GRAPPLE: Integrating Adaptive Learning into Learning Management Systems

Paul De Bra, Mykola Pechenizkiy, Kees van der Sluijs, David Smits
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 papers presents the overall architecture of GRAPPLE, and explains some of the main challenges in creating a truly general-purpose adaptive learning environment (ALE) that can be used with different LMSs and a distributed architecture of user modeling services. In particular the paper describes the *conceptual adaptation model*, its translation into *adaptation rules*, and some technological challenge that results from separating the concerns of *adaptation* and *user modeling*.

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 14 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, 2005) 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. We then give an overview of the GRAPPLE architecture and briefly describe the different components. We emphasize the adaptation engine and user modeling infrastructure and discuss the separation of these two components.

[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 architecture

Figure 1 shows the overall architecture of the GRAPPLE ALE part. Although inspired by past development on the AHA! system a significant architectural difference is the separation of the *adaptation engine* from the *user modeling service*. In GRAPPLE a learning application is represented at the conceptual level through a *conceptual adaptation model* or CAM. In terms of the AHAM reference model (De Bra et al, 1999) the CAM represents the Domain Model (DM) and the Adaptation Model (AM), still called the "Teaching Model" in (De Bra et al, 1999). In the CAM concepts are "connected" with each other at different (semantic) levels. Unlike in the AHAM-based LAOS model

(Cristea et al, 2003) which has 5 fixed levels, in the GRAPPLE CAM there may be arbitrarily many levels, and they may be different in each learning application. Typically the DM will correspond closely to a *subject ontology* (the subject domain of the learning application or course). There can be several levels related to adaptation. A *task* or *goal* level may relate tasks or learning goals to sets of concepts that need to be studied in order to achieve a goal or to be able to perform a task. A *prerequisite* level may connect pairs of (sets of) concepts to indicate that the *prerequisite* concepts should be studied before the dependent concepts. This may be used to perform link hiding or annotation, but also to automatically include *prerequisite explanations*. A *process* or *sequencing* level may indicate desired sequences of concepts to be studied in a fixed order, or a lattice offering limited freedom in navigating through the concepts. This may be used to generate a menu-like structure of links to the concepts, or a "next" button for sequential navigation. Several conceptual adaptation structures in the CAM will use and possibly define how to update the *user model* (UM). The language used to describe a CAM is still being designed. It will be inspired by elements from IMS-LD (IMS, 2003) and from the LAG language used in the LAOS research. Even though this is a high-level language, independent of actual adaptation engines to be used, authors cannot be expected to define the desired adaptation in any "technical" language. Therefore a graphical authoring tool for CAMs will be built to make the creation of CAMs possible without any technical knowledge of the CAM language. The use of the Graph Author in AHA! has already shown that with such a graphical tool authors can define reasonably complex adaptation. (We have been using AHA!'s Graph Author for a number of years already in a course on adaptive hypermedia.)

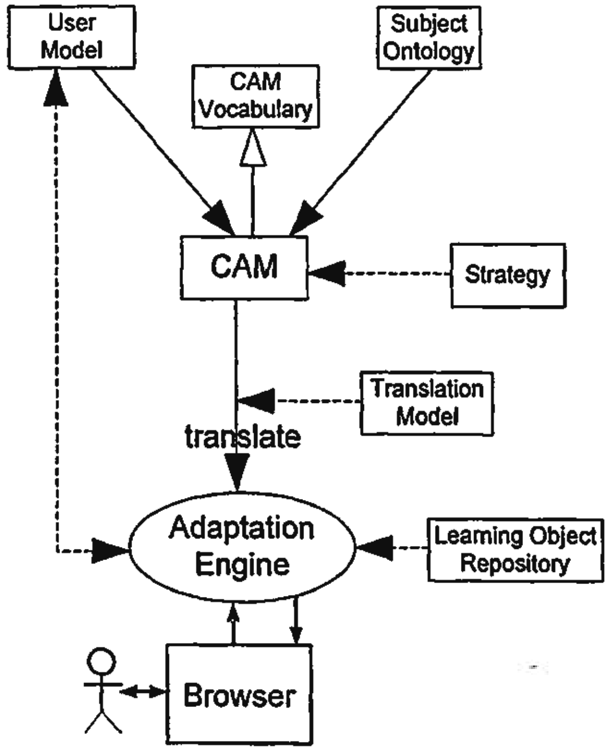

Figure 1. The GRAPPLE ALE architecture.

The CAM of a learning application needs to be translated to actual *adaptation rules* for the ALE. For instance in the CAM one may express that concepts A and B are a prerequisite for concept C. This does not imply any specific adaptation that needs to be performed by the adaptation engine. It also does not indicate whether there are (conditional) links from A and/or B to C. Depending on what the author wants a *translation model* can be selected that defines a translation to the low level rule language supported by one of the supported adaptation engines . Such a low level translation could for instance state that concept C is considered *suitable* when the knowledge of A and the knowledge of B both exceed 70%. So the low level rule could be something like

C.suitability := (A.knowledge > 70) and (B.knowledge > 70).

Figure 1 suggests that the rules are all executed by the adaptation engine. However, in reality there are two types of rules:

- *user model update rules*: from the CAM we may deduce that when a user accesses a specific concept the knowledge value of the user for this concept should be increased in UM. This increase may imply a (possibly smaller) change to the knowledge value of some other concepts as well, mainly to higher-level concepts (in the subject ontology). Although in principle the adaptation engine could execute these rules the resulting process would be very inefficient because it would involve numerous queries and updates to the UM, which would involve a lot of inter-process communication at the query language level. Therefore it is better to execute these rules within the UM (the "User Model" part of the architecture). Also, *prerequisites* typically define that some concepts become *suitable* when enough knowledge is obtained about some other concepts. The satisfaction of prerequisites can be determined efficiently by the UM part whereas the adaptation engine would need numerous queries to UM (for the knowledge of prerequisite concepts) to determine the suitability of concepts. (We come back to this choice later.)
- *adaptation rules*: from the CAM we may also deduce that when a user accesses a specific concept we need to retrieve a certain *resource* or *learning object* (LO) to be presented. The rules to determine which resource to present (or to query the LO repository for) are executed in the adaptation engine. Likewise, the conditional inclusion of *prerequisite explanations* and the actions of *link hiding* or *link annotation* and the generation of adaptively sorted (link) menus and guided tours are done by executing rules in the adaptation engine.

The separation between UM and the adaptation engine is necessary in GRAPPLE because UM is no longer an integral part of the ALE but is shared between the ALE and the LMS. What looks like a simple (black) box in the architecture may be a distributed infrastructure of services that all know something about the learner. In principle it is possible to connect not just two user models (from the ALE and LMS) but to connect arbitrarily many services, with different owners. Profiles that users create on different on-line sites can be queried and possibly even updated based on actions of the user in the learning environment (given that privacy issues are taken care of, which is a different line within our research). Our experience shows that to generate a single page within a typical application in an ALE many values have to be queried from the user model. In order to ensure reasonable performance of the overall system the adaptation engine will therefore cache UM information for currently active users. However, in the next section we will explain that such caching is not enough, and that the separation of the UM part from the adaptation part leads to inherent performance problems that need to be tackled.

The interaction between adaptation and user modeling

Learners use the ALE component to get page by page access to learning material (e.g. a course text). Some of the adaptation is based on *stable* properties of the user, like the user's *learning style*. The adaptation may then consist of the selection of a textual or graphical presentation (verbal/visual style) or a change in presentation order of examples, explanation, theory and activities (activist/reflector style). The adaptation engine can query the UM infrastructure or broker once, and cache the values obtained.

For adaptation to current knowledge values and for instance some associated prerequisites the situation is more complicated: each action of the user may cause one or more knowledge updates, which in turn may result in one or more prerequisites becoming satisfied. In a tightly integrated system like AHA! a single engine, with an in-memory copy of the user model, can compute the knowledge updates and the satisfaction of prerequisites in a (small) fraction of a second, before sending the adapted page to the learner. All user model updates are thus performed before the page is sent to the learner's browser, without causing any noticeable delay. In the distributed GRAPPLE scenario this is no longer possible:

- When an event is signaled (by the engine) to UM it is unclear when all the UM updates will be performed and thus when UM is ready for querying (the most recent up to date state). Some form of locking can of course be used to temporarily block queries. But because of the distributed nature of UM (split at least between the ALE and LMS) the updates may not be ready in a fraction of a second (like in AHA!).
- When the generation of a page or a navigation menu involves checking the suitability of the destination of each link this involves too many queries to UM to wait for the answers before generating the (adapted)

page. A solution is for UM to signal updates on a common *event bus* and for the adaptation engine to cache the user model. By means of caching an adapted page can be generated immediately, but after the page is sent to the learner some suitability values may still change. By using AJAX technology the resulting changes in the presentation can be sent to the browser "later", while the learner is studying the page. Careful evaluation with learners is needed to determine how this can be done in a way that does not disturb the learner. We expect (but this must be validated) that when a word or phrase changes into a link (because the link destination is determined to be suitable after all) this might not be disturbing. Removing a prerequisite explanation (deemed no longer nccessary) may be ill advised though.

Clearly the aspect of asynchronicity between the processes of user modeling and adaptation needs careful investigation as it will have considerable impact on the overall usability of the adaptive delivery of learning material.

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 integration however is not as easy as it looks: the aspects of *user modeling* and *adaptation* are necessarily split between different components, resulting in foreseeable performance problems. Because GRAPPLE is not just a research project but aims to deliver a production-quality ALE a lot of attention must be paid to performance issues right from the start.

GRAPPLE also emphasizes the usability of the ALE for authors (or course designers). Therefore authoring is done using comfortable graphical authoring tools (that will be designed with existing tools such as the Graph Author of AHA! in mind). Actual low level adaptation rules will be generated through templates (in a Translation Model), thus making the high level Conceptual Adaptation Model (CAM) completely system-independent and thus a candidate for going through a standardization process.

Further developments in the GRAPPLE project, before, during and after ED-MEDIA 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., 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., 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. (2005) 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. (2005) GUMO - the General User Model Ontology. *Proceedings of the 10th International Conference on User Modeling*, LNAI 3538, pp. 428-432, Edinburgh, Springer Verlag.

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.

IMS Global Learning Consortium. (2003). IMS Learning Design Information Model.

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.