catalyst IT LIMITED ©

Technical Evaluation

of selected

Learning Management Systems

Table of Contents

1. INTRODUCTION

This document is part of the *Evaluation of Learning Management Software* undertaken as part of the *New Zealand Open Source Virtual Learning Environment* project. One of the goals of the project is to select a best-of-breed Open Source Learning Management System (LMS) for development and large-scale deployment.

The criteria for this review expands on a subset of those criteria set forth in the *Evaluation of Learning Management Software*, focusing on the technical aspects of the systems.

In an effort to make this review relevant and accurate, a significant effort has been made in gaining *hands on* experience with each software package rated, and their related communities and documentation.

1.1 LMSS EVALUATED

1.1.1 Atutor

Platform: Apache, PHP, MySQL.

Version: 1.3.3

ATutor is a promising system that provides good documentation, ease of installation, and strong potential for development. While the user interface may not seem intuitive to many, the overall functionality is good (and wide open/modular) and the development team is committed to standards. The system is also install-friendly and receptive to new language versions.[1]

ATutor is one of very few LMS' that support learning object repositories. ATutor is very strong on standards and can import external content in IMS/SCORM format.

ATutor scores highly for openness. It is written in a modular format. It has many features and rates highly for usability, including accessibility for learners with disabilities.

1.1.2 Moodle

Platform: Apache, PHP, MySQL/PostgreSQL.

Version: 1.2 Beta

Moodle is one of the most user-friendly and flexible open source courseware products available. It has excellent documentation, strong support for security and administration, and is evolving towards IMS/SCORM standards. The key to Moodle is that it is developed with both pedagogy and technology in mind. Moodle has a robust development and user community. Great with languages although some development may be needed for robust handling of MathML and enhanced tracking features. Still, this program receives a high recommendation.[2]

1 http://www.xplana.com/whitepapers/archives/Open_Source_Courseware
2 http://www.xplana.com/whitepapers/archives/Open_Source_Courseware

Moodle is a student-centred course management system designed to help educators who want to create quality online courses. The software is used all over the world by universities, schools, companies and independent teachers.3

1.1.3 Ilias

Platform: Apache, PHP, MySQL

Version: 3 Beta

Ilias is an object-oriented course management system. Its user interface revolves around a desktop metaphor. Widely used in Europe and China, it has good internationalisation features. Version 3, under development, incorporates support for IMS/SCORM packages.

1.2 MAJOR EVALUATION CRITERIA

Working with the *Evaluation of Open Source Technologies* paper, and with a view towards long-term development and maintenance, the overall criteria for this review are:

- Overall architecture and implementation
- Interoperability
- Cost of ownership
- Strength of the development community
- Licensing
- Internationalisation, localisations
- Accessibility
- Document transformation

1.3 SUMMARY OF OUR FINDINGS

The short-listed systems show significant differences in their design, architecture and implementation. Regardless of the fact that they share a common underlying framework, the impact of the differences is important.

On the overall evaluation, Moodle shows a clear advantage, particularly in criteria that is critical to the long-term viability of the system. The primary differentiating advantages of Moodle are as follows:

1.3.1 System Architecture

Moodle's main strength is its simple but solid design and architecture developed by Martin Dougiamas.

Moodle's architecture sets an excellent foundation, following good practices of low coupling and high cohesion, which the other LMSs fail to achieve. This yields a system that is simple, flexible and effective; and easily accessible to developers.

A modular, extensible architecture is a key criteria for the project. The Moodle approach is pragmatic, using intelligent strategies e.g. the code is split into modules and standard libraries, and a standard API for modules. Authentication is

3 COL LMS Open Source Report - July 2003, pp17

modular and separate from the rest of the modules. This will allow easier integration with a portal framework, and an interface to student management systems. By comparison Atutor is not modular with authentication strongly tied to the database throughout the system code.

1.3.2 Community

There is a lively developer community built around Moodle, with programmers other than the main maintainer contributing sizeable modules and fixes; a second criteria the other systems fail to meet. The project code is mature.

Atutor lacks a true development community possibly due to the closed approach their developers have taken.

1.3.3 Conclusions

Moodle does have limitations, notably it lacks SCORM support, and its roles and permissions system is limited. However these problems can be fixed, and are part of the project roadmap.

Atutor, while strong in features and usability, has serious architectural problems although some features in Atutor warrant further investigation (such as IMS package creation), and may be candidates for porting to Moodle.

Ilias, while promising, has a complex architecture with tight coupling that is hard to work with and debug. The code is new, and lacks maturity. The developer community for Ilias is very small outside the core team. Some features in Ilias deserve to be reviewed for porting to Moodle.

Summary	ATutor	Ilias	Moodle	Comment
Architecture and implementation	Weak: no modularity	Complex: tight coupling	Good: high cohesion, low coupling	Considerations in architecture and implementation impact all other areas.
Interoperability	Bad	Good	Average	Overall interoperability has good potential, but is not implemented.
Cost of ownership	Medium	High	Low	Correlates with architecture and implementation findings.
Strength of the community	Low	Medium	High	Related to openness of process.
Licensing	GPL	GPL	GPL	Ilias offers optional features that depend on non-FOSS software.
Internationalisation	Weak	Average	Good	Good internationalisation support is critical for Maori and Pacific Island communities
Accessibility	Excellent	Bad	Average	Good accessibility is critical for users of alternative browsers (screen readers, etc.)
Document transformation	No	Average	No	Complex, needs further analysis.

1.4 RECOMMENDATION

Select Moodle as the core LMS for the *New Zealand Open Source Virtual Learning Environment* project.

2. LMS EVALUATIONS

2.1 GOALS

The objectives of this document are

- To define detailed technical criteria for the evaluation of Open Source Learning Management Systems.
- To gain understanding of the design, architecture and implementation details of the short-listed LMSs, with a view towards long-term development and maintenance.
- To evaluate the short-listed LMSs against set criteria.
- To engage members of the Open Source community in the process where relevant.

2.2 METHODOLOGY

The following approach was used:

2.2.1 Develop a Technical Evaluation Criteria

Based on the Initial Evaluation of Open Source Technologies paper provided by Open Polytechnic of New Zealand, the criteria focuses and expands on the technical aspects of the systems. For each criteria rated, a discussion is provided, indicating what is covered, its importance, and a check-list of the aspects observed.

Each overall criteria is presented with a brief discussion, with summary tables providing a weighted rating.

2.2.2 Deploy and Evaluate

The short-listed systems are deployed in a test-bed environment, where they are used and maintained for the duration of the review. The deployment stage involves set-up, configuration, branding and minor customisation, thereby providing practical experience with each tool. As much as possible, the evaluation criteria is checked against the actual performance of the system, rather than the published feature list.

Given the goals of understanding the challenges of long-term maintenance and development, every chance to study the feasibility of changes and extensions is used.

2.3 OVERALL ARCHITECTURE AND IMPLEMENTATION

To assess software systems for long-term operation, development and support, the key criteria are simple: quality of architecture and craftsmanship. This is comparable to considering other complex man-made objects for their long-term viability.

The modern definition of software architecture defines it as the structure which comprises software elements, their properties, and relationships [BASS2003] and [ANSI1471]. Unfortunately, it is only recently that the craftsmanship put into the system

has been deemed to be as important as its architecture [MCBREEN2002].

2.3.1 System is scalable

Scalability indicates a system's ability to maintain quality performance or service under an increased system load by adding resources. In this evaluation we are interested in the characteristics of the LMSs in terms of vertical and horizontal scalability, covering scenarios that range from a large server or server cluster handling a hundred educational institutions to a small legacy server for a single school.

To scale vertically or *scale up*, means to add resources to a single node in a system, such as adding memory or a faster hard-drive to a computer. To scale horizontally or *scale out*, means to add more nodes to a system, such as adding a new computer to a clustered software application. A truly scalable system must also *scale down* to more constrained resources, to allow small scale deployments.

The three LMSs in the review are strikingly similar in their scalability profile. While most web applications have a common architectural model, the LMSs evaluated use the same middleware and RDBMS solutions.

They embody, however, different strategies database usage, caching, and memory management. These can have a significant impact on scalability and performance.

System is scalable.	**ATutor**	**Ilias**	**Moodle**	**Comment**
Minimum requirements				
Scale up: high end servers	Excellent	Excellent	Excellent	Similar architecture
Scale out: allows caching	Good	None	Incomplete	Hard to achieve with PHP-based solutions.
Scale out: multiple app servers	Good	Good	Good	Better performance and availability. Similar limitations – Session handling and data directory.
Scale out: multiple database servers	Good	Good	Very good	Better availability and reliability, less performance. All support MySQL with log-shipping. Moodle also supports PostgreSQL with log-shipping and phased commits. None segregates read-write from read-only.
Scalability constraints	DB writes	DB writes	DB writes	Similar architecture.

2.3.2 Scale down – minimum requirements

What is the minimum hardware set-up for a small operation with 300 students, expecting a maximum of 30 active users at peak usage time? Systems are rated running on a small server running both the middleware and DB software.

See Appendix A – Benchmarks.

2.3.3 Scale up – high end servers

Does the system derive full benefit from high-spec servers? Using a modern Linux kernel (2.4 and newer) all the component of the tested applications make good use of the available resources.

There are no artificial limits to memory, CPU or disk usage, nor significant use of locks or "mutexes". The three LMSs show no significant differences, mostly due to the fact that they use the same framework.

It must be noted, however, that the application server does not derive particular benefits from high-end 64-bit CPUs. Properly configured database servers do benefit from 64-bit CPUs.

2.3.4 Scale out – allows efficient caching

Does the system derive full benefit from front-end reverse proxies and downstream proxies?

Reverse proxies, external proxies, and browser-level caching increase scalability by reducing the load on servers and bandwidth consumption significantly. To benefit from this effect, that the system must be able to identify what content can be cached safely. Note that this only applies to non-dynamic content, such as images and static learning objects.

This is evaluated at three levels. The application must send the cache control HTTP headers (cache control, expires, etc.). It must send the Content-Length HTTP header, which is required by most proxies to perform caching – this is particularly hard with PHP. It must also answer with a 203 (content hasn't changed) when the request carries an if-modified-since header.

Finally, end-user browsers tested must show content is cached when relevant.

ATutor: Good. Delegates serving of static objects to Apache, which complies with all requirements.

Ilias: Makes no use of cache control directives.

Moodle: Incomplete. Good use of cache control directives, fails to send content-length and 203 responses.

2.3.5 Scale out: can use multiple application servers?

Cluster with multiple application servers can be used to spread the processing load required by the application. They also provide higher availability. Multiple application servers are usually deployed behind one or more reverse proxies that distribute the requests, either randomly or with load-balancing strategies.

The three LMSs share their middleware software: Apache and PHP. This configuration is extremely flexible in terms of multiple application server set-ups. There are two architectural traits that limit their ability to scale out: the use of a disk-based “data” directory, and session management.

The three applications use a “data” directory in addition to a database. This data directory must be set up on a network drive (using NFS, for instance) that all application servers must be able to read/write to. This network partition, if heavily

used, can become a bottleneck.

An alternative solution to the “data” directory problem is to store this data in the main database, while maintaining a “file cache” on each application server, from where the files are actually served.. This increases the load on the database server, especially for file writes (when new files are uploaded), but the approach has significant advantages in terms of application architecture.

Session management must be implemented using a mechanism whereby all application servers share the sessions. This is usually achieved using a temporary database or temp files on a network drive. Alternatively, the traffic director layer could implement “session affinity” to ensure sessions always go to the same application server.

All LMSs use PHP's native session management, which can be configured to use database storage for outward scalability.

ATutor: Data directory stores course materials (few writes).

Ilias: Data directory stores course materials (few writes).

Moodle: Data directory stores course materials (few writes), and there's discussion of moving it into the database, with an on-disk file cache.

2.3.6 Scale out: can use multiple database servers?

A cluster with multiple database servers can be used to increase reliability and availability – usually at a small to medium cost in performance. The performance bottleneck on the database server scenario is that all changes (“writes”) to the database must go through one single authoritative server, to ensure there are no conflicts in the data changes.

This depends heavily on the RDBMS in use. The three LMSs share the ability to use MySQL. MySQL can be set-up in a master/slave set-up, where all the changes are made on the master database server, which sends logs of all the updates made to one or more slave servers. These provide redundancy and, if coupled with a switch-over mechanism, can be a High Availability (HA) solution. This technique is known as log-shipping.

Using log-shipping, if there are many database “read” operations, the application can use the “slave” databases. This usually requires a lot of work on the application, and yields marginal benefits.

PostgreSQL can be set-up to perform log-shipping. With more work, it can be configured to perform a 3-phased commit, where the same changes are made on 3 or more databases at the same time, and only go through if all the databases succeed. This improves reliability, but has a high cost in performance.

ATutor: Can use MySQL with log-shipping. Hard to segregate read and write operations.

Ilias: Can use MySQL with log-shipping. Hard to segregate read and write operations.

Moodle: Can use MySQL with log-shipping. Can use PostgreSQL with log-shipping, or with phased commits. Uses ADODB to manage DB

connections, (easier to segregate “write” from “read” operations).

2.3.7 Scalability constraints

The constraints that determine the system's upward, outward and downward scalability are strongly tied to the framework and architectural choices. Therefore, the three LMSs evaluated have very similar constraints.

Once all components are tuned optimally, database writes are the bottleneck. This is a common trait of web-based applications. However, using modern hardware and RDBMS software, it is unlikely that deployments using high-end hardware will be constrained by database write performance.

Upward: CPU processing power, memory, HD seek speed.

Outward: Single master database architecture, cluster network speed.

Downward: RAM, HD seek speed, HD size.

2.3.8 System is modular and extensible

We consider a system to be modular when its implementation is comprised of modules with high cohesion and low coupling. This is a strong indicator of the inherent maintainability and adaptability of a software system.

In short, coupling is a measure of how interrelated two software components are. Cohesion is a measure of how related the functions performed by a software component are.

High coupling implies that, when a component changes, changes to other components are likely. Low cohesion implies difficulty in isolating the causes of errors or places to adapt to meet new requirements.

System is modular and extendible	ATutor	Ilias	Moodle	Comment
Has module architecture	No	Yes, suffers tight coupling.	Yes, simple and complete.	Moodle achieves the goal of loose couplig/high cohesion.Ilias has a tighly coupled architecture/
Core functionality in modules	No	Yes, inconsistent.	Yes	
Solid support for modules	No	No	Yes	
Internal API	No	Ad-hoc	Yes, well documented	
External API	No	No	No	Possible to develop external APIs

Modular architecture

Does the application have a modular architecture? An ideal architecture has

simple but well defined internal layout, and the modules have small, well defined interfaces, high cohesion and loose coupling.

The modular architecture must be documented, simple and elegant. There is no requirement that is must be object-oriented or functional. The performance/resource trade-offs must be acceptable, and it should not impose artificial bottlenecks or inefficiencies.

ATutor: Monolithic architecture. All functionality resides in the core of the application. Extensions must be made part of the application, and are tightly coupled.

Ilias: Architecture is confusing and undocumented. Each module is implemented as a class, exposes an arbitrary interface, and behaves as a special case.

Moodle: Simple, straightforward and complete module architecture. The code is split in modules (*mod* directory) and libraries (*lib* directory). Modules must expose certain entry points as functions and particular files. Authentication is modular and segregated from the rest of the modules.

Core functionality in modules

Highly modular applications provide most of their end-user services as modules. Experience indicates that systems that do not follow this rule hint at a module infrastructure without extensive use, therefore incomplete and immature.

ATutor: There are no modules as such. All functionality resides in the core application.

Ilias: Most functionality resides in ad-hoc modules without interface consistency.

Moodle: Most of the end-user functionality is in modules. While roughly 50% of the code is in core libraries, 30% is in modules.

Solid support for modules

Solid support for modules means that the core application sets up a framework of services. Modules must have means to register with the framework (for user-level navigation, compatibility checking, etc.), set up database tables to store their data, schedule backend processes, retrieve user data, presentation templates, and offer configuration interface (either for the user or the administrator). If the application is being upgraded, modules must be able to execute upgrade routines.

Additionally, for modules to maintain their loose coupling with the core application, the framework must offer other well documented services such as retrieval of configuration data, database access, logging, authentication and authorization.

ATutor: There is no framework in place.

Ilias: There is no framework in place.

Moodle: The framework is well developed and mature. Modules can set-up their own DB tables, run upgrade scripts, cron scripts, etc.

Internal API

Internal mechanisms must be documented and expose functions or methods that act as entry points to trigger its functionality. These must follow documented (or well known) conventions, and be consistent and straightforward.

ATutor: There is no well defined internal API.

Ilias: Internal classes are documented using PHPDoc, but do not constitute an organized API.

Moodle: The modules infrastructure imposes a standard API for modules. Libraries are well documented.

External API

Key services are exposed to suitably authenticated programs that need to be integrated with the LMS but are not on the same server or are not part of the same application. This must be done over an RPC or ORB interface; popular implementations of such interfaces are XML-RPC and SOAP.

Surprisingly, none of the systems reviewed exposes an external API. It is possible, however, to build it for any of the three system using existing PHP libraries.

2.3.9 Multiple installations on a single platform

Can one hardware set-up – at any point of the scalability range -- handle multiple instances of the system independently? To achieve this configuration files (covering hostname, database, installed path and, data directory) and logging services must be abstracted.

Ideally a single installation of the program can be made to run with different configuration files.

ATutor: Can run multiple instances in different URLs. Configuration is restricted to a single file and well abstracted. Logging services are provided by Apache and can be segregated per instance.

Ilias: Can run multiple instances in different URLs. Configuration is spread across several files, but well abstracted, and prepared to run several instances from the same set of config files. Logging services are provided by Apache and also internal logging facility; can be segregated per instance.

Moodle: Can run multiple instances in different URLs. Configuration is restricted to a single file and well abstracted. Logging services are provided by Apache and can be segregated per instance.

Multiple installations on single platform	ATutor	Ilias	Moodle	Comment
Overall	Excellent	Good	Excellent	
Scripted administration	Excellent	Limited	Excellent	Ilias offers admin UI, cannot be easily scripted.

Multiple installations on single platform	ATutor	Ilias	Moodle	Comment
Table prefixes	Yes	No	Yes	

Scripted administration

It must be possible to automate the creation, administration and deletion of server instances using common Unix scripting tools. Scripted upgrades should be possible.

ATutor: Possible to manage through shell scripts, some parts of the installation/upgrade process may need extra work.

Ilias: Possible to manage through shell scripts, configuration files and database set-up more complex.

Moodle: Simple to manage through shell scripts. Version upgrades can be scripted across many instances.

Table prefixes

The use of configurable table prefixes allows many application instances to share one database using segregated database tables. This benefits allows many instances to be run in environments limited to one database (virtual hosting at an ISP, for instance), and also derive a marginal benefit in the use of persistent database connections.

ATutor: Yes.

Ilias: No.

Moodle: Yes.

2.3.10 Has reasonable performance optimisations

Strategies that maximize performance and make efficient use of resources are clear signs of good architecture and craft. Furthermore, application systems must be deployed at reasonable cost for large user bases.

Strategies and practices relevant to the PHP engine can and should be used to ensure the system delivers acceptable performance and scales well.

Also see *Allows efficient caching* under *System is Scalable*.

System has reasonable performance optimizations	ATutor	Ilias	Moodle	Comment
Compatible with precompiled code caches	Yes	Yes	Yes, limited compatibility	Moodle is not compatible with one of the code caches available.
Persistent connections	Yes	Yes	Yes	

Compatible with precompiled code caches

The PHP engine allows caching of precompiled copies of the code to speed up performance and reduce CPU load on the server. This is similar to similar engines, such as mod_perl or ASP.

This caching mechanism gives servers a great boost, but conflicts with some features of the PHP language. For applications to be compatible, the code must not use those features. Zend Optimizer, IONCube PHP Accelerator and MMCache are available as caching systems. They conflict with programs that make use of output buffering and output filters.

Alternatively, it is possible to perform some caching of (not yet compiled) code within PHP using PHPCache. This module places the same constraints on the application and delivers very marginal benefits on performance.

ATutor: Reported as compatible with Zend. PHPCache is implemented by default, can be turned off.

Ilias: Reported as compatible with Zend Optimizer and IONCube PHP Accelerator,

Moodle: Compatible with IONCube; shows conflict with MMCache.

Persistent connections

Do the applications use persistent connections to backend data sources?

Opening database connections is a step that is both resource-intensive and has a high latency. A busy server serving ten requests per second may be opening (and closing) several database connections per request. Persistent connection is a feature of the middleware that allows the database connections to be kept open, so that they can be used by subsequent requests, saving time and resources.

Connections to other backend services, such as LDAP and SOAP/XML-RPC services, can be optimised using persistent connections if the protocol allows.

Using the “table prefixes” feature there is also a marginal performance gain on set-ups that handle many databases. See “Table prefixes” under “Multiple installations on a single platform”.

ATutor: Database persistence: yes.

Ilias: Database persistence: yes. LDAP: No, but caches credentials.

Moodle: Database persistence: yes. LDAP: No, but caches credentials.

2.3.11 Look and feel is configurable

***Look and feel* is configurable.**	**ATutor**	**Ilias**	**Moodle**	**Comment**

***Look and feel* is configurable.**	**ATutor**	**Ilias**	**Moodle**	**Comment**
Headers and footers customisable / use themes	Yes, untidy.	Yes, limited.	Yes, as themes.	Ilias and ATutor have an untidy setup for headers/footers, where several files must be edited.
CSS and other applied styles are customisable / use themes	Good, as themes	Good	Good, as themes, well documented	
Theme scope	Site-wide, with course and user themes	User theme	Site-wide theme	Centralized theme control is better for wide-scale deployments.

Headers and footers are customisable or use themes

System administrators must be able to alter the headers/footers of the pages served by the application by editing a single set of files. Optionally, the customisations may be part of a theme framework, where the system administrator can define a mandatory theme for the application.

ATutor: Supports changes in headers and footers, without theme support. There are several sets of header/footer *includes*, used by different parts of the application, this is untidy and not well documented.

Ilias: Supports changes in the header only. The Ilias UI runs in a frameset, where the top frame, which provides the main navigation menu is configurable. This is simple, but limited. There is a themes framework, but the header is not covered.

Moodle: Supports changes in headers and footers, within a “themes” framework. One set of files controls headers, footers and for the whole application. The theme customisation is well documented and has many examples.

CSS and other applied styles are customisable/use themes

System administrators must be able to customize the overall *look and feel* modifying a CSS stylesheet and presentational HTML tags. CSS styles are preferred over HTML presentational tags for obvious reasons. A theme framework may support these customisations. Given the power and flexibility of CSS, the customisable tags must be well documented and straightforward.

Some applications supply templates for each logical page of the UI. This does provide more flexibility at the price of complexity.

ATutor: Yes, supports changes in CSS, within a “themes” framework. The theme customisation is straightforward and has many examples.

Ilias: Yes, supports changes in CSS, within a “themes” framework. Thetheme customisation is straightforward and has many examples. Additionally it offers page templates for customisation as part of the themes framework, but these are tied to the application internals.

Moodle: Yes, supports changes in CSS, within a “themes”framework. The theme

customisation is straightforward, well documented and has many examples.

Theme scope

Administrators must be able to control UI themes and/or customisations directly. Optionally, teachers or course author users can alter or personalise the look and feel on a per-course basis; and students may also be able to pick a theme or customize the UI.

Whenever a teacher or student user can pick a theme or otherwise customize the look and feel, the administrator must be able to override/disable this feature.

ATutor: Authors/teachers can set a look and feel (which includes colour/font scheme and page layout) for the course which overrides the default scheme. Students can override site and course defaults with their own choices. It is not possible to disable this.

Ilias: Any user can select a theme from a controlled set. Reducing the available themes locks down look and feel.

Moodle: Only administrators can alter the look and feel of the website through the theme mechanism. There is no user-level control of look and feel.

2.3.12 Security

The security of a web-based system consists of strategies that address three interdependent but well defined areas[GARF2003].

Server security covers strategies to secure the server against attack. Applications must be designed to avoid being an avenue of attack against the server.

User security and privacy must be preserved, the LMS system must keep communication and data private and avoid exposure of user's computers (client computers) to attacks.

Application security covers restrictions on what users can do (to prevent attacks and misuse) and data privacy. To prevent unintended access, the application must have mechanisms for *authorization* (checking of credentials) and *authentication* (checking of right to perform an action).

In these three areas, secure applications implement strategies that monitoring and log attack attempts and minimise exposure.

Security strategies involve the whole stack, from the operating system up through the libraries, framework, components and application layers. The Apache/PHP framework provides outstanding security features (not covered explicitly in this review), which these applications should take advantage of.

Security	ATutor	Ilias	Moodle	Comment

Security	ATutor	Ilias	Moodle	Comment
User privacy-security: SSL Integration	Good	Good	Good	
Server security: permission lock-down.	Good	Bad	Good	Ilias requires lax permissions on config files, and cannot be easily managed otherwise.
App security: authentication	Good	Good	Good	PHP Session management needs to be configured tightly.
App security: authorization	Limited	Good	Limited	
App security: logging and monitoring.	Bad	Bad	Bad	None offers logging of abuse attempts.
App security: Validation of input.	Good	Weak	Good	Security issues in input validation were identified during the review.

User privacy-security: SSL integration

SSL is the cornerstone of reliable identification, protection of authentication credentials and user data. It identifies the server reliably to the user, so the user knows the password is sent to a trustworthy server, and it all the communication between user and server (including passwords) from third parties [GARF2001].

Integration of the HTTPS protocol (HTTP over SSL protocol) happens on the webserver, the application has no need to be aware of it. Transparent and flexible use of SSL depends, however, on the application following standard practices when issuing redirects and serving binary files. Some applications do not work correctly under HTTPS.

HTTPS-aware applications can, on the other hand, make use of advances features of the protocol. Authentication of users can be done using authentication keys. This is a much safer alternative to passwords, and is useful for teachers and administrators accounts on the system.

ATutor: Works through HTTPS. Not HTTPS-aware.

Ilias: Works through HTTPS. Not HTTPS-aware, its modular authentication system can be made HTTPS-aware.

Moodle: Works through HTTPS. Not HTTPS-aware, its modular authentication system can be made HTTPS-aware.

Server security: permission lock-down.

The system must run correctly with its files set-up with strict permissions, so that a minor security flaw cannot modify the program files, or write to the directories where the program is stored. No *setuid* or

setgid programs[4] should be required.

Additionally, any directory where the program will write files must be configured non-executable, and only by the webserver user should have *write* access.

The three systems offer a web-based configuration to simplify the server administrator's job. Such facilities are convenient but unsafe, because they require lax permissions on the configuration and program files. It must be possible to keep strict permissions and perform configuration and upgrades via command line, and editing config files.

ATutor: Program files' permissions are correct. Requires lax permissions on one file for web-based configuration, but it is easily done securely by editing a config file. Data directory needs to be writeable by server, and must reside in the public web directory; this is a risk that may be alleviated by restricting access through apache configuration.

Ilias: Program files' permissions are correct. Requires lax permissions for web-based configuration, which cannot be easily done otherwise. Data directory needs to be writeable by webserver, but is segregated, and does not need to reside in public web directory.

Moodle: Program files' permissions are correct. Main configuration is easily done editing a config file, other web-configurable settings reside in database. Data directory needs to be writeable by webserver, but is segregated, and does not need to reside in public web directory.

Application security: authentication

Is the authentication process and authentication token handling secure?

Standard security practices focus on the handling of authentication credentials, and subsequent tokens to prevent replay attacks.

HTTP-AUTH is the authentication extension of the HTTP standard[5]. However, HTTP-AUTH is limited, not very secure, and does not offer a good user experience due to bad implementations in popular web browsers. Web applications do well in avoiding it [GARF2001].

Instead, the LMSs use a strategy known as *cookie-based authentication*: the user sends username and password through a form, and is issued an authentication token – stored in a cookie – to use in subsequent requests.

The three systems apply standard industry practices to authentication, and defer token generation and tracking unto PHP's native session management. This session management facility is capable of very good and secure token management. However it is configured by default with settings that make it open to replay attacks. In particular, the hashing algorithm needs a seed (which can be a

4 The terms *setuid* and *setgid* refer to programs that are configured to run with the privileges of another user, usually administrator privileges, and constitute a potential attack vector for privilege escalation attacks.

5 See http://www.rfc-archive.org/getrfc?rfc=2617

shared secret in the case of multiple application servers); but this is not enabled by default.

It is strongly recommended PHP session management is configured securely in public deployments of any of the reviewed LMS.

ATutor: Good. PHP sessions need configuration for better security.

Ilias: Good. PHP sessions need configuration for better security.

Moodle: Good. PHP sessions need configuration for better security.

Application security: authorization

After the user has been correctly authenticated, authorization mechanisms decide what the user is allowed to do.

The authorization systems should define available access rights, roles (as a set of access rights) and, where relevant, ownership relationships and groups. They may also define "access control lists" (ACLs), but this is not a requirement. Finally, roles should be configurable, not hard-coded (with the exception of the "super-user" account).

Only Ilias offers an authorization infrastructure. ATutor and Moodle manage to be useful without a proper implementation of flexible roles and user privileges. This is probably due to the well-defined profiles of the user-base, where the natural roles (students, teachers, authors and administrators) seem to be comprehensive enough to work in most scenarios.

ATutor: Very limited roles; hard-coded implementation; flexible ownership of courses. Roles are limited to student, teacher, content creator, administrator, and are hard-coded in the database schema and application code.

Ilias: Flexible support for roles, with role templates and user interface.

Moodle: Very limited roles; hard-coded implementation; limited ownership of courses. Roles are limited to student, teacher, administrator, and are hard-coded in the database schema and application code.

Application security: logging and monitoring

The application framework performs the core of the logging and monitoring for web applications. In this regard, Apache provides excellent logging and monitoring facilities. Certain attacks, however, can only be monitored and logged at the application level.

The systems are rated on their handling of:

- Dictionary attack against a single account.
- Attempt to access an object without access rights to it.
- Attempt to edit an object without edit rights to it.

ATutor: Operational errors are logged with the server log. Usage and abuse attempts are not logged.

Ilias: Operational errors are logged with the server log, plus an additional application log for application level error messages. Usage and abuse attempts are not logged.

Moodle: Provides an activity log, but only registers successful operations; as such it is only useful for monitoring of expected activity. Operational errors are logged with the server log.

Application security: Validation of input.

Does the system have strong validation of input? Does it effectively prevent users from abusing the system?

Software applications, as many other man-made systems, are vulnerable to users providing unexpected input. Systems that can be used anonymously must be hardened to validate all input from users.

In the case of web applications, lack of input validation renders applications vulnerable to HTML injection and SQL injection[6] attacks. HTML injection can lead to an attack against system users known as Cross Site Scripting (XSS)[7].

File upload forms also constitute a vulnerability point. Storage– and later retrieval– of files uploaded by users must have strict validation and sanitization of file names/paths to avoid *shell injection* attacks. Successful attacks on file uploads can result on remote users executing commands as the webserver user.

The PHP language implementation has, in past versions, encouraged questionable programming practices by way of a feature known as register_globals. Recent versions of PHP have register_globals disabled, and applications do not require it are safer.

Systems are rated on their handling of concrete SQL injection, XSS and shell injection attack attempts. We also consider unsafe practices such as use of register_globals.

The three systems tested have good escaping mechanisms against SQL injection attacks. They are far weaker in the face of HTML injection/XSS attack, in particular if coming from teacher or author account.

During the review two security problems were found and dealt with:

- A medium-risk shell injection vulnerability was found in Moodle. This vulnerability was fixed easily, and shared with the project developers, who replied immediately. The project maintainers had already identified the problem and had produced a similar fix.

- A high-risk shell injection vulnerability was found in Ilias. This vulnerability was partially fixed, and shared with the project developers.

ATutor: Students cannot post HTML, simplifying HTML-injection problems.

---

6 SQL Injection http://www.spidynamics.com/papers/SQLInjectionWhitePaper.pdf and Advanced SQL Injection http://www.nextgenss.com/papers/advanced_sql_injection.pdf

7 Cross Site Scripting FAQ http://www.cgisecurity.com/articles/xss-faq.shtml

However, teachers can perform HTML-injection attacks on all the website users. Filenames and paths are sanitized correctly.

Ilias: Students and teachers can post HTML; the HTML clean-up routines are weak and potentially vulnerable. Filenames and paths are sanitised correctly.

Moodle: Forums and other facilities that allow students to post HTML implement a very good clean-up. Content editing facilities for teachers and authors do not perform any validation. Problems in filename sanitization were fixed.

2.3.13 Modular authentication

Institutions deploying a LMS are highly likely to have a pre-existing infrastructure, one that is already maintaining authentication data, and maintaining segregated user lists is inefficient and error-prone. It is also very likely that a LMS will be coupled with other tools, web-based or otherwise, such as blogs and content management systems. Such scenarios make modular and flexible authentication a key feature, which we evaluate separately from overall system modularity.

Modular Authentication	ATutor	Ilias	Moodle	Comment
Authentication is modular	No	Yes, PEAR Auth	Yes, own modules	
Interoperates with standard authentication back-ends	No	Yes	Yes	Ilias offers a wider range of authentication back-ends.
Interoperates with external authentication ticket system	Yes	Yes	Yes	All use PHP's session management, which can use an external system.
Authentication services API	No	No	No	

Authentication is modular and straightforward

The authentication infrastructure needs to be modular to allow for alternative authentication back-ends to be used, through *drivers*, *plugins* or *modules.*

Institutions are very likely to require custom authentication protocols or business rules, therefore it must be straightforward to develop or customize authentication modules.

Systems are rated on documentation, sample modules and existing modules for clarity and simplicity.

ATutor: Authentication is **not** modular and strongly tied to the database throughout the system code. The ATutor code includes an authentication library (PEAR Auth module), oddly it is not used anywhere.

Ilias: Uses PEAR Auth library for authentication, which is modular and developed as part of the PEAR project. Modules are of medium complexity, but good examples and documentation are available.

Moodle: Authentication is modular, uses a very straightforward module format. There is good documentation and clearly written modules to serve as examples. The authentication modules are exclusive to Moodle.

Interoperates with standard authentication back-ends

The authentication infrastructure – regardless of modularity -- must be able to interoperate with different back-ends transparently. Common authentication back-ends:

- **Flat file**. A flat file containing user/passwords pairs, usually exported from other data repository.

- **DB**. A database other that where the LMS stores its data, hosted in a database server supported by PHP.
- **LDAP.** PHP can interoperate with Active Directory, Open LDAP and others.
- **IMAP/POP3**. Uses an email server for authentication, useful for institutions that already provide email services to students and teachers.
- **PAM**. Unix Pluggable Authentication Modules standard. PAM is itself a modular authentication infrastructure, with a host of plug-ins available, including unix-style password file, SMB/CIFS, LDAP, Radius, Kerberos, NetInfo, etc.

ATutor: Limited to its own database.

Ilias: Uses PEAR Auth libraries, which include modules for DB, Flat file, LDAP, IMAP/POP3, RADIUS, SMB/CIFS, SOAP, vPOPMail. Oddly, Ilias offers only LDAP in its configuration options, its infrastructure clearly supports all the modules.

Moodle: Defines its own authentication module structure, included modules are email (for self-subscribed, with email confirmation), DB, LDAP, NNTP (Usenet-style news servers), and IMAP/POP3.

Interoperates with external authentication ticket system

When integrating with other applications, it must be possible to configure the system to accept authentication credentials (in a cookie or session) provided by a different application, in fact overriding the internal authentication mechanisms.

An interesting technology in this space is the Security Assertion Mark-up Language (SAML), an OASIS Standard[8] for the exchange of authentication and authorization information. It embodies standard cross-website authentication practices in a standard XML-based protocol. Unfortunately SAML is in early stages, and not widely supported.

The three systems use PHP's native session support, which makes it straightforward to share session data with other PHP applications running on the same server or cluster. Session token interoperability with non PHP applications requires more work, but is achievable.

ATutor: Its internal authentication systems must be modified to work in this scenario.

Ilias: A new PEAR Auth module must be created to support this scenario. This should be straightforward.

Moodle: Provides an authentication module (called “none”) which allows this behaviour by disabling authentication checks. Minor customisation is required to ensure is achieved without security issues.

Authentication services API

To allow integration with other applications in scenarios where the LMS hosts the master login database, it must exposes its authentication API via SOAP, XML-RPC

8 Latest specification and discussion documents are available from http://www.oasis-open.org/committees/security/#documents .

or an equivalent protocol. This must be configurable and securable.

Alternatively, it must be straightforward to write a script to export the login information to a text file. In this case we evaluated the complexity of the database schema and potential pitfalls.

ATutor: Does not expose an API.

Ilias: Does not expose an API.

Moodle: Does not expose an API.

2.3.14 System is robust and stable

Robustness and stability are key concerns when deploying applications in large scale and long term. These criteria are observed not only in the passive count support issues during operation, but also on the proactive strategies taken by the application developers.

Issues covered under *Security* – such as logging and input validation – are also important to consider.

System is robust and stable	ATutor	Ilias	Moodle	Comment
Cross-browser support	Good	Good, requires frames	Good	Ilias does not work correctly with screeen readers and text-only browsers.
Not dependent on JavaScript	Yes	Yes	Yes	Moodle loses WYSIWYG HTML editor.
No maintenance downtime	Good	Good	Good	
Backup	Good	Good	Excellent	Moodle implements an XML based backup/restore facility in addition to standard DB backups.

Cross-browser support

The system must run correctly on all current, standard compliant web browsers (User Agents). HTML/XHTML and CSS should be validate and written to current best practices.

While minor compliance issues can be tolerated in a complex web application, browser specific features and implementations are not acceptable.

Reference browsers are MS Internet Explorer 5.0, MS Internet Explorer:mac 5.2, Mozilla 1.4, Safari 1.0, lynx 2.8.4.

Atutor: Keyboard shortcuts are only available on IE Windows. All other functionality is cross-browser.

Ilias: Use of frames impacts severely on text-only navigation with Lynx. Functionality is cross-browser.

Moodle: Runs correctly under all the browsers tested. HTML WYSIWYG editing functionality depends on JavaScript and DOM extensions only available on IE5+ for Windows and Mozilla 1.3+, therefore is not available on IE:mac, Safari or lynx; these browsers fall back on hand-editing of HTML. All other functionality is cross-browser.

Not dependent on JavaScript

The key functionality of the application must be available to browsers without JavaScript. While JS is widely available, not all implementations are fully compatible or consistent, and it may pose security concerns; in fact JS is often disabled for security reasons. Furthermore, screen readers and other alternative User Agents have no JS support.

ATutor: Runs correctly with JS disabled.

Ilias: Runs correctly with JS disabled.

Moodle: Runs correctly with JS disabled, loses WYSIWYG HTML editor.

No maintenance downtime

LMS applications should not require system downtime for regular clean-up, backups, indexing and backups.

None of the systems reviewed requires downtime for regular maintenance, thanks to their reliance on modern RDBMSs. However, older versions of MySQL (3.x series) may require downtime to perform index analysis for better query optimizations. System upgrades still require service outages, yet this is to be expected of most applications.

Backup

There is a method to backup and restore all the system data that is reliable, documented and scriptable.

MySQL and PostgreSQL provide means of creating a database dump for backup purposes, with wide availability of options. Additionally, MySQL offers mysqlhotcopy, a means to obtain a copy of the on-disk database files. Uploaded files are stored as regular files in the filesystem, and can be backed up and restored as well.

It must be noted that the database backup and restore processes are performed by the system administrator, and cover the whole database. There is no method to restore only individual records or groups of records from a backup.

Moodle offers an additional backup mechanism that allows teachers and administrators to selectively backup and restore content. A teacher can backup a single course at a given time, and restore it later without affecting the rest of the courses. This backup is XML-based, and can serve as a suitable migration tool.

ATutor: Standard DB and files backup.

Ilias: Standard DB and files backup.

Moodle: Standard DB and files backup, plus XML-based backup and restore module.

2.3.15 Installation, dependencies and portability

Ease of installation and maintenance is critical for a scenario of wide deployment, covering both large-scale centrally managed installations and stand-alone installations in the Local Area Network of a tertiary education organization.

Our considerations cover portability, robustness, freedom and overall ease of installation.

Installation, dependencies and portability	ATutor	Ilias	Moodle	Comment
Overall portability	Widely portable	Widely portable on Unix, limited on Win32	Widely portable	
Dependencies are robust	Yes	XML dependencies not robust	Yes	
Dependencies can be replaced	Yes	Some	Yes	XML-related dependencies cannot be replaced in Ilias.
Dependencies are FOSS	Yes	Required	Yes	Some optional dependencies for Ilias require commercial and nonFOSS licenses.
Dependencies are easily installed	Yes	Required	Yes	Some optional dependencies (Java) require separate installation on Debian systems.
System is easily packaged	Yes, packages not available.	Partially. Packages not available.	Yes, Debian packages available	

Overall portability

Overall portability depends on the server, middleware and database software. Open Source software, being distributed in source format, is usually more portable than comparable software distributed in only in binary executable format.

The three LMSs are known to run successfully on most major Unix variants, including BSD variants, Solaris and Mac OS X. As part of this review, they have been tested on Mac O SX v 10.3.3 (Panther).

However, we focus on portability to Win32 platforms, as a significant number of NZ institutions have Win32 servers available. The overall results are very encouraging.

Win32 ports of PHP and MySQL are mature and available in easily installable packages. PHP on Win32 can be used with MS IIS webserver. This is particularly important because Apache2, while available, is not mature, and the legacy version of Apache (1.3.x) is only experimental on Win32.

The PEAR library system and its libraries are also available for Win32 platforms.

Unfortunately PostgreSQL, the alternative RDBMS for Moodle, is not supported on Win32.

ATutor: Widely portable, runs on Win32 platforms as well as a range of Unix platforms.

Ilias: The core application is widely portable, and runs on a . However some libraries are cumbersome to compile on Win32 platforms (libXML2, libXSLT, etc.).

Moodle: Widely portable, runs on Win32 platforms as well as a range of Unix platforms (Linux, Solaris, OpenBSD, NetBSD, FreeBSD), plus Novell Netware. When running on Win32 platforms, RDBMS choice is limited to MySQL.

Dependencies are robust

Supporting application is much easier when dependencies are flexible, and the application is not tied to specific versions of libraries. This is particularly important over time, as libraries and applications are upgraded at different paces.

Obscure, unmaintained, experimental or otherwise deprecated libraries are also considered as risks.

ATutor: Only depends on zlib.

Ilias: Depends on GD (v2.x), zlib, JPEGlib, ImageMagick, libPNG, libXML2, libXSLT, Sablotron and expat, as well as the PEAR packages HTML_Template_IT and Auth. Two of these dependencies – libXML2 and libXSLT – are under very active development and can break backward compatibility.

Moodle: Depends on GD library (version 1.x or 2.x) and zlib, very common libraries.

Dependencies can be replaced

For long-lived applications, a key risk-management strategy is to maintain independence from major framework components. It is desirable that the LMS framework dependencies can be replaced by functional equivalents.

The webserver component is highly portable, thanks to the CGI standard. Viable alternatives include Zeus, iPlanet (formerly Netscape), AOLServer, khttpd, thttpd, and MS IIS on Win32. Apache itself has two different code bases (v 1.3 and 2.0), which can serve as alternative while minimizing impact of migration.

Note that, while PHP applications can run through CGI, performance will suffer, as PHP is run with Apache in a specially compiled module that is significantly faster than CGI. PHP has similar modules/plugins for MS IIS and thttpd.

The RDBMS component in most cases cannot be ported transparently. While the LMSs reviewed use SQL-compliant databases, the SQL standard is notably flexible, and porting costs can be significant.

ATutor and Ilias are tightly coupled to MySQL, and porting to a different RDBMS is likely to be a significant endeavour. Moodle uses a database-independent DB access library, and its SQL schemas and statements work on MySQL, PostgreSQL and, with a bit of work, Oracle, SAPDB and DB2. This makes Moodle not only able to run on more than one RDBMS today, but also simplifies future porting efforts.

ATutor: Can use Apache alternatives. Constrained to MySQL

Ilias: Can use Apache alternatives. Constrained to MySQL.

Moodle: Can use Apache alternatives, RDBMS alternatives: PostgreSQL, Oracle, DB2, SapDB.

Dependencies are FOSS

All the system requirements must be available as Free/Open Source Software (FOSS). We evaluate whether all the software modules are available under OSI-certified[9] licenses.

A review of the system components required by the LMSs indicates that they are all available under OSI-certified licenses, most of the under liberal BSD-style licenses.

Two dependencies for Ilias are not OSI-Certified; note however that both are optional. The SCORM component required a Java JVM. Currently mature JVMsare available from Sun and IBM under licenses that – while free of cost – are not OSI-certified. The user tracking and statistics module uses JpGraph, which is under two different licenses: the QT Free License (QPL) that allows free use only for non-commercial purposes and the JpGraph Professional License that applies when JpGraph is used for commercial purposes.

ATutor: All dependencies are FOSS .

Ilias: All *required* dependencies are FOSS, Java (non-FOSS, free of cost) and JpGraph (non-FOSS, licence fee) are optional.

Moodle: All dependencies are FOSS.

Dependencies are easily installed

All the system requirements must be easy to install on modern unix-based OSs. They should be available at least in source format, with a preference towards availability of binary packages in RPM/DEB formats, ideally included in the major distributions

We evaluated installation of Debian and RedHat-based systems, as well as Mac O SX using Fink packages. Our findings are encouraging, yet it must be noted that packaged versions, in particular those included in release versions of major distributions can lag significantly behind the *latest* versions available in source code.

ATutor: All dependencies available for Debian, RH and Mac O SX systems as part of the standard distribution.

Ilias: All *required* dependencies available for Debian, RH and Mac O SX systems as part of the standard OS distribution or via PEAR. Java is not included in Debian systems, but is an optional dependency.

Moodle: All dependencies available for Debian, RH and Mac O SX systems as part of the standard distribution. The Debian distribution in fact includes

9 OSI-Certified is a trademark of the Open Source Consortium (?), indicating that the license is compliant with the Open Source Definition. See http://opensource.org/

Moodle.

System is easily packaged

All the components that integrate the solution are available as standard-format software packages for the target platforms or are easily packaged. This simplifies administration of the LMS servers both for centrally maintained servers and for small scale locally maintained servers. Additionally, it should be possible to create “Live CDROM” systems based on the software packages. For simplified sampling and local deployment.

Debian's DPKG and RedHat's RPM are the standard software package management systems on the Linux platform; software packages should be available in those formats.

All the dependencies for these systems are available as part of the standard OS. The systems are straightforward to build into a package..

Atutor: There are no binary packages available.

Ilias: There are no binary packages available.

Moodle: Debian binary package is available as part of the main Debian distribution. The package is available both on the stable release, and on the upcoming release, currently in *testing*.

2.4 INTEROPERABILITY

Systems are seldom deployed in a vacuum. Deployment scenarios include integration with pre-existing systems and networks talking a myriad of protocols.

As discussed earlier, integration of authentication systems is by far the most likely scenario, which is explored fully under *Modular Authentication*. Integration efforts other than authentication also benefit from a clean, modular, architecture, discussed under *System is modular and extendible*,

2.4.1 Integration is straightforward

Ease on integration depends as well on the availability of libraries that implement common protocols. This is actually a feature of the framework, which the three systems share.

The three layers of the stack that comprises the framework have a very good availability of protocol libraries. The PHP library repository (PEAR), plus libraries and modules available outside of PEAR, have a wide spectrum of modules implementing internet-related protocols.

Additionally, there is a small range of Apache modules that implement common protocols, such as LDAP, Kerberos and RADIUS. Finally, there are hardly any modern protocols without an available (and open) implementation on Linux.

Put together, the Linux-Apache-PHP framework provides one of the most comprehensive development platforms available.

2.4.2 LMS standards support

The system is compliant with common standards for LO creation, retrieval and use – i.e. SCORM 1.2, IMS, DC metadata.

LMS standards support	ATutor	Ilias	Moodle	Comment
LO Standards	IMS export	IMS import	No.	There is no support for SCORM runtime environment. Moodle has an IMS/SCORM module in development.
Search and Retrieval standards	No. Future plans mention TILE.	No	No	

LO standards (IMS packaging, SCORM)

The LMSs are rated on their ability to use externally supplied Learning Objects packaged in IMS format (also known as SCORM 1.2), their ability to provide SCORM 1.3 runtime environment support for LOs that are SCORM-aware. Additionally, the ability to export content in IMS packages is evaluated.

ATutor: Can export course material as an IMS package. Does not support use of externally generated IMS packages or SCORM 1.3. These features are in the immediate development plans.

Ilias: Can use externally generated IMS packages; however this feature is in development, and we could not use it successfully for this review. There is no SCORM 1.3 runtime environment support.

Moodle: Does not support IMS packages or SCORM 1.3. These features are in the immediate development plans.

Search and retrieval standards (LORAX, TILE)

Use of LOs within the system is made easier by integrated search and retrieval of LOs from repositories such as the Learning Federation's Learning Exchange (LEX) and the TILE servers.

The TILE protocol is still under development. LEX is accessed using the LORAX protocol (http://www.thelearningfederation.edu.au/repo/cms2/tlf/published/8519/docs/SOAP_Specification_V0_4.pdf).

Unfortunately, none of the systems implement these protocols. ATutor includes TILE support in its *future plans* document.

3. COST OF OWNERSHIP

As part of this review exercise, the three systems have been deployed and customised, and bugs identified have been fixed whenever possible. These steps provide a factual background against which to assess the systems on a metric that is very hard to measure reliably.

Architecture, modularity, code quality and community support all factor in the long-term costs. This section summarises partially the findings in those specific areas.

Cost of ownership	ATutor	Ilias	Moodle	Comment
Implementation	Low cost, automated	Mostly automated	Low cost, automated	Ilias required manual configuration of each host. Others are automated.
Development and support	Unclear, complex	Unclear, complex	Straightforward, well supported	
Hardware and licensing	No licensing costs, reasonable HW costs	May have licensing costs, reasonable HW costs.	No licensing costs, reasonable HW costs	
Maintenance	Low cost, automated	Mostly automated	Low cost, automated	

3.1 IMPLEMENTATION

The cost, ease and timeframe for implementation is acceptable. We evaluate both single-hosting deployments and multi-hosting deployments, based on the off-the-shelf status of the applications.

Moodle: Low cost of deployment for single and multi-hosting environments. Can be used out-of-the-box.

Atutor: Low cost of deployment for single and multi-hosting environments. Can be used out-of-the-box.

Ilias: Medium cost of deployment for single and multi-hosting environments. The software is not available for immediate deployment, as it is still going through a beta cycle. Additionally, roles need configuration; is not usable out of the box.

3.2 DEVELOPMENT AND SUPPORT

Applications are rated on their cost of development and support. This covers availability of expertise and tools on server platform, ease of development, quality of documentation, cleanness of code, robustness/extent of bugs, use of high level languages and web-targeted middleware.

Additionally, effective experiences working with the application structure and code are reported.

The three systems share a framework/server platform on which tools, expertise and ease of development are outstanding.

Atutor: The code is straightforward, but lacks structure. It is unclear where should new functionality be added.

Ilias: The code is well structured, but the architecture is confusing and lacks documentation. Tracing and fixing bugs proved daunting, even in well-commented and structured code.

Moodle: The architecture is straightforward and documented, and finding and fixing bugs proved to be easy. It is easy to follow the logic, additional modules can be created based on the existing ones.

3.3 HARDWARE AND LICENSING

Hardware and licensing requirements for the systems must be of reasonable cost, and allow for cost-effective deployments of small and large scale. Further analysis of the hardware and licensing requirements can be found under *Scalability* and *Dependencies* subjects.

Atutor: All licenses required are FOSS. Hardware requirements are reasonable. .

Ilias: All licenses required are FOSS, some optional components are available under commercial licenses. Hardware requirements are reasonable.

Moodle: All licenses required are FOSS. Hardware requirements are reasonable.

3.4 MAINTENANCE

Long-term maintenance of installation is a major cost-centre. Maintenance should be straightforward and easily automated. Dependence on administration interfaces other than configuration files is discouraged – although their availability is considered a plus.

Additionally, good logging of warnings and availability of debugging flags are also considered.

Atutor: Easily managed configuration files. More logging available through Apache and PHP. Logs contain warnings during normal operation.

Ilias: Configuration files are managed via web-interface – hard to automate. Logging is clean, more logging available through Apache and PHP. Logs contain warnings during normal operation.

Moodle: Easily managed configuration files. Logging is clean, more logging available through Apache and PHP.

4. STRENGTH OF THE COMMUNITY

Dynamic development communities are a widely recognized feature of successful open source projects. As analysed by Eric Raymond[10], such communities are non-trivial to develop, yet they are a key component of a software project health and longevity.

Cost of Ownership	ATutor	Ilias	Moodle
Installed base, longevity	Small-medium installed base. Since 2002.	Small-medium installed base. Since 2001.	Medium (>1000) installed base. Since 2002.
Documentation	Good end-user documentation	Limited end-user and developer documentation	Good end-user and developer documentation
End-user community	Medium-small community. Active.	Medium-small community, split in German and English. Active.	Strong medium-size community.
Developer community	None	Small, limited	Small, thriving

4.1 INSTALLED BASE AND LONGEVITY

The number of installations is large, with "quality" organizations using the system. We also consider initial release date if available.

Atutor: Originally released in December 2002, early versions date back to January 2002. Known installations list only reports 18 servers, the actual installed based is likely to be much larger, at least by an order of magnitude.

Ilias: Originally released in July 2001, early versions date back to mid-2000. Known installations list reports over 100 servers, most of them at large universities.

Moodle: Originally released in August 2002, early versions date back to November 2001. Recently reported to have over 1000 known installations, several of them in major universities.

4.2 DOCUMENTATION

The system has high quality documentation and online help. Documentation should be available targeted towards end-users (teacher, student, author, administrator), systems administrators and programmers.

Atutor: Good end-user documentation.

Ilias: Limited end-user documentation. Some developer documentation.

Moodle: Has good end-user, administrator and developer documentation.

10Homesteading the Noosphere http://www.catb.org/~esr/writings/cathedral-bazaar/homesteading/

4.3 END-USER COMMUNITY

Responsiveness of the support community is good. Historical numbers of the user and developer mailing lists/forums are strong and show growth. Spirit in the mailing lists/forums is positive.

Unfortunately, the communities are centred around web-based forums instead of the traditional Usenet or mailing lists. This makes measuring community size by traffic nearly impossible without access to the forum database.

Atutor: Medium-sized user community, helpful and active.

Ilias: Medium-sized user community, helpful and active.

Moodle: Strong user community, helpful and active.

4.4 DEVELOPER COMMUNITY

There is an active developers community, helpful and with a good understanding of the application internals. There is one or more maintainers with CVS access or equivalent. There are many contributors credited in the Changelog.

Atutor: There is no developer community. Currently the development of Atutor is closed, and this has prevented a community from forming.

Ilias: There is a small but active developer community.

Moodle: There is an active developer community. Several developers have CVS access, and there is a well populated credits list.

4.5 OPEN DEVELOPMENT PROCESS

The development process is open, with access to the version control repository, nightly releases, a bug tracker, more than one core developers listed in the Changelog.

Atutor: Closed development process, only offers access to nightly builds, and a bug forum.

Ilias: Very open development process, with access to CVS, bug tracker and other tools.

Moodle: Very open development process, with access to CVS, bug tracker and other tools.

4.6 COMMERCIAL SUPPORT COMMUNITY

It is desirable that the application has a community of commercial companies offering related services and support.

Atutor: Only the original development company is offering Atutor-related services.

Ilias: Has not yet gained critical mass for related commercial offerings.

Moodle: Moodle.com, some ISPs and independent contractors offer Moodle-

related services.

5. LICENSING

To guarantee the future of the project, and of future investments in the project, the system must be truly Open Source. The license must be OSI-Certified, the terms of usage and redistribution well understood and acceptable. The community stakeholders must understand the terms, and act openly and accordingly.

The three systems considered are covered by the GNU General Public License, one of the founding Open Source licenses, and OSI-Certified.

6. INTERNATIONALISATION AND LOCALISATION

Some programs assume text is written in English (ASCII). For people who use non-English languages, these programs are barely usable. And more, though many systems can handle not only ASCII but also ISO-8859-1, some of them cannot handle multi byte characters for CJK (Chinese, Japanese, and Korean) languages, or combined characters for Thai.

We evaluate support for multiple languages, with a focus on Maori and Pacific languages at the data handling level and user interface levels.

Internationalisation is needed in the following places[11]:

- Displaying characters for the users' native languages.
- Inputting characters for the users' native languages.
- Handling storage and retrieval of data and files written in popular encodings that are used for the users' native languages.
- Using characters from the users' native languages for file names and other items.
- Printing out characters from the users' native languages.
- Displaying messages by the program in the users' native languages.
- Formatting input and output of numbers, dates, money, etc., in a way that obeys customs of the users' native cultures.
- Classifying and sorting characters, in a way that obey customs of the users' native cultures.
- Using typesetting and hyphenation rules appropriate for the users' native languages.

This review puts emphasis on the first three items, which are the base of internationalisation support.

The following terminology is used:

Internationalisation means modification of software or related technologies so that software can potentially handle multiple languages, customs, and so on in the world.

Localisation means implementation of a specific language for an already Internationalised software.

Internationalisation	**ATutor**	**Ilias**	**Moodle**

11 Internationalisation and Localisation requirements are inspired by Introduction to I18N, a document maintained by the Debian Project.

Internationalisation	ATutor	Ilias	Moodle
Localisable UI	Limited to text, no unicode.	Limited to text, unicode	Text and date/currency formats. Unicode. BiDi support.
Localised to relevant languages	12 lang	10 lang	31 lang
Unicode text editing and storage	No	Yes	Yes
Timezones, date localisation	No	No	Yes
Alternative language content	No	Yes	No

6.1 LOCALISABLE UI

Strings (texts) belonging to the UI are managed through an Internationalisation library, such as GNU Gettext, that separates UI strings from the code, and support locales for date and number formats. Must be Unicode compliant and support easy Localisation (tools such as those supplied with gettext are a plus). It is desirable that the UI supports BiDi text (for right-to-left languages).

None of the systems used standard libraries, preferring instead to roll their own. These solutions have different tradeoffs between memory consumption, execution speed and ease of maintenance.

Not all the systems support date and number format locales.

ATutor: Straightforward localisation scheme using a list of Localised strings held in the database (slow execution, hard to maintain, less memory consumption). Does not seem to support Unicode, it only to supports the ISO-8859 code page. Does not support BiDi text or format locales.

Ilias: Straightforward localisation scheme, using a list of Localised strings stored in specially formatted files that are (slow execution, easy to maintain, low memory consumption) and a locale setting. Supports Unicode – all Localisations are using UTF-8. Does not seem to support BiDi text.

Moodle: Straightforward localisation scheme, using a list of Localised strings stored in PHP-formatted files (fast execution, easy to maintain at the cost of more memory consumption) and a locale setting. Supports Unicode– although most Localisations are using legacy codepages– and BiDi text.

6.2 LOCALISED TO RELEVANT LANGUAGES

It is expected that LMSs have already existing locale and strings for English. It is desirable that they have a wider range of already implemented Localisations using UTF-8. However unlikely, it is desirable that they have existing Localisations to pacific languages.

ATutor: 12 languages available, none uses UTF-8.

Ilias: 10 languages available, all in UTF-8, with the Simplified Chinese

Localisation making use of high characters.

Moodle: 31 languages available, plus seven national Localisations. The Farsi language uses UTF-8 and is left-to-right.

6.3 UNICODE TEXT EDITING AND STORAGE

Where the system allows content editing, it must support editing of wide Unicode characters, with a focus on macronised vowels (for Maori support).

- Data storage must support Unicode.
- Editing interfaces (forms for forums, etc.) must support Unicode.
- It must be feasible to add extra support for entry of macronised characters.

ATutor: Only supports ISO 8856 posts, some browsers (Mozilla, for instance) work around this limitation HTML-encoding high characters. Other browsers display but not post outside of ISO 8856 characters.

Ilias: Uses UTF-8 as charset, application-wide. This makes support for high characters transparent, at the expense of legacy browsers.

Moodle: With the default ISO-8856, some browsers (Mozilla, for instance) work around this limitation HTML-encoding high characters. Other browsers display but not post outside of ISO 8856 characters. Setting the locale to use UTF-8 and altering the templates charset setting provides consistent Unicode support.

6.4 TIME ZONES AND DATE LOCALISATION

All dates must be stored with an unambiguous time zone (TZ), preferably UTC. This is to support scenarios where users are in different time zones.

ATutor: Supports only one TZ.

Ilias: Supports only one TZ.

Moodle: Stores time/dates in UTC, and recognizes in which TZ the server and each user is.

6.5 ALTERNATIVE LANGUAGE CONTENT

The system should allow for alternative language versions of the same resource.

Content negotiation support: The system should allow the user to choose directly (or indirectly via browser settings) to access alternative language versions for resources that have them.

ATutor: No support.

Ilias: Supports multiple languages for some content types, doesn't support content negotiation.

Moodle: No support.

7. ACCESSIBILITY

Not a complete evaluation on the overall accessibility of the systems, this review focuses on the availability of accessibility hooks in the existing implementation. Provided the appropriate hooks, particular accessibility problems can be worked upon.

Accesibility	ATutor	Ilias	Moodle	Comment
Text-only navigation	Excellent	No	Average	
Scalable fonts/graphics	Yes	No	Partial	

7.1 TEXT-ONLY NAVIGATION SUPPORT

The system supports text only navigation and other accessibility hooks,such as:

- Hidden links
- Link shortcuts
- Descriptive link texts
- Full support for ALTs for images and rich media.

Atutor: Excellent text-only navigation, including link shortcuts, ALT texts, etc.

Ilias: Cannot be used with text-only navigation, due to frames.

Moodle: Limited or no support for text-only navigation. Images lack ALT texts, navigation does not provide hidden links nor keyboard shortcuts. Some links use ambiguous link texts.

7.2 SCALABLE FONTS AND GRAPHICS

Graphics and text scale correctly when the user requests large fonts on the browser.

Atutor: Text and images are scalable.

Ilias: Text is scalable, graphics are fixed-size. Due to use of frames, the interface is not usable with large fonts.

Moodle: Text is scalable, graphics are fixed-size.

8. DOCUMENT TRANSFORMATION

System should support processes for single source documents creation and multiple output transformation, with a primary goal of authoring content that can be delivered in two specific output formats: web (HTML) and print (PDF).

It is desirable that the authoring facility works with a standard format, such as DocBook.

Atutor: There is no support for format transformations. Content authoring is in pseudo HTML. Content packaging is available, which may help the process.

Ilias: Latest version can perform format transformations from XHTML content to PDF using HTMLDoc. Authoring is in XHTML, but very limited.

Moodle: There is no support for format transformations. Content authoring is in HTML, text and wiki formats.

Bibliography

BASS2003: Bass, Clements, Kazman, Software Architecture in Practice (2nd edition), 2003
ANSI1471: , 6ANSI/IEEE Std 1471, Recommended Practice for Architectural Description of Software-Intensive Systems, 2000
MCBREEN2002: McBreen, Pete, Software Craftmanship- The New Imperative, 2002
GARF2003: Garfinkel, Spafford, Schwartz, Practical Unix & Internet Security, 3rd Edition, 2003
GARF2001: Garfinkel, Spafford, Web Security, Privacy & Commerce, 2001