7th International Conference on DEVELOPMENT AND APPLICATION SYSTEMS
S u c e a v a, R o m a n i a, M a y 27 – 29, 2 0 0 4

THE IMPLEMENTATION OF AN OPEN SOURCE SYSTEM FOR THE CREATION OF E-LEARNING CONTENT WITHIN AN IRISH UNIVERSITY

Kevin JOHNSON[1], Dorel PICOVICI[2], Timothy HALL[3]
University of Limerick,
Limerick, Ireland
[1)] Kevin.Johnson@ul.ie, [2)] Dorel.Picovici@ul.ie, [3)] Timothy.Hall@ul.ie

Abstract. *Open source software is becoming the most interesting 'new' phenomenon of the entire information technology area, generating a level of interest similar to that of the first moments of the Internet. The impact of open source technology is expected to be quite noticeable in the software industry, and in society as a whole. It allows for novel development models, which have already been demonstrated to be especially well suited to efficiently take advantage of the work of developers spread across all corners of the planet. It also enables completely new business models, which are shaping a network of groups and companies based on open source software development. This paper will look at work carried out on a project to create an effective authoring tool for use within an Irish Higher Education Institution that would allow for the creation and publication of electronic content for modules run within the University.*

Keywords: *Open Source technology, publishing content, authoring tool, Higher Education, dynamic content.*

Introduction

In 1984, Richard Stallman, a researcher at the MIT AI lab, started the GNU project. The GNU projects goal was, simply put, to make it so that no one would ever have to pay for software. Stallman launched the GNU project because essentially he felt that the knowledge that constitutes a running program – what the industry calls the source code – should be free. Where the proprietary commercial software vendors saw an industry guarding trade secrets that was tightly protected, Stallman saw scientific knowledge that must be shared and distributed. The basic tenet of the GNU project and the Free Software Foundation (the umbrella organisation for the GNU project) is that source code is fundamental to the furthering of computer science and freely available source code is truly necessary for innovation to continue (O'Reilly 1999).

This basic philosophy for the creation and distribution of software in the higher education realm has interesting parallels. Scientific knowledge is often in the public domain; it is one function of academic publishing to put it there.

With software, however, it was clear that just letting the source code go in to the public domain would tempt businesses to co-opt the code for their own profitability. Stallman's answer to this thread was the GNU General Public License, known as the GPL. The GPL says that you may copy and distribute the software licensed under the GPL at will, provided you do not inhibit others from doing the same, either by charging them for the software itself or by restricting them through further licensing. The GPL also requires works derived from work licensed under the GPL to be licensed under the GPL as well.

In the spring of 1997 a group of leaders in the free software community assembled in California. Their concern was to find a way to promote the ideas surrounding free software to people who had formerly shunned the concept. They were concerned that the free software Foundation's anti-business message was keeping the world at large from really appreciating the power of free software. The group agreed that what they lacked in large part was a marketing campaign, a campaign devised to win mind share, not just market share. Out of this discussion came a new term to describe the software that qualified as Open Source.

When we talk, in English, about 'free software', there is a dangerous ambiguity, due to 'free' meaning both 'freedom' and 'gratis'. Therefore, in this document, we will use mainly the term 'open source' when referring to users freedom of use, redistribution, etc., and 'gratis software' when referring to zero acquisition cost. Before going into more detail, it is a good idea to state clearly that open source software does not have to be gratis – in the sense of cost nothing money wise. Even more, it usually is not, or at least, not completely.

The main features that characterise free (open source) software is the freedom that users have to:

- Use the software as they wish, for whatever they wish, on as many computers as they wish, in any technically appropriate situation.
- Have the software at their disposal to fit it to their needs. This includes improving it, fixing its bugs, augmenting its functionality, and studying its operation.
- Redistribute the software to other users, who could themselves use it according to their own needs. This redistribution can be done for free, or at a charge, not fixed beforehand

It is important now to make clear that we are talking about freedom, and not obligation. That is, users of an open source program can modify it, if they feel it is appropriate. But in any case, they are not forced to do so. In the same way, they can redistribute it, but in general, they are not forced to do so.

To satisfy those previous conditions, there is a fourth condition that is basic, and is necessarily derived from them. This is that users of a piece of software must have access to its source code.

The source code of a program, usually written in a high level programming language, is absolutely necessary to be able to understand its functionality, to modify it and to improve it. If programmers have access to the source code of a program, they can study it, get knowledge of all its details, and work with it as the original author would.

Paradoxically, if this freedom is to be guaranteed for a given piece of software, with current legislation, it is necessary to "protect" it with a license, which imposes certain restrictions on the way that it can be used and distributed. This fact causes some controversy in certain circles, because it is considered that these licenses make the software distributed under them "less free". Another view, more pragmatic, is that software will be made "more free" by guaranteeing the perpetuation of these freedoms for all its users. Because of that, people holding this view maintain that it is necessary to limit the ways of use and distribution. Depending on the ideas and goals of the authors of a piece of code, they can decide to protect it with several different licenses.

Open Source For Higher Education

Advantages of Open Source in Higher Education

Motivations for using and developing open source software are mixed, ranging from philosophical and ethical reasons to pure practical issues. In this section, some of the most widely proposed practical advantages will be introduced.

Usually, the first perceived advantage of open source models is the fact that open source software is made available gratis or at a low cost. But this characteristic is not exclusive to open source software, and several proprietary software products are made available in similar ways (a prominent case could be Microsoft's Internet Explorer)(Libre 2000). What really distinguishes open source software from software available without fee is the combination of effects due to the characteristics listed above in the introduction.

All of them combined produce a synergistic impact that is the cause of the real advantages of the open source model. Here are some more details on how these characteristics turn into advantages:

- *The availability of the source code and the right to modify it* is very important. It enables the unlimited tuning and improvement of a software product. It also makes it possible to port the code to new hardware, to adapt it to changing conditions, and to reach a detailed understanding of how the system works. This is why many experts are reaching the conclusion that to really extend the lifetime of an application, it must be available in source form. In fact, no binary-only application more than 10 years old now survives in unmodified form, while several open source software systems from the 1980s are still in widespread use (although in many cases conveniently adapted to new environments). Source code availability also makes it much easier to isolate bugs, and (for a programmer) to fix them.

- *The right to redistribute modifications and improvements to the code,* and to reuse other open source code, permits all the advantages due to the modifiability of the software to be shared by large communities. This is usually the point that differentiates open source software licenses from "nearly free" ones. In substance, the fact that redistribution rights cannot be revoked, and that they are universal, is what attracts a substantial crowd of developers to work around open source software projects.
- *The right to use the software in any way.* This, combined with redistribution rights, ensures (if the software is useful enough), a large population of users, which helps in turn to build up a market for support and customization of the software, which can only attract more and more developers to work in the project. This in turn helps to improve the quality of the product, and to improve its functionality. Which, once more, will cause more and more users to give the product a try, and probably to use it regularly.

Perceived disadvantages of Open Source

Open source development models also lead to the perception of some disadvantages. However, some of them are only disadvantages if we are stick to classical (proprietary) development models, which is of course not the case with open source. Listed below are some of these disadvantages:

- *There is no guarantee that development will happen.* In other words: it is not possible to know if a project will ever reach a usable stage, and even if it reaches it, it may die later if there is not enough interest. Of course, this is also a problem with proprietary software, but it is more evident in the case of open source. Especially when a project is started without strong backing from one or more companies, there is a significant initial gap, when the source base is still immature and the development base is still being built. If it is not possible to get funding or enough programmers cooperating at this stage, the project just "dies", or perhaps slowly fades out. Usually, when it reaches a self-sustaining level, the user and development base is such that it can proceed by itself, without other external incentives.
- *There may be significant problems connected to intellectual property.* This point is especially important, now that some countries are accepting software and algorithm patents. It is very difficult to know if some particular method to solve a software problem is patented, and so the community can be considered guilty of intellectual property infringement. Developers consider source code not as an executable device, but a mere description of how a device (the computer) executes, and therefore uphold the idea that source code is not by itself (in absence of an executableprogram) covered by patent law even in countries where software patents are accepted. In any case, it still leaves problems for the users, who need the executable programs. Although the issue of software patents is a problem for

the whole software industry, open source is probably one of the more clear cases where it can be shown how they harm the regular process of software development. The specific problems are that availability of source code simplifies the detection of patent infringements by patent holders, and that the absence of a company that holds all the rights on the software also makes it difficult to use the mechanisms in use by companies to defend from patent litigation, like cross-licensing or payment of royalties.

- *It is sometimes difficult to know that a project exist, and its current status.* There is not much advertising for open source software, especially for those projects not directly backed by a company willing to invest resources in marketing campaigns. However, some people see this fact as a market opportunity, and several companies with experience in Internet based information services are approaching open source software with added value services which maintain information useful for people or companies trying to locate or evaluate open source software of some given characteristics.

It is extremely important to 'see' through the various interpretations of the advantages and disadvantages of open source, and if possible try to analyze with quantitative methods if open source can be helpful in a given situation, or for a given user or company.
Based on this research carried out on Open Source technologies, it was agreed that the tool would be implemented on an open source platform like Linux and using open source technologies like PHP and Mysql. More details on this follow in the subsequent sections. Aims And Objectives
The application was being designed with the following aims in mind:

- *Flexibility* – allow access any time, any where
- *Adaptability* – caters for the diverse and changing needs of Universities
- *Customisable* - allow custom content to be added by the different colleges within the University
- *Expandable* – permit the tool to be expandable to suit the changing need of the authors and students
- *Updatable* – allow for the easy updatability of content through the tools interfaces
- *Standards Compliant* – compliant with leading standard bodies AICC/IEEE/IMS etc
- *Low cost* – is less expensive than competing services

Once all of these objectives were known, the next step was to design the tool around the people who were going to be using it – the users.

System Users

Research was carried out on the additional tools and systems that are available in today's ever changing market. The functional specifications and features associated with these systems were reviewed and feedback was given to the design team. It was agreed that out tool would be viewed from three different viewpoints, that of the user, the author and the administrator.
The user was the intended end user for the tool as so it was important that their viewpoint was captured and feedback in to the development process of the tool. This allowed us to create a tool that suited the user and catered for their needs. Some of the functionality associated with the learner included:

- Log in/Log out of the tool
- Search Capabilities
- Take a new course
- Continue an existing course

The author on the system was viewed as being a lecturer or teacher of sorts. This author would have the following features available to them (Johnson 2002):

- Log in/ Log out

- Create Content
  1. Fragment – smallest possible level of granularity
  2. Topic
  3. Lesson
  4. Module
  5. Course
  6. Curriculum – A group of courses
- Publish Content
  1. Topic
  2. Lesson
  3. Module
  4. Course
  5. Curriculum – A group of courses
- Search Databases
- Manage Content Creation
- Management of publishing factors
- Management of users
- Tutorials/Help sections

The administrator is the final user on the system. Some of the tasks associated with this user included:

- Set privileges - allow only certain users to view content
- Maintain data base(s)
- Assessment/Report generation - for groups /courses/ individuals
- Track all other users for commercial and security purposes.
- Use the content as learner's

System Architecture

The Authoring system, and resulting tool, proposed by this paper essentially provides non-technical authors the means to create a series of topics, lessons or modules of learning, from low granularity learning objects and combine them to be published into a new structure. The author has the option to create basic textual learning objects and import learning objects created by other third party applications, such as word documents, Acrobat files, PowerPoint presentations, images, macromedia flash objects, JavaScript, HTML files etc. This group terms these low levels of granularity learning objects as "fragments". Once this content is uploaded, the file or content is stored in a specific location on the server (Phoenix 2003). The exact location is dependant on who is uploading the piece of content. The associated content has information or metadata associated with it and stored in a database for quick search and retrieval functions. It is worth noting that this tool will allow some additional new content to be created that is non-reusable, such as course summaries, aims and objectives etc that are specific to the aggregation of content (Concannon and Johnson 2003).
Following the population of a fragment learning object database, the authoring tool will allow the sequencing of these learning objects into new aggregations, compliant and implemented via the IMS Content Packaging (CP) Specification, as also outlined in ADL's (Advanced Distributed Learning) SCORM (Shareable Content Object Reference Model) suite (ADLNet). The author is presented with all of the available content that is uploaded and not copyrighted. The author selects the content that they wish to use and proceeds to sequence this in a format that is suitable for the user to view. This information is then stored in an XML (eXtensible Markup Language) file on the system. This file is compliant with the IMS CP Specifications. The author will have the option to build new topics, lessons, modules, courses and programs from imported learning object fragments. Therefore the fragment resources will be available to the authoring tool along with the relevant metadata. The author can create high-level maps of learning episodes, indicating a best path, or sequence through a collection of learning material. This system was implemented as the first draft of the tool and used within the University for a testing period of 3 months. The feedback received from this testing enabled us to modify and update the tool for its next evolution.

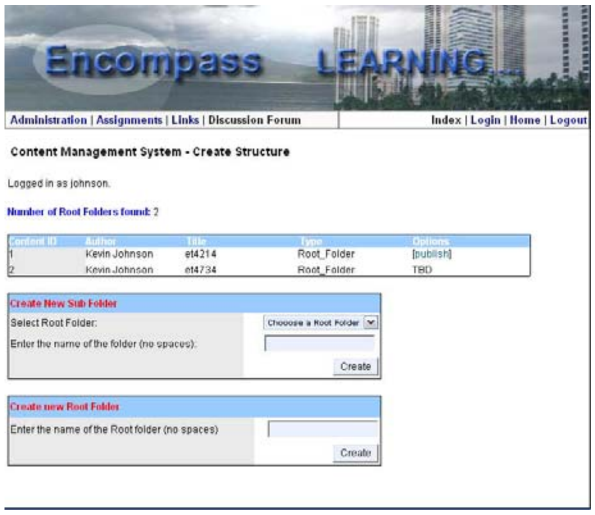
Phase Two

The second phase of the tool build upon what was initially coded and released at the start. One of the main drawbacks that the users and authors found pertaining to the tool was that it was limited in the assignment upload area. This information was relayed to the development team and changes were drafted and implemented with a couple of weeks. The next version of the tool would allow the students to see the assignment deadline approaching (based on a Unix timestamp command) and therefore know instinctively that the assignment needed to be uploaded soon. Upon completion of uploading an assignment, the students can view the directory listing and see the uploaded file, verify it is the right size as the original on the local machine and if they are not happy with this they can delete the file and upload again. The information is stored in a database as well to state that the student has uploaded the assignment. Once they have uploaded an assignment a teaching assistant or lecturer for the module, namely someone with author or admin privileges, can view the uploaded assignments and grade them. The final stage of this process is that the author gives the students some private form of feedback for their personal viewing and some general form for all the other students in the forum to see. This modification worked well and the students were happy with it and the author was pleased that they could now grade the assignments as they are uploaded and give the students feedback also. This feedback will prove useful to next year's students when they are reviewing the uploaded content and associated feedbacks. For the final stage of the tool, authors wanted to be able to dynamically create folder structures on the server from the tool or graphic user interface console. They wanted to be able to create their own structure to the course as opposed to sticking to the default layout. Hence the need for another revision of the source code.

Phase Three

This stage allowed the author to create a new folder structure for the content that was to be uploaded and tagged with metadata content. The layout of the site was now in the hands of the author. They could create folders and sub folders at their discretion. A new feature was added that allowed the author specify the type of content that was being uploaded to the server – namely lecture notes, lab solutions, lab problems or Exam papers. This, in turn, allowed for a more refined publishing system to be installed and permit the author to publish only certain content from each immediate category. This stage is still in testing at the moment.

One advantage of the system is its Open Source software. The system is designed and built on a Linux environment running an apache web server configured to run PHP (PHP Hypertext Preprocessor) (PHP) and MySQL (open source database system) (MySql). One of the main factors for the small turnaround time between phases was this open source technology. There was no delay in coding new functions or working on existing functions. The system was available to the team and easy to modify and expand to meet the new needs of the users. The Linux environment and PHP worked flawlessly together and PHP had a lot of the features built in to deal with the database functions as well as the file management scenarios that arose in phase three.


Encompass LEARNING...
Administration | Assignments | Links | Discussion Forum
Index | Login | Home | Logout

Content Management System - Create Structure

Logged in as johnson.

Number of Root Folders found: 2

Content ID	Author	Title	Type	Options
1	Kevin Johnson	et4214	Root_Folder	[publish]
2	Kevin Johnson	et4734	Root_Folder	TBD

Create New Sub Folder

Select Root Folder: Choose a Root Folder

Enter the name of the folder (no spaces):

Create

Create new Root Folder

Enter the name of the Root folder (no spaces)

Create

Figure 1. Uploading Process on the Authoring Tool

Conclusions

The authoring tool was created to meet the needs of the university system here, within the Department of Electronic and Computer Engineering in the University of Limerick, Ireland. The tool was rolled out and reviewed and tested, allowing for updating and expansion when the need arose. The tool permitted the publishing of content online for students within the courses and also allowed the user to give feedback to the students based on their work and progress within the class.

One of the strikingly attractive features of the tool was that it was based on an open source infrastructure – from the technology used to hose the website to the code that was written to deliver it. This tool, while still being a long way from the likes of commercial systems like WebCT and Blackboard, still has it advantages and, from one point of view, it may be only a matter of time before it catches up.

Acknowledgements

The work presented in this paper form part of a major project, called ENCOMPASS, which has been funded by Enterprise Ireland's Advanced Research Technology Program under the National Development Plan. See the website: http://encompass.amt.ul.ie.

References:

[1] ADLNet "*Advanced Distributed Learning Network (ADL Net)* Advanced Distributed Learning, SCORM Past."

[2] Concannon, F. and K. Johnson (2003). *Designing an Authoring System to Support Learning Object Repositories for Small to Medium Sized Enterprises*. Elearn 2003, Phoenix Arizona.

[3] Johnson, F. C. a. K. (2002). "*Technical Specification for the Encompass Tool - Draft 2*."

[4] Libre, W. g. o. L. S.-. (2000). Free Software/ Open Source: Information Society Opportunities for Europe ?

[5] MySql "MySql Database available at http://www.mysql.com."

[6] O'Reilly, T. (1999). *Open Sources Voices from the Open Source Revolution*, O Reilly and Associates.

[7] Phoenix (2003) "Phoenix Web site Available at http://phoenix.ul.ie/."

[8] PHP "PHP Website available at http://www.php.net/."