GRAPPLE: Personalization and Adaptation in Learning Management Systems

Paul De Bra[1], David Smits[1], Kees van der Sluijs[1], Alexandra I. Cristea[2], Maurice Hendrix[2]
and many other members of the GRAPPLE Research Team
GRAPPLE Project,
Eindhoven University of Technology (TU/e)
Eindhoven, The Netherlands
grapple@win.tue.nl

Abstract: Learning Management Systems such as Blackboard, Moodle, Sakai and Claroline focus on supporting the learning *process* at a fairly global level of courses and tests. Recent additions focus on interaction (discussion forums, chat rooms and wikis). GRAPPLE tackles an important omission: integrating the adaptive delivery of course material into the supported learning process. 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 overall architecture of GRAPPLE, and then describes how to create and deliver adaptive course material through GRAPPLE, on any (supported) LMS.

Introduction

Support for learning is highly concentrated on the *management* and the *process*. Many (perhaps even most) educational institutions like schools, colleges and universities are using a Learning Management System (LMS) to *manage* or keep track of the students' progress through the curriculum. For the institutes it is very important to know who is enrolled in which courses, what their exam results and credits are, and thus also when the students are ready to graduate. Especially in the open-source LMSs a lot of effort has also been invested in the creation of tools to support the learning *process*. Public and private messages, discussion forums and chat rooms for communication and wikis for knowledge sharing, are all aimed at supporting the *learning* rather than the *management*. However what is missing is good support for delivering (prepared) learning material to the students. Many courses have a repository of files: text files, PowerPoint, perhaps video lectures. In some cases it is possible to access a (remote) website while passing the learner's identity. However these external applications cannot communicate back to the LMS. The GRAPPLE project is focused on the *seamless* and *two-way* integration of adaptive course material into the learning process.

Adaptive technology-enhanced learning (or adaptive TEL) has been the main application area for *adaptive hypermedia*. The seminal papers by Peter Brusilovsky (Brusilovsky 1996, Brusilovsky, 2001) have been a source of inspiration for many researchers into personalized information access. Over the years many new methods and techniques were developed, as summarized in (Knutov et al, 2009). Many special-purpose tools were developed and also one general-purpose adaptive system: AHA! (De Bra et al, 2006), that has been used by researchers and educators from all over the world[3]. However, to date the use of adaptive technology in learning applications has remained rather limited because it was never *integrated* into the *learning management systems* used by the schools and universities.

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 authoring systems and adaptation languages (Cristea et al, 2009), 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, Giunti Labs and IMC Information Multimedia Communication AG). The goal of GRAPPLE is to have the *adaptive learning environment* (or 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 the overall GRAPPLE architecture, emphasizing *how* the ALE and LMS can work together to offer an integrated adaptive learning solution. We then describe the process of creating adaptive course material and using it from within the LMS. We describe the conceptual authoring tools and the different ways to create content. The tools are going through a continuous process of improvement based on input from evaluation questionnaires given to the early adopters of the GRAPPLE technology.

Communication between an ALE and LMS in GRAPPLE

The standard way to set up an LMS environment involves creating user accounts (for teachers and learners) on the LMS. Applications other than the LMS have no access to these accounts. One possible exception is that some LMSs have a provision for accessing another application from within the LMS and passing the user id. At the TU/e we have used this to give students with an account on our Sakai LMS transparent access to the adaptive course text for our course 2ID65 on hypermedia. Unfortunately the "Link Tool" that is used (at TU/e within Sakai) to provide this access is insecure. Furthermore the access is one-way only: from Sakai students can access the adaptive course text but the interaction with the course text cannot be communicated back to Sakai, for instance to have the progress of the student recorded in his/her grade book. Also, tests or exams the student may perform in the LMS cannot influence the adaptation done by the course text as only the student id is passed on.

In GRAPPLE we have arranged for a multi-way communication between LMSs and instances of the ALE through the GRAPPLE User Model Framework (GUMF). An LMS can tell GUMF that a student has performed a test (and what the grade was) and GUMF can provide that information to the ALE. The adaptation can then take the test results into account. Likewise, the ALE can tell GUMF about the progress of the student and this can be communicated with the LMS, perhaps to decide when to enable access to a test or assignment. An infrastructure using an asynchronous event bus is used to facilitate the communication between all GRAPPLE components.

GRAPPLE is intended to support *life-long learning*, meaning that GUMF must be able to share information about a learner with different LMSs. When the learner moves to a different school, university or company, the information a learner decides to disclose must be accessible for a new LMS. Therefore GRAPPLE is designed in a system-independent way. This can be done through the project-specific "GRAPPLE statements" but also through the IMS Learner Information Packaging (IMS LIP[4]) standard. Within GRAPPLE five different LMSs can communicate with each other and with the GRAPPLE ALE (GALE). These LMSs are the open source LMSs Moodle, Sakai and Claroline, and Elex from Giunti Labs[5] and Clix from IMC[6]. Figure 1 shows the overall architecture of GRAPPLE.

As can be seen from Figure 1 all GRAPPLE components communicate with each other through a shared GRAPPLE Event Bus (GEB). This is an asynchronous communication bus, meaning that components can send data or requests to another component through the bus and can listen for answers but cannot just wait for an answer to their requests. As a consequence, components should be designed to continue their normal operation without the answers to their requests; the answers will arrive asynchronously. This is an important issue for the adaptive delivery of course material as GALE may not always be immediately notified when a student completes a test on the LMS, and may thus not immediately perform adaptation according to that changed user model. GALE has an internal user model store to enable it to offer adaptivity based on interactions with GALE almost instantly.

The GRAPPLE environment is distributed by design. The Authoring Tool (GAT), The Adaptive Learning Environment (GALE), the User Model Framework (GUMF), the Event Bus (GEB), the LMSs and additional tools such as special visualization tools (GVIZ) may all be running on different servers anywhere on the Internet (and in the project setup they actually do run on servers in different countries). In order for users to be known (and

[1] Paul De Bra, David Smits and Kees van der Sluijs work at the Eindhoven University of Technology, the Netherlands.
[2] Alexandra I. Cristea and Maurice Hendrix work at the University of Warwick, UK.
[3] 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.

[4] http://www.imsglobal.org/profiles
[5] http://www.giuntilabs.com
[6] http://www.im-c.com

authorized) on all systems we use Shibboleth[7] as a single-sign-on facility. In the remainder of this paper we concentrate on the use of GAT and GALE for creating an adaptive course text.

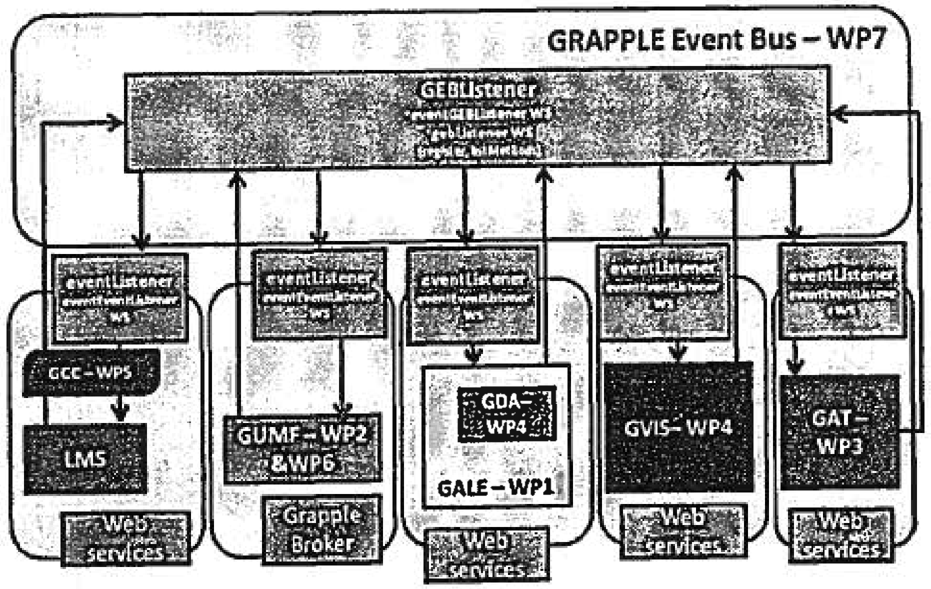

Figure 1: Global GRAPPLE Architecture

Creating the conceptual adaptation model for a course

Figure 2 gives an overview of the authoring process in GRAPPLE as far as the conceptual structures are concerned. We concentrate on the part that is within the dashed rectangle. Creating an application (not considering the actual content in the form of pages) consists of the following steps:

1. An author first creates a conceptual domain model (CDM, or DM for short) for the application, consisting of *concepts* and *concept relationships*[8]. Relationships are named binary connections between concepts. The relationships have a meaning within the application (course) domain, but they have no associated adaptation behavior because they are not *pedagogical* relationships. To some extent a domain model can be considered as an *ontology*, although typically it is smaller and more easy for teachers to create, as it only describes what is used in the course. Figure 3 shows a partial domain model in the DM editor of GAT. It shows an example application called "Milkyway" that we also use to illustrate other parts of the authoring process in this paper (and in other presentations). In the Figure we see the Sun, planets and moons and relationships like "isMoonOf", connecting moons with their planet, and "typeOf" connecting moons with the abstract concept "Moon" and connecting planets with the abstract concept "Planet". Some parts of the model in this figure are incomplete: "isPlanetOf" relationships between planets and the Sun are still missing, as are some moons. Each concept in DM can have an arbitrary number of *properties* with values, and *resources* (references to files) with properties and values. The properties can be used by the adaptation engine, for instance to select a resource to show depending on some user model properties.
2. The author then needs to define a *pedagogical* structure over the DM, indicating recommended ways to study the course. Using the CRT editor an author can create *Concept Relationship Types*, possibly but not necessarily associated with relationships that exist in DM. Some CRTs are predefined, like *start* (for the start concept of a course), *prerequisite* that indicates that some concepts should be studied before some

[7] See http://shibboleth.internet2.edu/ for details on Shibboleth.
[8] SDM and VR-DM in Figure 2 refer to *simulations* and *virtual reality* extensions, not described in this paper.

other concepts, and the *knowledge propagation* CRT that indicates that knowledge obtained by studying course pages should be propagated up the *concept hierarchy*, typically consisting of chapters, sections, subsections and pages. An author may thus not need to define new CRTs to achieve the desired adaptation.

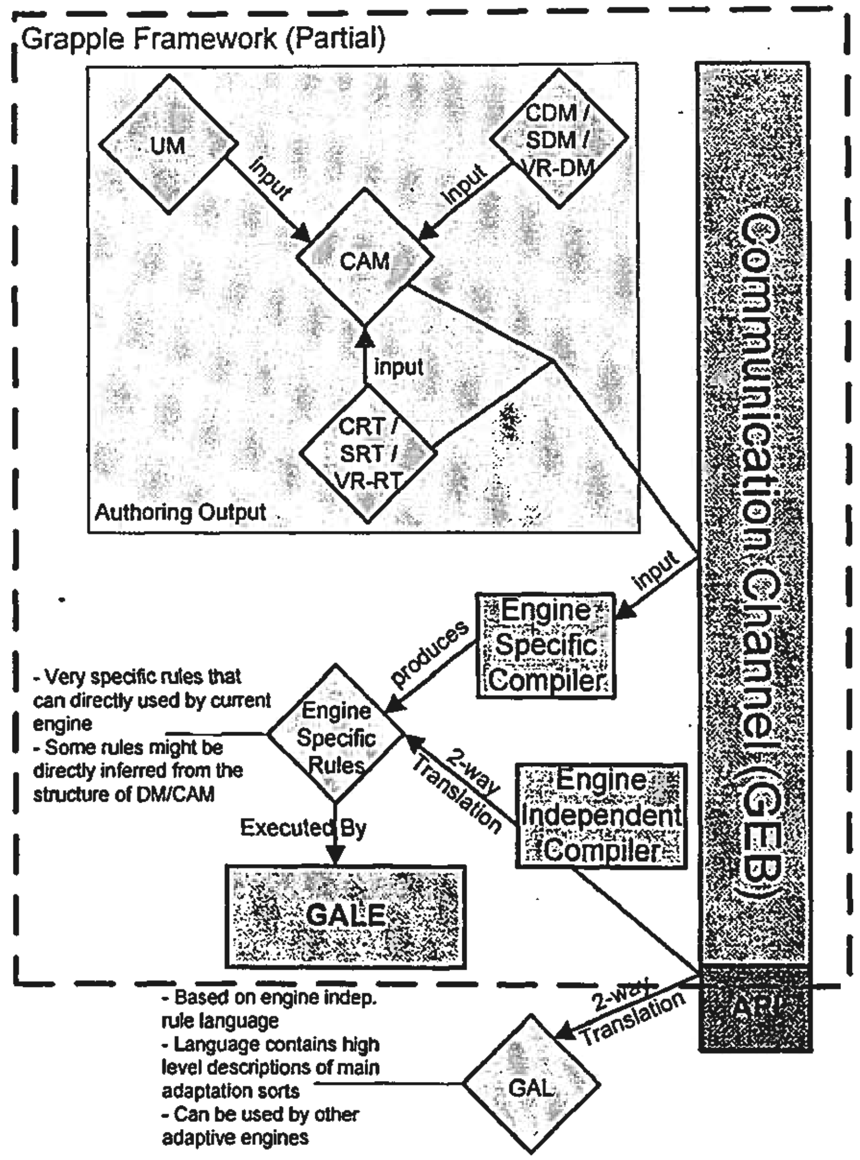

Figure 2: Authoring of conceptual structures in GRAPPLE

3. Concepts from DM are connected using CRTs to form the *Conceptual Adaptation Model* (or CAM) (Hendrix et al, 2008). In past research on AHA! (De Bra et al, 2006) and in other systems as well the CRTs typically were either unary or binary relationships between single concepts. In GRAPPLE a concept relationship can connect any arbitrary number of concepts or even sets of concepts. For this it uses *sockets* each containing one or more concepts. Figure 4 shows a screen shot of a first part of the CAM for the Milkyway example. Allowing sockets with multiple concepts allows for the creation of CRTs with associated *adaptation rules* that take all concepts of a socket into account. Also, it greatly reduced the number of relationship (instances) that need to be created.

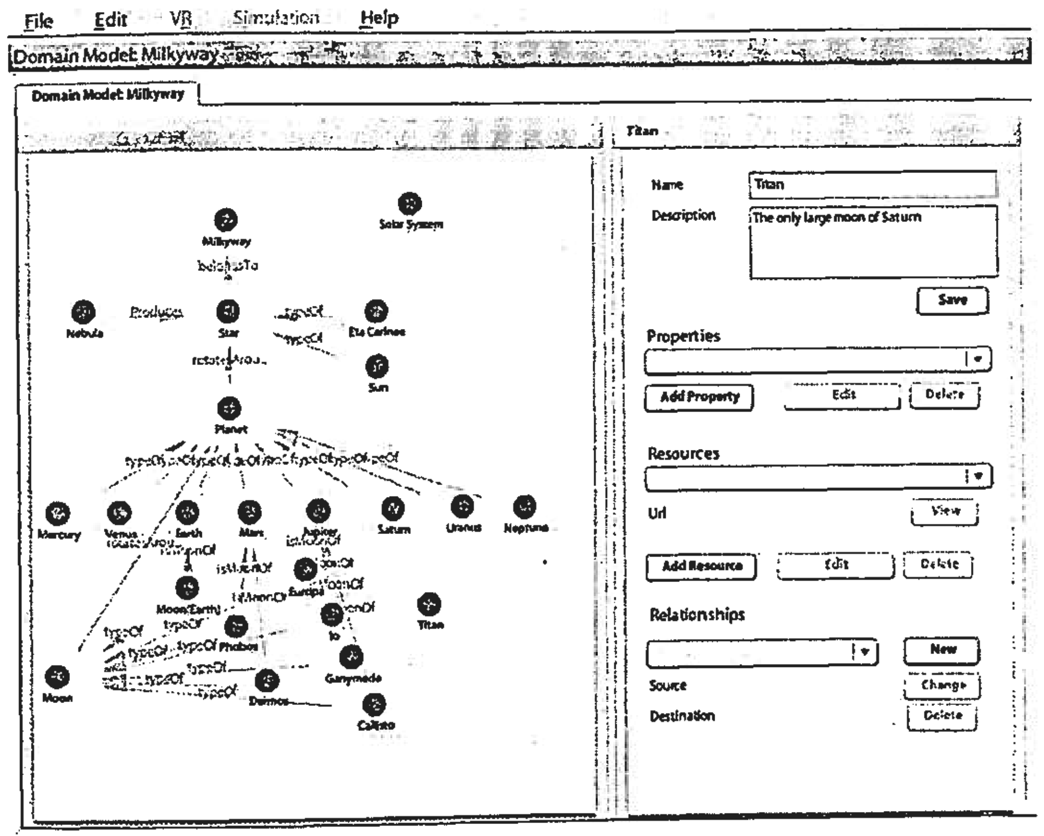

Figure 3: the DM editor, showing the Milkyway example.

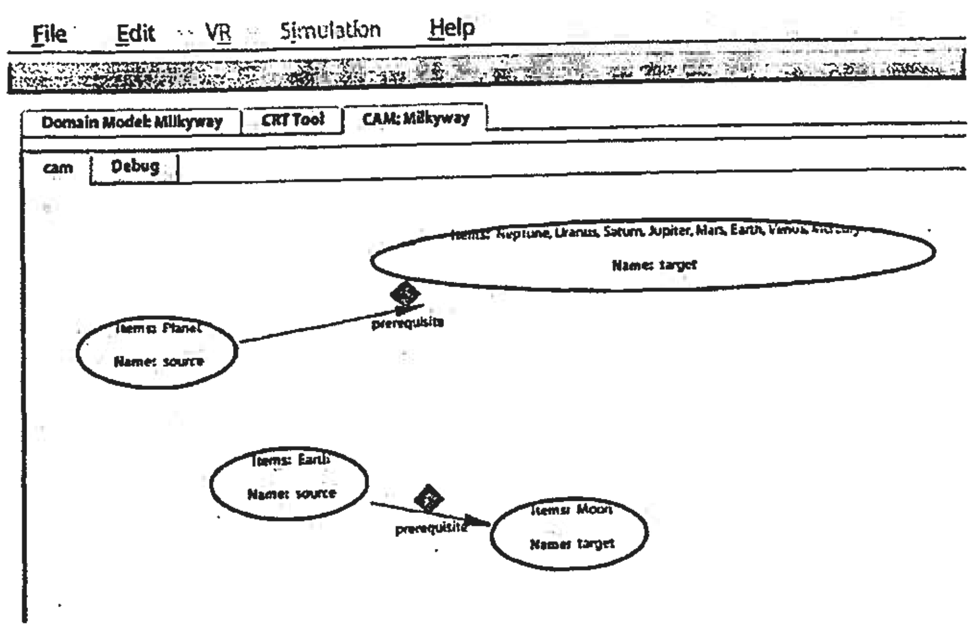

Figure 4: the CAM editor, showing two prerequisites in the Milkyway example.

4. When the CAM has been completed the author selects the "Deploy" function (from the File menu). This sends the CAM to the GALE engine through what Figure 2 calls the "Engine-Specific Compiler"[9]. The CAM includes the DM (or DMs) used in the CAM and also the definition of the CRTs used in the CAM. The compiler creates GALE-concepts and relationships from the DM description and translates *template* adaptation code specified in the CRTs into *default* code for initializing user model variables and *event* code for associating user-initiated events such as a page access or system-generated events such as a user model update with code to generate user model updates (possibly defining adaptation behavior). We illustrate the adaptation code with an example below. Please note that an author who uses predefined CRTs does not need to write or even see such code, and that the majority of authors will only use the predefined CRTs.
   In the Milkyway example we indicated that "Earth" is a prerequisite for "Moon". The prerequisite CRT has the following adaptation code:

   ```
   %target% { #suitability & !`${%source%#knowledge}>80` }
   ```

   This code specifies that the `suitability` attribute of each concept in the "target" socket should be initialized not with a plain value but by executing (hence '!') the code `${%source%#knowledge}>80`. This bit of code is replicated for each concept in the "source" socket and if there are several such concepts the expressions are combined using the logical and operator ('&'). For our example prerequisite this means that the `suitability` attribute of concept "Moon" will be initialized with the value that is the result of evaluating the expression `Earth#knowledge>80`.

One remaining aspect we have not covered is that of the "location" of user model information. In the CRT tool an author defines that a CRT uses some user model attributes (for concepts appearing in a specific socket). Normally the (run-time) values of these attributes of concepts are only known by the GALE engine with which the learner interacts directly. The values are used for generating adaptation, for instance based on the learner's knowledge of some concepts, in combination with the existence of prerequisite relationships. However, the author may indicate that some attributes are *public*. When an attribute is *persistent* (i.e. stored and not calculated on the fly like the `suitability` in the example above) and *public*, GALE will communicate all changes of this attribute to GUMF, making the up to date value available to other GRAPPLE components (like an LMS or perhaps also another GALE server serving another course). When an attribute is *not persistent* and *public*, GALE will request the value from GUMF and will register a *listener* on the event bus to stay informed of updates of that attribute. Other than having this specification (public or not, persistent or not) the author need not be concerned with the communication with GUMF when defining the CAM or creating the content of a course.

Creating content for a GRAPPLE course

Each concept in (the DM of) a course can be associated with one or more *resources*. Typically a resource is a file (an HTML page) that forms a possible presentation of the concept. Having multiple resources allows for the automatic selection of one of several alternative presentations, for instance a version for a beginner and a more advanced learner. The adaptation rules for selecting the appropriate version can be specified at the conceptual level using the CRT tool.

In an online course we expect consistency in the presentation format. In GRAPPLE there is an additional constraint that the presentation must also fit "within" the LMS presentation structure. To this end GALE always generates a single HTML page (at least if the input is also (X)HTML), using tables for different parts of its presentation. Achieving presentation consistency is done by means of two complementary techniques: *layout definitions* and *page templates*.

1. In GALE concepts can *extend* other concepts, meaning that they inherit attributes from these other concepts. An *extends* relation between concepts is used for this. The standard "admin" application contains a concept "_layout" that has a single attribute `layout` with the following default value:

   ```
   <struct cols="20%;*"><viewname="static-tree-view"/><content/></struct>
   ```

   Every concept that extends the "_layout" concept will be presented using a navigation menu for the course on the left (in a column taking up 20% of the width) and the content page on the right (in the remaining 80%). GALE has a number of predefined "views" to show part of the DM of an application. In the standard

[9] The GRAPPLE architecture is designed to be flexible and allow other ALEs to be connected to the architecture, though developing other ALEs is outside the scope of the project.

layout the `static-tree-view` is used, which shows an accordion menu where the top-level concepts are shown and the part of the concept hierarchy containing the current concept is expanded (and the remainder of the concept hierarchy is collapsed).

2. For the pages themselves, there are also two main approaches: creating each page "from scratch" (writing pages or importing "ready-made" pages from a content authoring system, such as the MOT adaptive hypermedia authoring system (Foss & Cristea, 2010)) or using *template pages* The hypermedia course 2ID65 at the TU/e describes many different topics. The structure of different pages or parts of the course is different. Each page in this course is therefore authored separately. The only common aspects of the presentation are the common layout (and of that also two versions are used in this example). The left part of Figure 5 illustrates this authoring approach.
The Milkyway example presents some information about celestial objects. The actual content is copied from Wikipedia. The presentation of similar objects is identical. Figure 6 shows cropped screendumps from the pages about Jupiter and Saturn. They start with a title, then describe the position of the object in the solar system ("Is Planet of: Sun"), then gives an image, followed by textual information, and finally there is a list of moons of the planet. It is reasonable to expect that any change to the presentation an author may come up with should be applied to the pages about each planet. Therefore we associate the concepts with a common *resource* which is the template page. Every item that is "planet-dependent" is either the value of a *property* of the concept or is contained in a file of which the URL is a property of the concept. GALE offers some xml tags that can be used in pages (using both the xhtml and gale namespaces) to include such information:

- The `<gale:variable>` tag is used to present the value of a user model attribute or the result of an expression over (domain- and) user model. Two examples:

  ```
  <gale:variable name="gale://gale.tue.nl/personal#name" />
  ```

  will be replaced by the name of the user. (`personal` is a pseudo-concept that represents the user)

  ```
  <gale:variable expr="${?title}"/>
  ```

  will be replaced by the title property of the current concept.
- The `<gale:object>` tag is used to include a file in place of the object. The file must contain a valid xml fragment. The html `<img>` tag can also be used to include a file which must be an image. In order to determine the source for the object or image from the domain or user model the `<gale:attr-variable>` tag can be used. This is a "trick" because inside an xml tag no xml tags can be used.

  ```
  <img width="300"><gale:attr-variable name="src" expr="${#image}"/>'
  ```

  will load the image of which the URL is obtained by evaluating the expression that retrieves the value of the `image` attribute of the current concept.

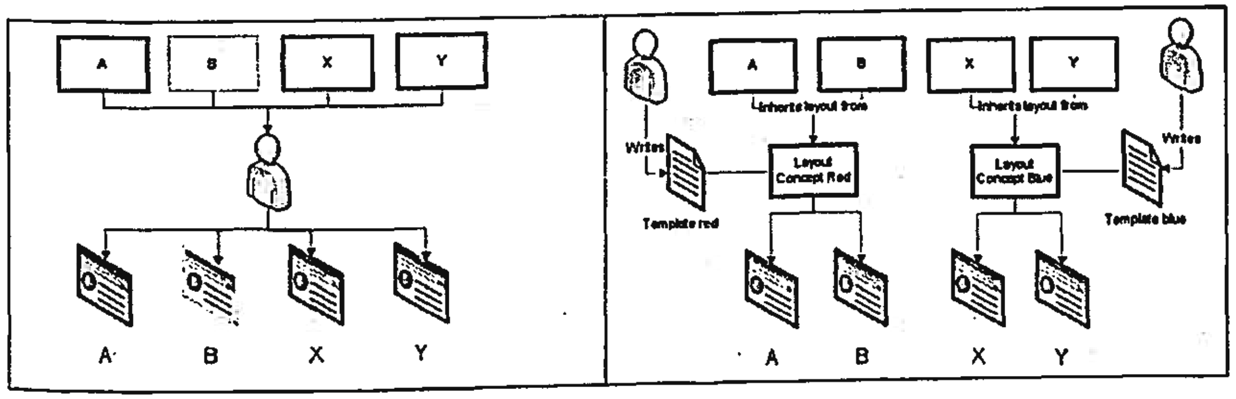

Figure 5: Authoring pages separately (left) or using *template* pages (right)

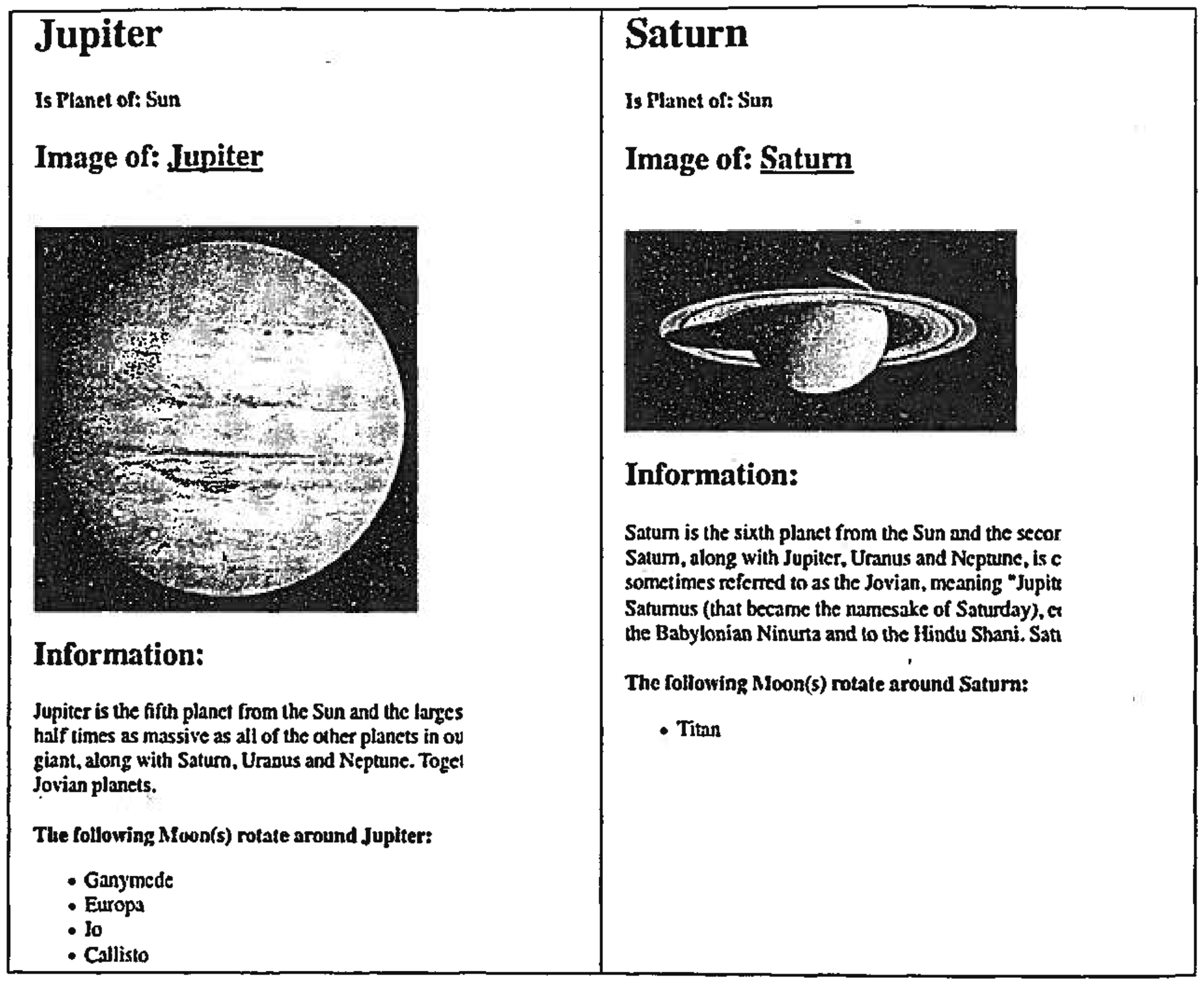

Figure 6: Consistency in the presentation of different concepts, achieved through template pages.

Conclusions and Future Work

The GRAPPLE project attempts to bring adaptive course delivery to the masses using a three-pronged approach:

1. it incorporates the adaptive learning environment (ALE) in popular learning management systems (LMSs),
2. it offers easy-to-use graphical authoring tools for the conceptual adaptation model (CAM) and
3. it offers a powerful adaptation engine that allows most popular adaptation techniques for both content and navigation structures.

Although the project and the software development are still ongoing, training events have already been held in academic and industrial environments. Each event is followed by an evaluation (questionnaire) from which suggestions for further improvement are extracted. Students in a course on adaptive systems are currently using the authoring tools (GAT) and adaptation engine (GALE) in group assignments in which they are creating adaptive course material. GALE is also already in use for course delivery at the TU/e and has proven to have high performance and reliability.

Although GAT+GALE are currently actively being used and evaluated, the two-way communication between GALE and different LMSs is still undergoing testing. Also, the use of Shibboleth for the single sign-on facility is still being tested.

Further developments in the GRAPPLE project, before, during and after ED-MEDIA 2010, 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 15 GRAPPLE project partners.

References

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.

Cristea, A. I., Smits, D. Bevan, J. & Hendrix, M. (2009) LAG 2.0: Refining a reusable Adaptation Language and Improving on its Authoring. *Proceedings of the 4th European Conference on Technology Enhanced Learning* 2009, pp. 7-21, Springer LNCS, Subseries: Programming and Software Engineering, Vol. 5794, Nice, France, September 14, 2009.

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.

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/.

Foss, J. K. & Cristea, A. I. (2010) Transforming a linear module into an adaptive one: tackling the challenge. *Proceedings of the 10th International Conference on Intelligent Tutoring Systems*, Springer LNSC, Pittsburgh, Pennsylvania, USA (to appear).

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.

Hendrix, M., De Bra, P., Pechenizkiy, M., Smits D. & Cristea, A. I. (2008) Defining adaptation in a generic multi layer model: CAM: The GRAPPLE Conceptual Adaptation Model, *Proceedings of the 3rd European Conference on Technology-Enhanced Learning 2008*, pp. 132 - 143, Springer LNCS, Maastricht, The Netherlands.

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.

Knutov, E., De Bra, P., Pechenizkiy, M. (2009). AH 12 years later: a comprehensive survey of adaptive hypermedia methods and techniques.

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.

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.