Adaptive Moodle:

Micheál Tiarnaigh.

B.A. (Mod.) CSLL

An integration of Moodle (Modular Object-Oriented Dynamic Learning Environment) with an AHS (Adaptive Hypermedia System).

Final Year Project, May 2005

Supervisor: Dr. Vincent Wade
Declaration

I hereby declare that this thesis is entirely my own work and that it has not been submitted as an exercise for a degree at any other university.

________________________________________________  April 2005

Micheál Tiarnaigh.
Acknowledgements

Sincere thanks to my project supervisor, Vincent Wade, who was a welcome source of inspiration and direction during the course of this project. Thanks also to Ian O’Keefe, Owen Conlan and Alex O’Connor who have helped me out on more than one occasional with some rather annoying technical issues.

My thanks also to Dr. Carl Vogel for his unerring support during my time in college and for the tremendous amount of work and dedication he puts in for everyone involved in this course.

Finally, and most importantly, I would not have got to this stage without the love, support (both financial and moral) and occasional prod from my parents, and I am indebted to them for having faith in me at all times.
“Il faut imaginer Sisyphe heureux”
Albert Camus.

"The eagle never lost so much time as when he submitted to learn from the crow"
Nobody, Dead Man (1995) (41:29)

"A prison has not been built yet that can hold me. And I'll get out of this one even if it means spending my entire life here!"
Virgil Starkwell (Woody Allen), Take The Money And Run.

"Colourless green ideas sleep furiously"
Noam Chomsky.

“America is the only country that went from barbarism to decadence without civilization in between.”
Oscar Wilde

“All animals are equal, but some animals are more equal than others”
George Orwell, Animal Farm (1945).

"AI is the science of making machines do things that would require intelligence if done by humans."
Marvin Minsky
# Table of Contents

0  Abstract .................................................. 6

1  Introduction ............................................. 7
  1.1  Motivation ............................................ 7
  1.2  Objectives ............................................ 8

2  State of the Art ......................................... 10
  2.1  CMS ..................................................... 10
  2.2  AHS ..................................................... 14

3  Requirements & Design .................................. 22
  3.1  Key information-sharing points .................. 22
  3.2  Different approaches ................................ 24

4  Implementation & Experimentation (Evaluation) .... 32
  4.1  Trial of the final integrated system ............. 32
  4.2  Evaluation ............................................ 41

5  Conclusions & Future work ............................ 43
  5.1  Overall achievements ................................ 43
  5.2  Future work .......................................... 44
  5.3  Concluding remarks .................................. 44

6  Bibliography .............................................. 46
Abstract.

This paper discusses an integration of two separate software systems, namely the CMS Moodle (Modular Object-Oriented Dynamic Learning Environment) and the AHS APeLS (Adaptive Personalized eLearning Service). By combining the strengths of both systems to act as one, it is shown that a more rounded and advanced learning experience can be offered to the online learner, as he/she is offered both the personalized learning services of the AHS and the navigational and support facilities of the CMS. The reasoning and possible benefits of using such an integration are discussed in detail, as is the specific architecture and design I implemented to achieve this. A brief overview of the current state of the art in CMS and AHS technology is also given, which is then followed by an outline of the design and implementation I chose to create the successful integration of both systems. Finally, some general observations are stated about the overall worth of this project, and some insights are also given into what impact this project may have on future developments.
Chapter 1: Introduction.

In this chapter, I will give a brief overview of what the reasoning was for choosing this project. I will present the motivation for choosing to develop an integration of Moodle and an AHS, and some of the possible benefits this may bring to the field of eLearning. I will then present some of the objectives I hope to achieve by developing this system, and show why an Adaptive Moodle can provide so much more than a traditional CMS could ever hope to do.

Section 1.1: Motivation.

In the current eLearning landscape, the most prevalent architecture used to deliver content to a learner is via a course management system (CMS)\(^1\). These systems maintain all of their content centrally in a local database and deliver it to their learners over the internet, usually via a web browser. Although this approach is still highly successful, it still means that course developers and publishers of content must essentially design their content to work in a particular Learning Environment (LE), thus restricting the reuse and marketability of content from one system to another. Thus, if a developer creates some content in a system like WebCT, and then tries to use the same content in another system like Blackboard or Moodle, he will quite often have portability problems and more often than not, be forced to rewrite the content each time, depending on the CMS being used. Although certain advances have been made in the packaging of content for reuse\(^2\) in different environments, portability it is still a key-issue amongst different CMS’s. As the creation of content is the most time consuming and irksome\(^3\) part of offering online courses to users, an alternative approach to content reuse should be investigated. On this note, we introduce the Adaptive Hypermedia Services (AHS).

The AHS is essentially a stand-alone service which can provide content (that may have complex play rules associated with it) to learners and the ability to offer personalized courses to individual users. An AHS generally resides on a separate server on the internet, which can be accessed by a user in a manner similar to that used to access a traditional CMS. As mentioned, the AHS is capable of offering a broad range of content to the user. However, unlike the CMS, it is quite limited in terms of user management, or administrative functionality. It can essentially deliver content to a user’s browser, but it

---

1 Also referred to sometimes as a Learning Management System (LMS), Virtual Learning Environment (VLE) etc.
2 For instance, the IMS Content Packaging and SCORM. It should be noted however, that these standards are generally geared towards the packaging of static content, and cannot as yet fully support the complex play-rules associated with some pieces of content.
3 Quite often also, the most expensive part.
cannot offer many of the extra options which are taken for granted in traditional CMS environments.  

“LE’s provide administration and support facilities, while AHS’s provide better quality education, personalization and pedagogical control to learners.”

For these reasons, if one were to integrate both systems in a comprehensive way, allowing for communication and information sharing between both systems, we could essentially combine the strengths of both systems to act as one individual and far more powerful system. This is the main motivation behind this paper, namely, to create a seamless integration between the Open-Source CMS known as Moodle, and the AHS developed at Trinity College Dublin called APeLS.

Section 1.2: Objectives.

As stated above, the main motivation for this paper is to design and implement an integration of the CMS Moodle and the AHS APeLS. In so doing, the strengths of both systems can be exploited and can be made to work together as a single system. The CMS can provide the usual administrative functions to a course teacher and the usual options to a user, whereas the AHS can provide individual personalized courses to users and a more interactive form of learning, over which the user and teacher have more control.

In Chapter 2, I will give a general overview of the current technologies used in both the CMS and AHS fields and also give an overview of some current research projects which are still in the development state. Section 2.1 will describe exactly what a CMS is and their practical application to the field of eLearning. It also gives an overview of some of the main competitive systems in use in both educational and commercial environments and the respective strengths and weaknesses of each. Section 2.2 will then describe Adaptive Hypermedia systems and their practical role in an educational environment. It also discussed two of the most recent systems in use and comments on some of the good and bad points of each.

Chapter 3 investigates some of the main information sharing issues necessary to achieve a successful integration of both systems. It outlines what possible points of information need to be shared between both systems to achieve a seamless integration, and then described some different approaches to achieving such an integration. I then go on to describe the specific approach I chose to achieve this integration and discuss why I felt this was the most suitable approach.

Chapter 4 gives an overview of using this new integrated system to set up new courses within Moodle and demonstrates that the implementation that I chose works exceptionally well. It presents a step-by-step account of using the system, and finally

---

4 For instance, user forums, administrative tools such as user logging and tracking etc.
6 Such as access to forums, personalization of the user profile, interactive chat options, access to lecture notes, options for feedback etc.
gives an overall evaluation of some of the key issues I encountered during the implementation of this integration.

Finally, Chapter 5 will give a brief synopsis of the main issues I tried to solve during the course of this project, and will also discuss some possible future implementations or modifications to the work carried out. I will show that by using the approach I have designed, CMS’s and AHS’s can be used in multiple practical applications and that the value of this integration may have far reaching consequences. Finally, I will conclude with a few closing remarks.
Chapter 2: State of the Art.

In this chapter, I will explain in detail the various technologies used and encountered during the course of this project. Firstly, I will explain what a CMS or LMS is and give some examples, which are currently in use. I shall then explain what an AHS is and show some different examples and implementations of them.

Section 2.1: CMS (Course Management System).

A CMS or Course Management System\(^7\), put simply is a software system that is designed and marketed for faculty and students to use in online teaching and learning. It is a tool that can allow instructors, universities, or corporations to develop and support online education. While CMS’s are still predominantly used in academic institutions, their use is now also spreading to the commercial world, as they are being used to provide online learning and training facilities to employees, stakeholders and customers. Their potential as teaching tools has become quite evident, and they offer a good alternative to the more traditional classroom based environment. It is also worth noting that a CMS can offer a far more cost-effective\(^8\) and flexible learning environment to its users, as it may be accessed on a 24-7 basis by all who use it, and may also be customized by a course teacher or administrator to suit the needs of a specific group, or class of learners. Thus, a user may use the CMS to learn at her own pace, but the administrator may still maintain some control over this, by setting regular assignments and bench-marks for the user to complete. In summation, a good CMS should support a collaborative learning community, offering multiple modes of learning - from self-paced coursework (Web-based seminars and classes, downloadable CD-ROM and video content) to scheduled classes (live instruction in classroom settings or online) to group learning (online forums and chats).

Evaluating a Course Management System\(^9\)

When attempting to evaluate a course management system, certain factors should be taken into account. For instance: high availability, usability, scalability, interoperability, stability and security. In this section, we will take a brief look at each of these issues and

\(^7\) Note: The terms CMS (Course Management System) and LMS (Learning Management System) may be used interchangeably.

\(^8\) For the employer that is. Implementing a CMS may avoid having to offer on-site or off-site personalized training to employees.

\(^9\) Taken from http://www.clomedia.com/content/templates/clo_feature.asp?articleid=91&zoneid=29
see why they are critical to the function of any enterprise or academic management system.

- High availability: The CMS must be robust enough to serve the diverse needs of hundreds of learners, administrators, content builders and instructors simultaneously.
- Scalability: The infrastructure should be able to expand—or “scale”—to meet future growth, both in terms of the volume of instruction and the size of the student body.
- Usability: To support a host of automated and personalized services, such as self-paced and role-specific learning, the access, delivery and presentation of material must be easy-to-use and highly intuitive—like surfing on the Web or shopping at an online store.
- Interoperability: To support content from different sources and multiple vendors’ hardware/software solutions, the CMS should be based on open industry standards for Web deployments (XML, SOAP) and support the major learning standards (AICC, SCORM, IMS and IEEE).
- Stability: The LMS infrastructure can reliably and effectively manage a large enterprise implementation running 24-7.
- Security: As with any outward-facing collaborative solution, the CMS can selectively limit and control access to online content, resources and back-end functions, both internally and externally, for its diverse user community.

Some Example CMS’s / LMS’s
In this section, I will give a brief overview of some of the currently available CMS’s. Starting with the two main proprietary rival systems, Blackboard and WebCT, I will then discuss some alternative Open-Source systems, Claroline and Moodle, which offer a reasonable and affordable alternative to the above.

**Blackboard**

**What is Blackboard?**

- Learning Management System (LMS) software partially owned by Microsoft
- Popular software used around the world in academic & corporate circles
- Licensed annually (HSU\(^{10}\) pays approx. $8,600 each year)
- Used at HSU since 2001
- Over $1 million spent each year in the CSU on Blackboard licenses

Blackboard features include threaded discussion forums, digital drop box for submission of student assignments, online grade book, document sharing and e-mail messaging. Although Blackboard has proven itself as a stable and effective CMS, it is still rather limited in many ways. Being a proprietary system, the option to alter it by the receiving institution is eliminated, as this is in breach of copyright laws. Although

\(^{10}\) Humboldt State University (Reference taken from [http://www.humboldt.edu/~jdv1/moodle/all.htm](http://www.humboldt.edu/~jdv1/moodle/all.htm))
Blackboard has added some options for additional expansion\(^{11}\), it is still in stark contrast to the Open-Source CMS’s, which can be altered or added to at will by the receiving institutions developers, or any member of the Open-Source community. On the other hand, Blackboard is still widely used, and is often cited for its ease of use\(^{12}\) and efficiency in offering various courses in an e-learning environment.

**WebCT**

Blackboard’s leading rival in the proprietary CMS / LMS market, WebCT offers many of the same options as BB (As listed above). WebCT however has also been quoted as being more powerful and flexible in its presentation and structuring of courses. WebCT is also

"better for ‘individual, self-paced’ learning because of its ‘selective release’ feature. [...] and also “had more support for individual instructors with ‘specialized needs’ “\(^{13}\).

Several other features in WebCT, including: an enhanced course creation facility for uploading content (this allows an admin to upload a single piece of content, or an entire course in one go); a more controlled Calendar facility (to allow administrators to only show content to users at a given time in the year); a Selective Release feature (to allow admins to offer more or less content to specific users, depending on their level of knowledge); further course customization options (including a variety of icons and clip-art for faculty to use in courses): would seem to suggest that WebCT offers a lot more options to a prospective institution than its rival Blackboard\(^{14}\). For this reason, WebCT continues to spread in popularity in the e-learning field, and is used in academic and commercial institutions equally all over the world.

It should be noted however, that although WebCT does offer many options of customization and personalization, it is still a proprietary system, and thus cannot be fundamentally altered by the receiving institutions developers, without breaking copyright and international protection laws. Equally, WebCT can prove to be quite an expense, as regards licensing, to a prospective institution, with annual licensing and upgrade fees running to thousands of Euro each year. For these two reasons, several institutions are now beginning to turn towards the Open-Source community for affordable and customizable alternatives. The first of these alternatives (Claroline) is discussed below.

**Claroline** \((http://www.claroline.net/))

---

\(^{11}\) This can be done by developing new Building Blocks \(http://www.blackboard.com/dev/DevOverview.htm\) for Blackboard. These are part of BB’s new open architecture initiative.


\(^{13}\) ibid. page 13.

\(^{14}\) For a fuller list of these features and a detailed comparison of Blackboard vs. WebCT, see \(http://www.wcu.edu/it/cio/planning/cmsfinalreport.pdf\) (page 12 – 16).
Originally developed by the Université Catholique de Louvain-La-Neuve as an Open-Source alternative to Blackboard / WebCT, Claroline is a simple, intuitive, and user-friendly VLE\textsuperscript{15}, developed in PHP, with a reasonably large user community (Currently used at UCL to teach multiple courses – \url{www.icampus.ucl.ac.be}). Claroline however has been found to lack in certain functionality – specifically in terms of tracking and quiz functionality, a lack of good user documentation, and the fact that most of its developer documentation is written in French. It is worth noting however, that Claroline has recently been split into two different groups. It appears the UCL team has patented the name, but is no longer spending as much on developing it. The second group (Which is now called Dokeos\textsuperscript{16}) is still developing the product, and may offer a more advanced system to prospective institutions. Although much of the lime-light in the Open-Source CMS community has been stolen by Moodle of late, Claroline may yet be a reasonable alternative in the near future.

**Moodle** (**http://www.moodle.org**)

Moodle, or the *Modular Object-Oriented Dynamic Learning Environment*, is an Open-Source CMS, which is strongly grounded in social constructivist philosophy\textsuperscript{17}. Development was first begun on the project by Martin Dougiamas, who wished to get away from WebCT and create a new alternative and easier to use system:

“I encountered many frustrations with the WebCT beast and developed an itch that needed scratching - there had to be a better way (no, not Blackboard :-)”\textsuperscript{18}

Since then, he launched the first official release of Moodle (version 1.0) to the public in August 2002, and ever since, the project has evolved greatly (Currently version 1.5), with contributors from all around the world adding new features, better scalability and improved performance.

One great strength of Moodle is that it is platform independent, meaning it can be run on Unix, Linux, Windows, Mac OS X, Netware or any other system which supports PHP. Its data is stored in a single database, which is generally either a MySQL or PostgreSQL DBMS, though it can also run with Oracle, Access, Interbase, ODBC and others systems, through use of its ADOdb technology\textsuperscript{19}.

Moodle is very simple to install on any platform that supports PHP. It is simple, lightweight, and efficient and is compatible with almost any browser interface. Courses can be easily created within Moodle by a teacher or site administrator, and these can be categorized and searched at will. Emphasis is also placed on strong security throughout. Forms are all checked, data validated, cookies encrypted etc. Most text entry areas (resources, forum postings, journal entries etc) can be edited using an embedded

\textsuperscript{15} Virtual Learning Environment

\textsuperscript{16} \url{http://www.dokeos.com}

\textsuperscript{17} For a full discussion on this, see \url{http://moodle.org/doc/?file=philosophy.html}

\textsuperscript{18} Martin Dougiamas, \url{http://moodle.org/doc/}

\textsuperscript{19} \url{http://adodb.sourceforge.net/}
WYSIWYG\textsuperscript{20} HTML editor, thus making it very simple for inexperienced users to create new courses, or add their own content. There are also multiple site management options provided for the administrator (including a list of plug-in themes, to alter the site appearance; plug-in activity modules, which can be added to existing Moodle installations; plug-in language modules, allowing for over 43 different types of language localizations), multiple user management options (including a range of different authentication mechanisms, from manual user account creation, to automatic email-based authentication), multiple course management options (including a choice of course formats, such as by week, by topic etc.; an extensive range of course activities, which can be added to at any stage; user tracking and logging facilities etc.), and multiple other options and features\textsuperscript{21}. For these reasons and others, Moodle is constantly increasing in popularity amongst academic and commercial institutions alike and is expected to do so even more in the future. As it is an Open-Source project, it has an extensive team of developers from different backgrounds and experiences, and hence will evolve to suit everyone’s needs. Its simple but intuitive design and its great ability to be customized for different purposes (for example, by adding or removing various different modules and activities) makes Moodle a serious contender on the CMS / LMS market, and from my own personal experience\textsuperscript{22} of working with Moodle, it comes highly recommended.

\section*{Section 2.2: AHS (Adaptive Hypermedia Systems).}

In this section, I will give a brief overview of Adaptive Hypermedia and some of the currently available AHS’s (Adaptive Hypermedia Systems) being developed.

\textbf{AHS Overview.}

One of the key features of Adaptive Hypermedia Systems is that the system should be able to adapt accordingly for each individual user, thus offering a more personalized and customizable course to a user than a static web-page or other piece of content which the user has little control over. Adaptive hypermedia (AH) systems build a model of the goals, preferences and knowledge of the individual user and use this throughout the interaction for adaptation to the needs of that user.

\begin{quote}
\textit{“The system collates information about each user into a user profile and this model is used to make assumptions about how best to change the system to benefit an individual user.”}\textsuperscript{23}
\end{quote}

\textsuperscript{20} \textit{WYSIWYG} = “What You See Is What You Get”
\textsuperscript{21} For a full list of these features, see the documentation at: \url{http://moodle.org/doc/?file=features.html}
\textsuperscript{22} As you will see during the course of this paper, I wrote a new activity module for Moodle and used this to create an integration of Moodle with an AHS developed at TCD. This would have been virtually impossible to do in WebCT or Blackboard, unless I was given explicit access to their source-code. In Moodle however, it was quite simple to do, with other sample Modules to work from and good documentation available.
By generating a learner model for each individual learner, the system is able to determine what specific piece of content the AHS should choose based on a learner and tutors requirements. Thus, the learner model can contain a set of objectives for a given learner and based on this, the AHS can produce an appropriate body of content. Adaptation can also assist the user in a navigational sense, which is particularly relevant in a large hyperspace.\(^{24}\) Knowing user goals and knowledge, AH systems can support users in their navigation by limiting browsing space, suggesting most relevant links to follow, or providing adaptive comments to visible links. Thus, the notion of Adaptive Hypermedia Systems can be summed up as follows:

“by adaptive hypermedia systems we mean all hypertext and hypermedia systems which reflect some features of the user in the user model and apply this model to adapt various visible aspects of the system to the user.”\(^{25}\)

As Brusilovsky mentions, for a system to be classified as an AHS, it must essentially meet three criteria: it should be a hypertext or hypermedia based system, it should have a valid user model, and it should be able to adapt the hypermedia using this model (i.e. the system may appear differently to each user depending on each consecutive user model). Taking these factors into account, adaptive hypermedia techniques can be useful to solve a number of problems associated with the use of educational hypermedia. Firstly, the knowledge of different users can vary greatly and the knowledge of a particular user can grow quite fast. The same page can be unclear for a novice and at the same time trivial or boring for an advanced learner. Bearing this in mind, an AHS can use the information contained in the user model to offer only appropriate content to an individual user and discard other, less relevant information.

Different forms of adaptivity may be provided by the AHS, including Adaptive Navigation, Adaptive Presentation, Structural Adaptation and Historical Adaptation.\(^{26}\) Systems like AHA!\(^{27}\) tend to opt towards navigational adaptation, dynamically changing the navigation structure for each user. Similarly, ELM-ART\(^{28}\) and Interbook\(^{29}\) also use adaptive navigation to create colour-coded annotations which indicate how suitable or relevant a link or topic is for a given user. Other systems like AVANTI\(^{30}\) lean instead towards adaptive presentation. The AVANTI interface can dynamically tailor itself to the abilities, skills, requirements and preferences of its users, to the different contexts of use, as well as the changing characteristics of users, as they interact with the system. This system is built to adapt to both disabled as well as able-bodied users.

\(^{24}\) This can help avoid the “Lost in hyperspace” problem.
\(^{25}\) Brusilovsky, P. “Methods and Techniques of adaptive hypermedia”
\(^{26}\) See Wade, V, Conlan, Owen, “Can Technology Learn From The Learner?” for a fuller discussion of these.
\(^{27}\) De Bra, P. “AHA! Adaptive Hypermedia for All”.
\(^{30}\) AVANTI Project. http://www.ifac.cnr.it/avanti/contents/home.htm
**APeLS (The Adaptive Personalized eLearning Service).**

The principle AHS which I was interested in for this paper was the Adaptive Personalized eLearning Service (APeLS), which was developed as a service to deliver personalized educational courses based on a multi-model, metadata driven approach\(^{31}\). This system has been used in TCD to teach under-graduate courses in SQL and has so far proven highly successful.\(^{32}\)

The architecture of APeLS consists of essentially three models – the learner, narrative and content models. These models are taken at runtime and passed to an Adaptive Engine (AE). From this, a personalized course is then delivered for an individual user. The APeLS also has a number of metadata and information repositories, namely the:

- **Learner Metadata Repository** – This stores all of the metadata representing the individual learners in the system.
- **Content Metadata Repository** – This stores all of the metadata conforming to the Content Model corresponding to each piece of learning content.
- **Content Repository** – This stores all of the pagelets referred to by the Content Model Repository.
- **Narrative Metadata Repository** – Stores the metadata records that describe the learning objectives and pedagogical approach for each narrative in the narrative repository.
- **Narrative Repository** – Stores all of the narratives used to construct personalized courses.

Finally, there are two candidate group repositories, namely the:

- **Candidate Content groups** – Which reference the groups in the Content Metadata repository that fulfill the same learning goal.
- **Candidate Narrative groups** – Which contain groups of narratives that encapsulate the same knowledge, but employ different pedagogical approaches to structuring the content.

---


At the heart of APeLS lies the Adaptive Engine (AE). This uses the Java Expert System Shell (JESS) with customized functions as the basis of its rules engine. Taking a narrative and a learner model as its inputs, the AE produces a model for a personalized course. This XML-based course model encapsulates the structure of the learner’s course and contains the candidate content groups that fulfill the learner’s learning requirements in accordance with the narrative. The AE can also use a candidate selector for choosing the appropriate narrative by reconciling information in the learner model with the candidate narrative groups. This candidate selector is also used to choose the appropriate piece of content to deliver from a candidate content group when the personalized course content is being generated from the personalized course model. Finally, the AE also has a learner modeler component which can take input from the course or a designated pre-test and translate this into changes in the learner’s information. This can be used to populate the learner’s model when the learner initially enters the AHS\textsuperscript{34}, or during runtime to modify or update the learner’s model.

\textsuperscript{33} Taken from Conlan, O., Wade, V., Bruen, C., Gargan, G. “Multi-Model, Metadata Driven Approach to Adaptive Hypermedia Services for Personalized eLearning”, 2002.
\textsuperscript{34} Note: In my implementation, this step is done within Moodle. The user completes a pre-test questionnaire within Moodle, which then creates an appropriate learner model for this user. This model is then passed back into the AHS and the personalized course is then created.
It should also be noted that, in contrast to some of the earlier AHS’s which were essentially stand-alone systems, APeLS can also be offered as a service and can be easily integrated with a preferred Learning Environment or CMS. The benefits of this are self-evident. The learner can be offered an adaptive course from within the CMS, as though it is a part of that CMS, thus offering the learner the best of both static and adaptive content for his learning needs. The benefits to the course administrator are also self-evident, in that he can easily offer a new course to his students by simply “plugging-in” an AHS course, and also complementing this with more traditional lecture notes, overhead slides etc. Equally, the AHS can be integrated with the CMS as part of an overall larger course; the AHS component being only a small or large part of this. Finally, by integrating the CMS with the APeLS system, a course administrator will also have access to valuable tracking and assessment information for both the adaptive course and native CMS content. This can then be used later to grade students based on their time spent covering the content and overall understanding of the subjects covered35, or equally to provide feedback to the course administrator on what topics are most relevant to students and which topics should be improved upon.

The first successful attempt to integrate APeLS with a CMS was done as part of the EASEL IST project36. For this, APeLS was successfully integrated with Fretwell-Downing Educations Learning Environment using the Content Interworking API initially defined by the AICC37. In this project, APeLS and the LE were able to exchange learner identification information, as well as performance and assessment information. Essentially, APeLS could request the learner’s identification from the LE when the service was initially launched and based on this APeLS could then check if the learner already had an appropriate learner model or not stored in its learner model repository. APeLS could also return any relevant assessment information to the LE once the learner had completed their session, and this was then stored in the LE’s database. This interaction was achieved using an API and data model based on a JavaScript implementation of the ADL SCORM38. For a fuller description of the exact implementation used, see Conlan, O., Wade, V. & Gargan, M. “An Architecture for integrating Adaptive Hypermedia Services with Open Learning Environments”. Thus, APeLS can be offered as a stable and stand-alone AHS, offering a fully adaptive course to an individual user, or as was shown in the EASEL project, as a pluggable service, which can be offered to a CMS and run as a local service.

35 This can be done by using the inbuilt questionnaires of the CMS to test the students once they have completed a course, or equally, by using the course assessment option within the AHS.
36 Educators Access to Services in the Electronic Landscape (EASEL), EC IST project 10051, http://www.fdgroup.com/
The following diagram best illustrates this integrated approach:

![Diagram](image)

**KnowledgeTree**

A new and reasonable alternative to APeLS is the KnowledgeTree\(^{39}\) system. This is essentially a distributed re-usable intelligent learning system, which aims to bridge the gap between the modern approach to Web-based education based on CMS and educational material repositories and powerful but underused ITS\(^{40}\) and AH technologies. As Brusilovsky puts it:

*“We believe that a way to the future starts on the crossroads of courseware re-use and adaptive educational systems.”*\(^{41}\)

KnowledgeTree aims to replace the current monolithic course management systems with a community of communicating servers. The architecture to achieve this anticipates the presence of at least three kinds of servers: activity servers, learning portals and student model servers.


\(^{40}\) Intelligent Tutoring Systems

The learning portal has a similar role to a current CMS, in that it provides the user with a centralized single-login point from which he can access and work with all the available learning tools and content fragments which are distributed over the web. This portal can also allow a teacher or course administrator to structure the access to various distributed fragments according to the needs of a specific course, thus offering him control over the content which a student should cover. Like the CMS, a learning portal should provide some course-authoring interface to a teacher, and likewise, should provide a runtime interface to the user, from which he can navigate the various content. The main difference between this approach however and the more traditional CMS approach is that there is a separation of the unique course structure from the reusable course content. This is because the content and learning activities reside in separate activity servers, rather than being embedded in the learning portal. Thus, a learning portal server is essentially an administrative point for a teacher and a starting point for a user, and has the ability to query activity servers for relevant activities and launch activities selected by students. An activity server is essentially a repository of reusable content and services, which can be accessed by a learning portal. It is not however merely a static database for storing simple learning objects which can be copied and inserted into courses, but rather, it can also deliver the activities to the appropriate portals. As Brusilovsky puts it:

---

42 Taken from: Brusilovsky, P. “A Component-Based Distributed Architecture for Adaptive Web-Based Education”, 2003.
“The duty of an activity server is to answer portal’s requests for specific activities and to provide a complete support for a student working with each of the activities residing on the server.”

Finally, a student model server is needed to collect data about student performance from each portal and activity server, and can equally provide information about the student to adaptive portals and activity servers that are able to personalize their communication with the student. This server can be seen as a centralized user modeling service, which enables each external server to have access to all information about student progress. One such student modeling server has already been created by Brusilovsky et al. which is called CUMULATE.

It should be noted however, that although KnowledgeTree has so far been successfully implemented by Brusilovsky et al. using one learning portal, one student modeling server and four activity servers, it is still a relatively new area of research, and much work still needs to be done on it before it can be offered as a commercially viable system, or a true alternative to the more traditional CMS approach. Nevertheless, KnowledgeTree, like APeLS has a great deal of potential, and is certainly a new and exciting field of research which may produce some impressive results.

Chapter 3: Requirements & Design.

In this chapter, I will outline some of the key issues I encountered during the design of this project. I will first off, attempt to identify the key information sharing points which were necessary to create a successful integration of Moodle and the AHS. Following this, I will outline some different possible approaches I could have taken for my design and finally, I will show why my final chosen design was the better and more practical choice.

Section 3.1: Key Information Sharing Points.

Single sign-on

Having investigated both Moodle and the AHS APeLS thoroughly, the first step towards a successful attempt was the issue of single user sign-on. What this means essentially is that a user should only have to log into Moodle once, and from within Moodle, to launch the AHS course as though it were an internal part of Moodle. The user should not even be made aware that two separate systems are being used. Thus, a student should log in to Moodle as normal, using an individual username and password. Once this is done, he will be given a list of the courses he has been enrolled in and various other options (For instance, to view any recent site news, or other important announcements etc.). Once he selects a course in which the AHS is being used, it should be simply a matter of choosing the AHS (Adaptive Course) hyper-link within this course and he will be automatically logged into the AHS course. By doing this, a seamless integration is given to the user between the two systems, and as mentioned above, he shouldn’t even be aware that two separate systems are being used.

Updating of learner model

The second key-point which I identified for integration was the option to update the learner model from within Moodle. Using APeLS\(^{45}\), a learner model is normally generated for each user by first offering them a pre-test\(^{46}\) within the AHS itself. This is done by re-directing the user to a JSP page with the appropriate questionnaire if he has not already filled it out on a previous visit. If the latter, then the user is simply directed to his course and does not need to re-take the pre-test again (Unless of course he wishes to rebuild the course at any stage). As a step towards extracting away from the specific limitations of creating the learner model within the AHS, I looked at different ways of creating this model within Moodle, and then passing this information back to the AHS.

---

\(^{45}\) As explained in Chapter 2

\(^{46}\) A series of question to assess their overall knowledge of the subject and different aspects of a course
Using Moodle’s local database, we can gain access to many of the learner details which can be used in the learner model. For instance, we can extract the username, the first and last names, the user’s email address, his home address, phone number etc. from Moodle and then use this information in the learner model. Information about learning styles\(^47\), assessment or tracking could also be potentially extracted from Moodle, depending on the specific application chosen. Thus, by adding the pre-test and learner model to Moodle, instead of leaving it to the AHS to determine, we can greatly increase the information contained therein, and offer a more flexible approach to creating the adaptive course for the user.

**Adding Tracking Information.**

A third key-point which I felt essential to a successful integration of Moodle and the AHS, was the option to add additional tracking information for a user of the adaptive course. Like many commercial CMS’s, Moodle already comes with a rather robust tracking option for users who take courses within Moodle. This can be used to view details on how often a user uses the system, and what parts of the system (For instance, which specific courses and options within each course) he uses on a given visit. This information is stored locally within the Moodle database, and can be freely accessed by a course teacher or site administrator at any time. Visual graphs, which show the amount of activity at a given time or day are also available to an administrator, and can prove to be a useful resource to monitor a student’s progress and use of the system.

On the other hand, when I investigated the APeLS system, I saw that no worthwhile tracking option was available within the system. Thus, I had to look into various ways of adding this option so the tracking information could later be extracted. As I will explain in more detail later, I essentially had to alter some of the existing code in the JSP files, so I could add this tracking option. Having done this, an administrator within Moodle can then choose to enable tracking for the AHS course for a specific user. When tracking is turned on, the AHS will log each interaction the user makes with the system, and pass this information back to Moodle, where it is stored locally in Moodle’s database. By doing this, an admin can then view a user’s progress both in the overall Moodle courses, and also view detailed tracking information on a user of the AHS course. This information can then be used later to verify that a user has in fact covered the core material, or could equally be used for data mining operations for another application.

**Adding Assessment Information.**

The final key-point of integration between the two systems which I identified was that of user assessment information. By this, a course teacher or site administrator should have the option to assess a user’s overall knowledge of the course material covered and test them on their understanding of the subject. The results of this assessment may be used as

\(^{47}\) Note: In the version of APeLS which I worked with for this project, the learning styles option was quite limited. This has since been greatly improved, and can accommodate 4 different learning styles, namely: pragmatist, activist, reflector and theorist. This information is obtained by offering the user a separate questionnaire, based on a sub-set of the Honey & Mumford questions, to obtain the correct learning style.
part of an overall grade for a course (summative test), or can simply be used as a guide to show the users progress in the course (formative test). Once again, like any good CMS, Moodle comes with various options to provide questionnaires, choice forms, quizzes and workshops for this kind of assessment option. An administrator can create an assessment questionnaire within Moodle and then offer this to his students as a test. A grade and feedback can also be provided using this option. Alternatively\(^{48}\), an assessment option could equally be implemented within the AHS itself and offered to a user when he has finished going through the adaptive course. This assessment option should essentially be a multiple-choice questionnaire, based on the content covered and have some options to provide user feedback\(^{49}\). The user will be given the option to take this assessment test as part of his overall grade, or if he has already taken it, then he can simply use it as a learning tool for future use. The results of this assessment should then be fed back to Moodle and stored locally there for future analysis.

Section 3.2: Different approaches to Integration.

During the course of the design stage for this project, I looked at some different possibilities for integrating the AHS with Moodle. The first of these is based on Conlan, O., Wade, V. & Gargan, M. (2002)\(^{50}\), where they managed to successfully integrate the APeLS system with the Fretwell-Downing VLE (Virtual Learning Environment).

For this, they focused on three main points of integration:

- Importing the location of the AHS into the LE
- Launching the AHS from the LE
- Communicating between the AHS and LE

For the first of these, they used the IMS Content Packaging archive to allow for the importing of the location of the AHS as part of the IMS Content Package. This approach is normally used for static content, whereby the IMS Content package is simply uploaded into the LE (As a series of separate files, or as a single zip archive) and, using its XML IsManifest form to check that the content is valid, the package is simply installed into the LE’s content repository. In the case of static content, which does not include any complex play rules are other interactions, this approach works fine. Indeed, within Moodle, a SCORM package can simply be added to a course as a new activity. A zip archive is uploaded by a course teacher and if the archive is well-formed, it is simply added to the course as a separate piece of content (Or indeed as a separate course). It is worth noting however that many problems still occur with the use of SCORM compliant packages, and this approach is still far from robust or reliable. For the Conlan, Wade implementation, they describe how the IMS Content Package location can simply be imported into the LE,
with the manifest describing the adaptive service and also containing a URL which the LE would use to launch the service. It remains to be seen if this is in fact possible to implement in Moodle\textsuperscript{51}, though it would certainly be an interesting experiment.

For the second part of the Conlan, Wade implementation (Launching the AHS from the LE), they describe how the AHS firstly initializes a connection to the LE, and then checks whether the learner has already been engaged in this course component on a previous occasion. If this is the case, then the AHS simply rebuilds the course for them, based on a previous snapshot taken prior to the content being paused. If the user hasn’t used the AHS component previously however, then the AHS must determine the learning objectives for that user, and also, whether the learner has been engaged in any course component offered by the AHS before. This can be done by providing the learner with a pre-test to determine their learning styles and level of knowledge of the subject.

Finally, for the third part of this implementation (Communication between the AHS and the LE), the communication was achieved by using the SCORM Runtime Communication API (SCORM v1.1). By modifying the HTML frame layout of the AHS, this enabled calls to be made to API functions residing on the LE from the AHS content. This is described in fuller detail below:\textsuperscript{52}

The remote AHS calls the Content Interworking API to access the data model on the LE using the following process :–

a) The learning content (right browser frame) and JavaScript API (left browser frame, hidden) are delivered to the learner’s browser.

b) An API function, in the left hand API frame, (e.g. LMSGetValue(“cmi.core.leason_status”)) is called from the content frame.

c) The API frame communicates the request to the Learning Environment.

d) The Learning Environment returns the value (in this case of cmi.core.leason_status) to the API Frame.

e) The function returns the value to content frame.

f) The value may be passed back to the Adaptive Hypermedia Service.

\textsuperscript{51} This author in particular, having tried many times to import both local and remote packages, remains highly sceptical!

\textsuperscript{52} Taken directly from Conlan, O., Wade, V., Gargan, M. “An Architecture for integrating Adaptive Hypermedia Services with Open Learning Environments”, 2002. See this paper for a full discussion of the design and implementation.
Although the Conlan, Wade implementation seems to have worked rather successfully and it proved that it is indeed possible to successfully integrate the APeLS system with a CMS, this author remains highly skeptical as to whether an identical approach would work equally well using the Moodle CMS instead of the Fretwell-Downing system. Indeed, much of the reasoning behind this stance is based on the fact that Moodle as yet is still rather under-developed as regards its support of IMS Content Packaging and SCORM. Perhaps as Moodle matures, this will be greatly improved and the above implementation will become possible, but for the time being, it would appear that a different approach to integration is necessary.

The second possible means of achieving integration between the two systems (And the one which I chose to work with) is quite similar in many respects to the above, but also includes many other features and alternative approaches. As mentioned in Section 3.1, I tried to identify the main key information-sharing points between the two systems, and based on these, to develop a system which would allow each point to function correctly.
Adding Single Sign-On

The first step in so doing was to create the single sign-on from Moodle to the AHS course. As mentioned above, Moodle has a great deal of learner information already stored in its local repository. Based on this, we can extract a given learner’s username and using this, check to see if that name already exists in the AHS course. This is done by querying the AHS’s XML database\textsuperscript{53} and checking if the username is contained in the “userlist.xml”\textsuperscript{54} file. If the username is already contained in the AHS database (Indicating that he has already used the AHS course), then the user should simply be redirected to his appropriate course. If his username is not already there however, then it must be added. This can be done by using XUpdate from the PHP script\textsuperscript{55} to add the appropriate username and password to the “userlist.xml” file. In this latter case however, as the user has not used the AHS course on a previous occasion, he needs to be given a pre-test, to assess his knowledge of the material covered. The results of this pre-test are then used to populate a learner model for that user and eventually to create a personalized adaptive course. In the original AHS, this pre-test was a separate JSP page (test.jsp) and was an inbuilt part of the AHS. To extract away from this, and hence, create a more integrated model (i.e. adding this functionality to Moodle instead), I decided to offer this pre-test within Moodle instead. I will describe how this is done in the next section.

Creating and Updating the Learner Model

The learner model, which essentially stores details about a specific user (i.e. learner name, forename, surname etc.), also stores details of the specific course which should be generated for that user. The learner model, as mentioned above, is created by offering the user a pre-test to assess his knowledge of the course and subject area. The pre-test for the adaptive SQL course for instance, is essentially a series of 5 questions, namely:

- Have you learned about the Relational Model and Database concepts?
- Did you ever design your own Database and write SQL commands to create it?
- Have you ever populated a Database using different Data Types?
- Have you used SQL commands to retrieve data from a Database using complex joins?
- Have you ever embedded SQL in a C application?

Based on the user’s responses to these questions, an appropriate course is generated for this user. For instance, if the user responds negatively to each question, then he will be given a large amount of content to cover for his course. On the other hand, if he is confident about a certain section, then he is not given this as part of his course. As said above, all of this information is stored explicitly in the learner model. The following for instance, is the learner model generated for a user who answered negatively to question 1 and positively to questions 2 – 5:

\textsuperscript{53} This query is done by using an XQuery from the PHP script. XQueries can be executed in PHP through use of the SOAP interface (http://www.w3.org/TR/soap/).
\textsuperscript{54} This file contains details of usernames and passwords for users of the AHS.
\textsuperscript{55} This is done by using the XmlRPC interface (http://cvs.sourceforge.net/viewcvs.py/exist/exist_phpapi/)
As you can see, the various candidate pieces of content are listed in this model, as well as specific details of the user. This information is then passed to the Adaptive Engine, along with an appropriate narrative and a personalized course is generated for that user.

For my implementation, the pre-test questionnaire was implemented in PHP, and is offered within Moodle. The results of this questionnaire are then checked and an output XML file is generated (Which represents the learner model). This XML file is then uploaded to the AHS local XML database, again using the XmlRPC through the PHP script. Once the learner model has been generated and added to the AHS database, the Adaptive Engine can be called and a personalized course is then generated. Likewise, if at any stage during the learner’s usage of the adaptive course, the learner wishes to rebuild the course structure, he is again forwarded to this pre-test. Upon completion, the learner model is updated with the new information and a new course is generated for the user.
Adding the tracking information option.

The next step in my design was to get the tracking option to function correctly. As I mentioned above, Moodle already has reasonably good tracking options for all users of the Moodle courses. The AHS however doesn’t have any such tracking, and as such, I had to find a way to add it. Having investigated the JSP pages, I saw that the most efficient way of doing this, was to edit the “page.jsp” file, so that each time the user loads a new page (i.e. moves from one page in the adaptive course to another), this event is logged. Initially, I experimented with adding these events to a simple text file, but found that this could generate rather large logs; much of which would be rather irrelevant. Instead, I decided to add an option to Moodle’s tracking facilities which would enable tracking to be switched on for a specific user of an AHS course. This is done by offering an administrator or course teacher a drop-down list of course users, and the option to switch tracking on or off for each one\(^{56}\), as shown below:

![Choose which logs you want to see](image)

When the admin does this, a boolean value (0 or 1) is then set in the “mdl_ahs_tracking” table of Moodle’s MySQL database for that user. A value of 1 is added, which switches tracking on, or a value of 0 is added to disable tracking.

Once this has been set, when a user of the AHS course launches his course, each page that he visits is logged. This is done by modifying the “page.jsp” file of the AHS. First off, a check is done to see if tracking is enabled for that user. This is done by querying the Moodle database\(^{57}\) firstly to see if tracking has been enabled. If so, then each page, as well as the time and date it was visited, the course section and subsection names, and the page number are logged, and these are then stored back into the Moodle database in the table “mdl_ahs_tracking_results”. Having done this, a course administrator or teacher can

\(^{56}\) I also added the option to enable tracking for all users (All Participants).

\(^{57}\) Done through use of the "org.gjt.mm.mysql.Driver" driver from the JSP.
then query these logs at a later date, to view a user’s activity for the AHS course. A sample of this is shown below:

As can be seen from the above diagram, a user’s progress through the AHS course can be tracked successfully. This progress can be seen as rather linear, as he moves from page to page, and section to section. It should also be noted however that once this data has been recorded in Moodle’s database, future applications can be written which may use this data to extrapolate more meaningful results. For instance, one may wish to write an application which verifies whether the user has covered all sections of the course material, or likewise, whether the user has spent more than a certain number of hours per week using the AHS course. This additional features were outside the scope of this paper, but can be easily added at a later date if need be.

Adding the assessment option.

For the final point of integration, I designed a summative\(^58\) questionnaire within the AHS which the user can then take once he has finished his learning. This assessment test is based on the course material covered and is broken into various sections. For instance, the test consists of 20 questions in total, with questions 1-6 testing the user’s general knowledge of the subject; questions 7-12 testing his database retrieval knowledge; questions 13-14 testing his knowledge of database insertion commands and finally,

\(^58\) Meaning it should be taken only once, as it is used to record a course grade for a particular user. The opposite of this is a “formative” assessment, which can be taken multiple times, and an average of these marks are then presented back to the user.
questions 15-20 testing for slightly more advanced knowledge and overall command of the material covered. As mentioned earlier, this test could have been created within Moodle59 and then offered to the user upon completion of the AHS course. However, I felt that to create a more integrated approach, it would be better to embed this test within the AHS course itself, and to give the user the option to take it at any stage along their progression. As mentioned above, the test is used to record an overall grade for the course for an individual user, but once the user has actually submitted the test results, he can then use it as many times as he wishes after that. Thus, the test can be used as a learning tool for the user to test his knowledge as he progresses through the course. If the latter, then the user is forwarded to an alternative results page, where he is instructed that he has already taken the test, and given a display of what his results would have been this time around. He is also given a break-down of the individual parts of the test, and which areas of the course he should concentrate more or less upon.

Finally, the results of this assessment test are again stored back in the Moodle database, where they can be accessed at a later date by a course administrator or a teacher. These results are stored to the “mdl_ahs_scores” table of the Moodle database, which stores each individual answer for each user, along with his overall score60 and the scores he achieved on each section of the test. This offers a seamless integration between the two systems, as the test is offered as part of the AHS, and the scores are then fed back into Moodle for future use and overall grades.

In conclusion therefore, I feel the design I chose for this integration has worked rather successfully. I have successfully implemented each key-point of integration which I identified initially, and each feature works very well. As far as the user is concerned, the AHS course is simply another part of Moodle and can be launched seamlessly from within Moodle at any stage. The user need only log in to Moodle and from there, can access the AHS course in one instant. He is also given the pre-test questionnaire from within Moodle and this is then fed back to the AHS to create the personalized course, thus displaying the active communication between both systems. Finally, both the tracking and assessment options work very well, offering even more administrative options and course feedback to a potential course teacher or administrator. Taking both systems as one, as has been illustrated here, the user is thus offered an exceptional learning experience, where he is presented with the multiple options of Moodle, as well as the customizable and adaptive features of the AHS. Finally, it is also worth noting that this design is not simply limited to the APeLS AHS, but can also be easily modified to accommodate alternative AHS’s. This design therefore has greatly increased the use of Moodle, from a traditional, static CMS, to a new and exciting system which can easily be integrated with other Adaptive Hypermedia systems.

59 By using one of the inbuilt Questionnaire or test activity modules in Moodle.

60 Note: In this implementation, a score of 3 is awarded for a correct answer, and a score of -1 is awarded for an incorrect answer. This scale can easily be changed, or equally, each question can be rated according to its’ respective level of difficulty.
Chapter 4: Implementation & Evaluation.

In this chapter, I will present an overview of creating a new course within Moodle, and adding an AHS component as a new activity module. I will also demonstrate some of the functionality available to a teacher or course administrator of Moodle and also show how each of the features I have added (as described in Chapter 3 above) can be easily used. I shall then present an evaluation of the overall system and make some comments and suggestions on some of its good and bad points.

Section 4.1: Trial of the final integrated system.

For a potential user or institution who wishes to change over to Moodle (from WebCT or Blackboard for instance) as their centralized CMS, or one who wishes to install Moodle as a fresh new system on a trial basis, the installation and distribution of Moodle could not be made simpler. All one needs is some web-server software (such as Apache) which supports PHP, and a valid back-end database system (like MySQL, PostgreSQL, Oracle etc.), and Moodle can be installed almost instantly. A distribution of Moodle can be freely downloaded at http://download.moodle.org/ which comes in the form of a .zip archive. This archive should be extracted to the web-server’s root directory, for instance, “http://yourwebserver.com/moodle”. Once this is done, the administrator need only access the Moodle main URL using a web browser, where he will be presented with a series of questions to configure the system. This automatic installer will also test the server environment and also make some possible suggestions on how to fix any potential problems. An empty database also needs to be created in your database system, along with a special user (e.g., “moodleuser”) who has access to this database. When the automatic installer has finished, it should re-direct the administrator to the “admin” page, where some more configuration options are offered and the Moodle database tables are created and populated with the relevant entries. At this stage also, the administrator can configure the look and feel of the site, as well as the main welcoming page etc. This is shown below:

---

61 For instance, a prospective university or other institute who wish to test Moodle to see if it’s a valid online supplemental to their traditional class-room environment.

62 Or in some cases, access http://yourserver/install.php directly.
Adding a new course in Moodle

The next step is to create a new course in Moodle which users will then be able to access. This is done by selecting the “Courses” hyperlink to the left of the main page. A course can simply be created in the default location, or if the administrator wishes to set up different categories, then he can do so here by selecting the “Add new Category” option and entering the name of this new category. When the admin decides where the course should appear in these categories, he simply selects the “Add a new course” button, and will then be presented with the following screen:

63 For instance, a potential category might be “Maths courses”, “Computer courses” etc.
As can be seen from the above screen shot, the administrator is given multiple options here. He can enter the full name and short name of his course, as well as a unique ID number. He can also provide a brief summary of what the course is about in the WYSIWYG\(^64\) html editor provided, and is also given plenty of other options, such as defining what structure the course should take\(^65\), when the course should begin and how long it should last, an option to restrict the enrollment period for students, an option to make the course accessible to all students, or only those who provide an appropriate key etc. When the admin has finished configuring these options, he needs only select the “Save changes” button at the bottom of the screen and the course will be created for him\(^66\). When this is done, the admin will be forwarded to the next screen, where he can choose who the teacher (Or teachers) for this course should be. This is shown below:

\(^{64}\) WYSIWYG = “What you see is what you get”  
\(^{65}\) A choice between “Weekly format”, “Topics format” and “Social format”  
\(^{66}\) Note: The admin or course teacher can also edit or change any of these options again at a later date if he should so desire.
When the admin chooses the “Add teacher” hyperlink, this user will be assigned as the teacher for this course. The admin can also choose to assign more than one teacher for a course. If this is the case, then he can also assign the order of these teachers. This is shown below:

Finally, when the admin chooses the “Save changes” button, these users will be committed as teachers of this course. At this point, the course is fully set up within Moodle. The site admin also has the option to manually add users to this course, or can
simply allow users to sign up for courses automatically. In the next section, I will show how a course teacher can then configure his individual course.

**Configuring an individual course in Moodle**

When the course teacher logs into Moodle, he will be presented with the site’s main introductory page, and to the left of this, he can see a list of the courses which he is currently subscribed to. Following our current example, if the user then chooses the course he has been made a teacher of (In this case, the “Adaptive SQL Course”), he will be forwarded to the following page:

![Moodle course configuration page](image)

As can be seen from the above screen capture, the teacher is now presented with multiple extra options. He can add new blocks for instance to the right-hand side of the screen, he can add any relevant announcements or news topics on the top-right which might be

---

67 This can be done by configuring the mail server to send a confirmation e-mail to the user, asking him to verify his account.

68 Following the above example, this will be either “Alan” or “Antonia”

69 He also has the option to view all other courses in Moodle, and in some cases, can subscribe to these also.

70 Note: In this screen shot, I have chosen firstly chosen the “Turn editing on” button on the top right of the screen. This gives a series of options to the teacher to add new activities and events to each section of the course.

71 These blocks can be a “Calendar”, a “Course Summary”, a list of the “Online Users”, or a summary of each week.
relevant to visiting students, or he can also choose from the multiple series of options on
the left of the screen. Finally, if he wishes to add a new activity or resource to a given
week, then he can simply select one of these from the drop-down menus and will then be
forwarded to an appropriate screen to configure this option.

**Adding an AHS course in Moodle**

Finally, if the course teacher wishes to add an adaptive AHS course to a given week of
his course\(^72\), he simply selects this from the drop-down list of “Activities” as shown
below:

![Adding an Adaptive Hypermedia course in Moodle](image)

When the teacher selects the “AdaptiveHypermedia” option, he will then be forwarded to
the following screen:

![Editing Adaptive Hypermedia](image)

As can be seen above, the teacher need then only enter the name of this adaptive course
(In this case “Adaptive SQL Course”) and when he has finished, he simply clicks on the
“Save Changes” button. This will add a new AdaptiveHypermedia activity for this week,
which all users of this course can then access. Once the teacher has clicked on the “Save
Changes” button, he will be forwarded to the Adaptive course, where he will be either
asked to fill out the pre-test questionnaire (If this is his first time using the course), or if

---

\(^{72}\) This is the option which I have encoded, as described in the earlier chapters.
not, then he will simply be forwarded to his personalized AHS course within Moodle (As shown below).

Once this AHS activity has been added, as mentioned above, any member of this “Adaptive SQL Course” can then access the AHS course, once they log into Moodle. As described earlier, they will be given a pre-test if it is their first time to use the course, otherwise, they will be forwarded automatically to their personalized course. They will also have the option at any stage during their learning to rebuild their course, or likewise, to take the “Assess” option in the AHS course, which will grade them on their overall knowledge of the material covered. Bearing all this in mind, it is evident that the incorporation of this AHS activity greatly increases the overall learner’s experience. Not only does he/she have access to all the options provided by the teacher within Moodle\(^73\), but now also has access to a fully personalized AHS course, which he/she can modify at any stage of their learning to suit their own needs. Thus, through integrating both the CMS and AHS systems, the overall e-learning experience of the user is rather exceptional. All the more so, in that another AHS course can just as easily be added to another course within Moodle. For instance, if the user is taking courses in both Databases and Maths, and an adaptive course in Maths is available, the Maths teacher can just as easily plug-in the AHS Maths course to his Moodle course, thus offering the best of both worlds to the users.

**Viewing course logs and tracking information.**

\(^{73}\) For instance, course notes, over-head slides, questionnaires, forums, tests etc.
The final option I would like to illustrate is the tracking functionalities I have added to Moodle, which can be used to track a user of an AHS course. In the following example, I show how a course teacher (In this case, the user “Alan”) can use the “Logs…” feature of Moodle to view specific tracking information on a given user. First off, the teacher selects the “Logs…” hyperlink from the left-hand side of the screen on the main course page. Once he has done this, he is forwarded to the following screen:

As you can see, the teacher is given multiple options here. Firstly, he can use the inbuilt functions of Moodle to view activity records for All Participants, or a given user on a given day. He can also restrict this to only show logs for a specific activity, or alternatively, he can choose to only show live logs from the past hour. The second part of the above diagram displays the extra functionality I have encoded for switching tracking on or off for a given user of the AHS course. For this, the teacher need only select a given user from the drop-down list, and then choose the “Select user(s)” button. If tracking has already been activated for this user, then the system will indicate this appropriately. Otherwise, tracking will be switched on for that user.74

Once tracking has been enabled for a user, the teacher can once again use this “Logs…” page, only this time, he can select the specific user from the top section, followed by the date he is interested in, followed by the specific activity (In this case, the “Adaptive SQL Course”). Once he chooses the “Show these logs button”, he will be forwarded to a new screen, which will contain Moodle’s default log information75, and also some additional information for the AHS course under the heading “Additional tracking information from AHS SQL course for: Alan Tierney”.76 This can be seen in the following screen shot:

74 This is done by setting a Boolean value of 0 or 1 in the “mdl_ahs_tracking” table, as described in Chapter 3.
75 As seen in the top part of this diagram, under the heading “Displaying 7 records”
76 I encoded this additional information feature to show the activity logs for the AHS course.
As can be seen from the above, the user’s progress through the AHS course is recorded in a sequential manner. One can easily see which course section and subsection he has visited, as well as the page number in this sequence and the date and time he visited it at. Thus, through using this additional feature, a course administrator or teacher has access to the full functionality of Moodle’s default tracking options, as well as the additional tracking information returned by the AHS. This shows the final point of integration which I chose to implement, and as you can see from the above, it worked out rather successfully.
Section 4.2: Overall evaluation.

In this section, I will give an overview of the entire integrated system and comment on some of its good and bad points.

As outlined in chapter 3, having investigated both systems thoroughly, I decided to concentrate on four main points of integration between Moodle and the AHS system for this project. These were namely: single user sign-on, updating of the learner model from Moodle, tracking capabilities (Using Moodle to track a user’s progress in the AHS course) and assessment capabilities. For the first of these, a user of Moodle can log in to Moodle and launch his adaptive course at any stage, without having to manually log in to the AHS. This point works very well in my implementation, as the user does not even realize that two separate systems are involved in this process. As far as he is concerned, when he chooses the AHS course within Moodle, it is merely another part of Moodle, like the forum, quiz, wiki’s and other options. In practice, the user’s unique username and password is passed from Moodle to the AHS, and this is then stored in the AHS’s authentication file “userlist.xml”. It is worth noting however, that although this implementation works smoothly, it is by no means the most secure. In this implementation, the username is extracted from Moodle and a generic password is created for the user. This is then added to the AHS database through XUpdate. The result of this is that the user’s details are being transmitted openly over the network. For a future implementation, this system might be improved upon by using some form of encryption (For instance md5, or RSA), to encrypt both the username and password before sending. The current implementation, although quite insecure, seems to work quite well however.

The second point concerns the updating of the learner model. Again, this is done behind the scenes, and as far as the learner is concerned, this information does not concern him. For the interested reader however, some points are worth noting. The learner model is generated in Moodle using a questionnaire written in PHP. This questionnaire then creates an XML file representing the learner model, which is then uploaded directly to the AHS database. Similar to the user details mentioned above, this file is sent over the network without any form of encryption. Ideally, the file should first be encrypted (Or compressed), and then sent to the database, but due to time constraints \footnote{And the fact that this would really be outside the scope of this paper.}, I was unable to implement this feature successfully.

The third point of interest concerns the implementation of the tracking feature. Currently, this tracking feature works extremely well, and produces a full list of a user’s activities within the AHS course. A course administrator is given the option to enable or disable tracking on a user at any stage, and can view the results of tracking that user in a readable and efficient form (As shown in section 4.1). It is worth noting however, that this information is not very informative of the user’s overall progress in the system. The
results are presented simply as a table to the administrator or course teacher, and based on this, the teacher must scan through the data to check for any individual patterns etc. In an idealistic situation, the system should produce more intelligent results from this data. For instance, an extra component could be written to show graphs of the user’s progress and the number of subjects covered. Likewise, the system could produce a list of the topics the user spent a lot of time on, or very little time on, thus allowing the course administrator to improve on or eliminate certain pieces of content offered to the user in the future. As the aim of this project however was to demonstrate that tracking was achievable from within Moodle of the AHS course, the author felt that these extra features were outside the scope of this paper. It should be noted however that the tracking results are stored within Moodle’s database, so in the case of future development, these extra features could easily be added, should the developer so wish.

Finally, as for the assessment option. The assessment option is provided to a user of the AHS course as a means to grade his overall knowledge of the material covered. He should take this multiple-choice assessment test when he feels confident that he has covered the relevant material. Once he has taken the test however and his grade has been recorded, he is also given the option to take the test as many times as he wishes, thus giving him the option to use it as a learning tool against which to measure his progress. This author feels that the assessment option works extremely well, and is an exceptional tool for both the course teacher (Who can use it to grade students) and the student (Who can use it as a learning tool). As a possible improvement however, it may be good choice to split the assessment test into more sections. Currently, the user is simply given 20 multiple-choice questions which he then answers and is given feedback on. The test itself however is simply encoded as a single JSP page, thus making it rather static within the system. A possible improvement would be to split the assessment test into smaller components, which could later be re-used by another course to create a new or modified test. This idea follows the same ideology as the AHS content, whereby each piece of content is a stand-alone unit which is described by its meta-data. This content can then be accessed and combined with other content by the AHS to form a larger course. If this were done with the assessment questionnaire, it might lead to greater reusability of individual components, and thus make it easier in the future to create new and improved tests. For the current implementation however, the simple JSP page which is offered works rather well and provides the basic functionality which this author set out to achieve initially.

In conclusion therefore, I successfully implemented each of the key-points of integration which I had initially envisioned for this project. Some minor improvements (As suggested above) may be made in future developments of this project, but for the time being, this author is quite content with the overall functionality and degree of integration between both systems which was achieved.
Chapter 5: Conclusions & Future Work.

In this final chapter, I will restate what my main objectives were for attempting this project. I will outline how my chosen design has successfully achieved what I set out to do initially and how it may have far reaching consequences for future work in this field. I will then conclude with a few closing remarks and comments about some lessons learned during the course of this project.

Section 5.1: Overall Achievements.

In this paper, I set out to prove that it was possible to achieve a successful integration of the two software systems, Moodle and APeLS, and in so doing, to combine the strengths of each into a single and seamless integrated system. By integrating both systems, the strengths of Moodle, as a well-founded and robust CMS are combined with the flexibility and adaptivity of the APeLS system, thus providing a prospective institution with the best of both worlds. I firstly presented some of the main CMS and AHS contenders currently on the market and their respective strengths and weaknesses, and then explained why I chose Moodle as the most appropriate system to use for integrating with another AHS. Following this, I gave a description of two possible design strategies which could be used for my implementation. The first of these, which is based on Conlan, Wade (2002) was used in a previous attempt at integration, but as mentioned earlier, this author felt the same approach would not work very well with Moodle for this project. The second, which I devised myself following an intimate study of both systems, is based on four main points of integration, namely: single user sign-on, updating of the learner model, assessment options and tracking options. As stated previously, these key points demonstrate the interaction between both systems, as a learner can seamlessly launch and interact with the AHS course from within Moodle and not even be aware that a separate system is involved. The passing of learner model information, as well as tracking and assessment information between both systems was also achieved successfully, proving that the integration does indeed function as this author had initially envisioned. I have also demonstrated how the integrated system functions in Chapter 4, and as can be seen, it is both highly intuitive and simple to use for a prospective course administrator and prospective students. In this chapter, I also provided some evaluation of the system and made some comments on possible improvements which could be added to the system at a later date.

Overall, this author feels the final result of this project works remarkably well. The AHS can be easily “plugged-in” to Moodle and information is then shared back and forth

---

78 As indeed can any other AHS.
between both systems in a simple and efficient manner. Given also that Moodle is completely Open-Source, it is possible to add to or remove any of the options I have encoded during the course of this project. The extent of detail or specific features one course teacher may desire over another is difficult to determine, but as is the nature of this CMS, any additions in the future can be made rather simply. I will now discuss some possible additions which this author feels might be of interest for a future development.

**Section 5.2: Future work.**

Based on the current integrated system, some interesting additional future work in this field might be to investigate a tightening of the security features between Moodle and the AHS. As mentioned earlier, information is passed between the two systems in an open form, thus leaving the systems rather vulnerable to attack. Were this information to be firstly encrypted at the sending side, and then decrypted at the receiving side each time (through use of a public/private key or other form of encryption), the overall stability and security of the integrated system would be greatly improved. These features however were felt to be outside the scope of this paper, but would certainly make for some interesting future work.

Another interesting field of research might be to attempt an integration, or some form of information passing between the above system and Brusilovsky’s KnowledgeTree system. For instance, as in the above approach, the learner model could be generated in Moodle and in combination with one of the activity servers of KnowledgeTree, a course could be generated and displayed within Moodle. Essentially, this would mean that Moodle would act as a learning portal for the KnowledgeTree service. To this author’s knowledge, this approach has not been attempted thus far, but would certainly make for an interesting project.

**Section 5.3: Concluding remarks.**

The field of eLearning and educational systems like those described above is indeed a fascination and innovative field. By integrating two or more existing stable systems as has been described in this paper, the potential for content reuse and improvements in the overall quality of education offered is greatly increased, thus saving a great deal on time-consuming and expensive content creation, whilst also providing the learner with more advanced and interesting ways of learning. This paper has outlined one possible integration system, which works quite well and efficiently. I am sure however, that alternative approaches will be tried in the not so distant future, some of which may be just as useful (If not even more so!) as the one outlined here, or equally, new and improved systems will be developed to cater for a user’s and teacher’s needs. The field is still relatively young, but nonetheless, is sure to produce some very interesting results in the near future.
This author for one has gained some interesting insights into online learning and distributed systems as a result of this project. Though at times, the encoding and design of the experimental integrated system may have proven rather frustrating (Or at times, simply overwhelming!), in retrospect, I have benefited greatly from this experience and feel it has given me a good taste for developing new and improved systems. Perhaps in the coming years, I shall continue to work in this field, and produce an alternative and even better eLearning system than that described above!

In conclusion, I hope I have managed to convey my interest in the subject area to the reader, who has hopefully also learnt something along the way.
Bibliography.

• The Apache Software Foundation (http://www.apache.org/)
• The Apache Jakarta Project (http://jakarta.apache.org/)
• PHP (http://www.php.net/)
• EasyPHP (http://www.easyphp.org/)
• Java Server Pages Technology (http://java.sun.com/products/jsp/)
• JESS, the Rule Engine for the Java platform (http://herzberg.ca.sandia.gov/jess/)
• Extensible Markup Language (XML) (http://www.w3.org/XML/)
• XQuery 1.0 : An XML Query Language (W3C Working draft 04 April 2005) (http://www.w3.org/TR/xquery/)
• XUpdate – XML Update Language (http://xmldb-org.sourceforge.net/xupdate/)
• LMS http://www.clomedia.com/content/templates/clo_feature.asp?articleid=91&zoneid=29
• Fretwell-Downing Education http://www.fdgroup.com
• IMS Content Management, Version 0.1, June 2000

46


• Conlan, O.; Wade, V.; Bruen, C.; Gargan, M. “Multi-Model, Metadata Driven Approach to Adaptive Hypermedia Services for Personalized eLearning.” Second International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, Malaga, Spain, May 2002,


• Conlan, O., Wade, V., Dagger, D. “Personalisation for All: Making Adaptive Course Composition Easy”.


• O’Connor, A., Wade, V., Conlan, O. “Context-Informed Adaptive Hypermedia”

• Zakaria, M.R., Moore, A., Stewart, C.D., Brailsford, T.J. “Pluggable user models for Adaptive Hypermedia in Education”