application (or course), and it also has a "MainView" view type, used to present an adapted version of a page or resource.
When the frameset is returned (to the browser) it will result in the browser requesting the same concept as in the original request, but now with a view associated with it. This will tell AHA! not to call the layout processor again, and not to update the user model (which was done when the original request was received).

- The LoadProcessor, HTMLProcessor (when needed to convert HTML into XHTML) and ParseProcessor together produce a DOM tree in memory. The XMLProcessor is responsible for performing the adaptation to that DOM structure. It walks through the DOM tree, and when it encounters an XML tag (element) associated with a *module* in the configuration file it calls that module to perform adaptation. (See next subsection.) When the XMLProcessor reaches the end of the DOM tree the SerializeProcessor converts the in-memory structure back into a (textual) XML file to be returned to the browser.

AHA! Modules

AHA! can perform adaptation to any XML format. In the GRAPPLE project we not only consider XHTML but also other XML formats including X3D used for virtual reality, SMIL used for multimedia, etc. In the configuration files (for handling concepts) the <xmlprocessor> part expects a <modules> element that contains the definition of modules that handle certain XML tags (or elements). An example of such a module definition is:

```
<linkmodule merge="replace">
   <code>nl.tue.aha.ae.processor.xmlmodule.LinkModule</code>
   <tagname>a</tagname>
   <condition><jexpr paramnames="element"
                     paramtypes="org.w3c.dom.Element">
      "conditional".equals(element.getAttribute("class"))</jexpr></condition>
   <layoutattribute>class</layoutattribute>
   <conceptattribute>href</conceptattribute>
</linkmodule>
```

Every *module* definition should contain two tags: "code" and "tagname". The XMLProcessor uses these to decide which (Java) code to use for the module and which XML tag is adapted by this module. The name "linkmodule" can be chosen arbitrarily and has no meaning to either the XMLProcessor or the module itself.

The remaining tags within the module definition are passed onto the module code itself, and have no meaning to the XMLProcessor. The default "LinkModule" that AHA! provides will perform the following actions:

- The *condition* is used to decide whether the module will adapt the element or not. In this case we have defined the condition to check whether there is a "class" attribute that has the value "conditional".
- The *layoutattribute* defines which attribute will be generated and/or adapted, in this case the "class" attribute. (This happens to also appear in the *condition* but that's not required in any way.)
- The *conceptattribute* defines which attribute contains the concept to be used for the adaptation. For this module, dealing with links, each link refers to a concept and the adaptation of the link is defined through expressions over the UM attributes for this concept. We have omitted this part of the definition here because of lack of space.

AHA! comes with a number of modules predefined, for instance to handle "if" tags defining conditionally included fragments of text, "object" tags defining conditionally included objects (files), "variable" tags used to insert values from or expressions over the user model (an example is the inclusion of the user's name in a header), a module for generating multiple-choice tests, etc. Other modules can be added at will, and modules can be associated with any XML tag. In our example hypermedia course for instance we use the HTML *imagemap* feature (with client side imagemaps) and we associated the LinkModule with the "area" tag so as to adapt "area" tags just like "a" tags. Of course when the GRAPPLE project will be "delivered" the ALE (AHA!) will be equipped with a very large number of predefined and implemented processors and modules, because we do realize that most authors and application designers will only *configure* the adaptation process and not *develop* new processors and modules to suit their specific adaptation needs.

Conclusions and Future Work

The GRAPPLE project promises to bring adaptive TEL to the masses by incorporating an adaptive learning environment (ALE) in popular learning management systems (LMSs). This ALE is based on a redesign of the AHA! system (called AHA! version 4). Because GRAPPLE is not just a research project but aims to deliver a production-quality ALE a lot of attention must be paid to its flexibility and extensibility so as to enable it to perform adaptation in any way an author or designer may want. Because we as developers of the adaptation engine cannot foresee all possible adaptation desires, we have opted to create an architecture that makes it easy to tell the engine to use different *processors* and *modules* for performing adaptation, and to provide a large variety of configuration options for processors and modules.

Further developments in the GRAPPLE project, before, during and after ELearn 2008, can be followed at www.grapple-project.org.

Acknowledgement

This work is has been performed in the framework of the IST project IST-2007-215434 GRAPPLE which is partly funded by the European Union. The authors would also like to acknowledge the contributions of their numerous colleagues from all 14 GRAPPLE project partners.

References

Bieliková, M., Kuruc, J., Andrejko, A. (2005). Learning Programming with Adaptive Hypermedia System AHA!., *Proc. of ICETA 2005 – 4th Int. Conf. on Emerging e-learning Technologies and Applications*, Slovakia, pp. 251-256.

Bhosale, D. (2006). AlcoZone: An Adaptive Hypermedia based Personalized Alcohol Education. *Master Thesis, Virginia Tech*, available at http://scholar.lib.vt.edu/theses/available/etd-05172006-153545/.

Brusilovsky, P. (1996). Methods and techniques of adaptive hypermedia. *User Modeling and User-Adapted Interaction,* 6 (2-3), pp. 87-129, Kluwer.

Brusilovsky, P. (2001). Adaptive hypermedia. User Modeling and User Adapted Interaction, Ten Year Anniversary Issue, 11 (1/2), pp. 87-110, Kluwer.

Conlan, O., Hockemeyer, C., Wade, V., & Albert, D. (2002). Metadata Driven Approaches to Facilitate Adaptivity in Personalized eLearning systems. *The Journal of Information and Systems in Education*, 1, 38–44.

Cristea, A., De Mooij, A. (2003). LAOS: Layered WWW AHS Authoring Model and its corresponding Algebraic Operators. *Proceedings of the WWW Conference, Alternate Education Track*, pp. 301-310, Budapest, Hungary.

De Bra, P., Pechenizkiy, M., van der Sluijs, K, Smits, D. (2008) GRAPPLE: Integrating Adaptive Learning into Learning Management Systems. *Proceedings of the AACE ED-MEDIA Conference*, pp. 5183-5188.

De Bra, P., Houben, G.J., Wu, H. (1999) AHAM: A Dexter-based Reference Model for Adaptive Hypermedia. *Proceedings of the ACM Conference on Hypertext and Hypermedia*, pp. 147-156, Darmstadt, Germany, 1999.

De Bra, P., Santic, T., Brusilovsky, P., (2003) AHA! meets Interbook, and more... *Proceedings of the AACE ELearn 2003 Conference*, pp. 57-64.

De Bra, P., Smits, D., Stash, N. (2006). The Design of AHA!, *Proceedings of the ACM Conference on Hypertext and Hypermedia*, pp. 133, Odense, Denmark. The adaptive version of this paper is available on-line at http://aha.win.tue.nl/ahadesign/.

Heckmann, D., Krüger, A., (2003). A User Modeling Markup Language (UserML) for Ubiquitous Computing. *In Proceedings of User Modeling 2003, 9th Int.* Conf., Johnstown, PA, pp. 393-397, LNCS, Springer Verlag.

Heckmann, D., Schwartz, T., Brandherm, B., Kröner, A. (2005a) Decentralized User Modeling with UserML and GUMO. *Proceedings of the Workshop on Decentralized Agent Based and Social Approaches to User Modelling* (DASUM 2005), Edinburgh, Scotland, pp. 61-65.

Heckmann, D., Schwartz, T., Brandherm, B., Schmitz, M., Wilamowitz- Moellendorff, M. von. (2005b) GUMO - the General User Model Ontology. *Proceedings of the 10th International Conference on User Modeling*, LNAI 3538, pp. 428-432, Edinburgh, Springer Verlag.

Hendrix, M., De Bra, P., Pechenizkiy, M., Smits, D., Cristea, A. (2008) Defining adaptation in a generic multi layer model : CAM : The GRAPPLE Conceptual Adaptation Model. *Proceedings of the 3rd Europeanl Conference on Technology Enhanced Learning* (EC-TEL), Springer LNCS. (page numbers still pending)

Henze, N., Nejdl, W. (1999). Adaptivity in the KBS Hyperbook System. 2nd Workshop on Adaptive Systems and User Modeling on the WWW, workshop held in conjunction with the World Wide Web Conference (WWW8) and the International Conference on User Modeling.

Hockemeyer, C., Held, T., & Albert, D. (1998). RATH –- A Relational Adaptive Tutoring Hypertext WWW–Environment Based on Knowledge Space Theory. In C. Alvegård (Ed.), *CALISCE`98: Proceedings of the Fourth International Conference on Computer Aided Learning in Science and Engineering* (pp. 417–423). Göteborg, Sweden: Chalmers University of Technology.

Kravčik, M., Specht, M., Oppermann, R. (2004). Evaluation of WINDS Authoring Environment. *Third International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH2004),* pp. 166-175, Eindhoven, LNCS3137, Springer Verlag, 2004.

Keewoo Lee, Hyosook Jung Seongbin Park (2005). Applying adaptive hypermedia technologies to a learning tool. *Fifth IEEE International Conference on Advanced Learning Technologies*, ICALT, pp. 202-204.

Van der Sluijs, K., Houben, G.J. (2006). A Generic Component for Exchanging User Models between Web-based Systems, *International Journal of Continuing Engineering Education and Life-Long Learning*, Vol. 16, Nos. 1/2, p. 64-76, Inderscience.