

Developing, Deploying, Using and Evaluating an Open Source Learning Management System

Heinz Lothar Grob, Frank Bensberg, Blasius Lofi Dewanto
Department of Information Systems, University of Muenster

Leonardo Campus 3, D-48149 Muenster, Germany

<http://www.wi.uni-muenster.de/aw>

grob@uni-muenster.de, bensberg@uni-muenster.de, dewanto@uni-muenster.de

Abstract. *Every modern institution involved in higher education needs a Learning Management System (LMS) to handle learning and teaching processes. It is necessary to offer e. g. electronic lecture materials to the students for download via the internet. In some educational contexts, it is also necessary to offer internet tutorials to be able to give the students more personal support and accompany them through the whole lecture period. Many organisations have introduced commercial LMS and gained the experience that monolithic solutions do not fulfil the dynamic requirements of complex educational institutions and are very cost-intensive. Therefore, many universities face the decision to stick to their commercial LMS or to switch to a potentially more cost-effective and flexible solution, for instance by adopting available Open Source LMS. Since we have made profound experience in developing and operating an Open Source LMS, this contribution enlightens the main characteristics of this alternative. This paper describes a use case dealing with a full product lifecycle (development, deployment, use and evaluation) of an Open Source LMS at the University of Muenster (Germany). It identifies relevant instruments and aspects of system design which software architects in practical application domains should pay attention to.*

Keywords. Java, Java 2 Enterprise Edition (J2EE), Enterprise Java Bean (EJB), Open Source Software (OSS), Component-based Software Engineering, e-Learning, Learning Management System (LMS), Open University Support System (OpenUSS), CampusSource.

1. Introduction

In the beginning of the year 2000, we were facing a dilemma concerning our LMS [1]. The problem was that we had a LMS implementation but it was based on proprietary technologies.

Should we refactor and extend our proprietary LMS? Or should we redevelop the LMS, so that it will be more open and easy to extend? Another approach would be just to buy a commercial LMS. There were a lot of good commercial LMS on the market at that time. Still today we can find big vendors offering LMS like Blackboard and WebCT¹. Most vendors offer good LMS, which are functional complete and in case that the system does not match the specific needs of the institution, the vendors can extend the functionalities with the support of their consultants. The main problem of this approach is the financial aspect. Almost all LMS vendors ask for an expensive site license, which has to be renewed each year. For our department alone it is thus impossible to follow this approach. It would be feasible if the *whole* university would buy a commercial LMS. However the buying process will be very complex because of the number of different requirements. Learning processes differ significantly between faculties like economics, history and art. Besides, at that time the real value of a LMS for our university as a whole was vague.

As the facts above we decided to rebuild the LMS from scratch, because we also realised that e-learning will be sooner or later a strategic application for our institution. Therefore, the *Open University Support System (OpenUSS)* was born to offer a robust LMS which should be able to cope with the increasing internal demand for e-learning services, first at our department and later at the whole university (*bottom-up* introduction strategy), with over 40,000 students. Following requirements were defined:

- According to our experience, a critical success factor of a LMS is flexibility to meet evolving standards and new conceptual re-

¹ Blackboard and WebCT can be found at <http://www.blackboard.com> and <http://www.webct.com> respectively.

quirements. In order to guarantee the modifiability of the source code, we decided to release the product under the GNU General Public License (GPL). This legal framework ensures an open development process, such that other educational institutions are entitled to use and modify our code.

- In order to develop a scalable, extendable and maintainable product, we decided to take advantage of multi-tier architecture. We also used component-based software engineering approach [2]. In particular, we have chosen Java and J2EE (Servlet and EJB) as the foundation technology [3]. This was a critical decision, since J2EE was not widely adopted in the year 2000. Java itself was a good choice. On the one hand, there were already a lot of java libraries available at that time. On the other hand, Java was gaining momentum to become standard language in computer science education.
- In order to achieve economies of scale, OpenUSS should support the Application Service Provider (ASP) model [4]. It should be able to manage any number of institutions e.g. universities, schools and companies, within one instance.

2. Project specific benefits

Since the year 2000, we have implemented OpenUSS and use it at our university to provide e-learning services for more than 13,000 students and lecturers (see <http://www.openuss.org>). We published our source code via the Open Source mediators SourceForge.net and Campus-Source.de. Other academic institutions have also adopted OpenUSS for their individual educational needs and successfully installed their own instance [5]. In our experience, the main driver for system adoption was the use of a clearly designed and component-based architecture which is shown in **Fig. 1**.

J2EE makes the development and deployment of OpenUSS components independent of the container vendors. Therefore, we can install OpenUSS components in an Open Source or a commercial J2EE container [6]. However, for our own OpenUSS instance we only use Open Source J2EE containers, libraries and frameworks to exploit the advantages of OSS. Using an Open Source *implementation* on a standardised *specification* like Java and J2EE reduces

system specific investments, such that economic barriers of adoption are eradicated.

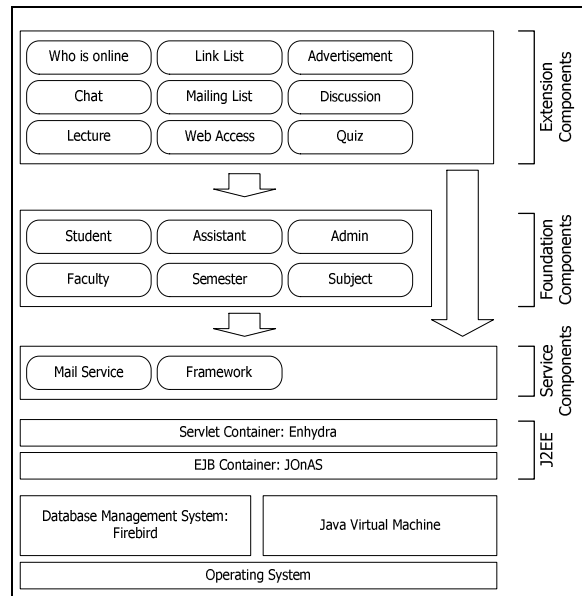


Fig. 1: OpenUSS Component Architecture

In case that the chosen Open Source J2EE container does not scale and perform adequately, it is easy to move OpenUSS components into a commercial J2EE container. Besides, each OpenUSS component can be executed on any distributed network node. Consequently, the system is able to distribute workloads on peak and shows a high degree of scalability. This feature is very useful, since LMS are – generally speaking – periodical applications. In particular, workloads do not occur uniformly but occasionally. For instance, the load of an LMS is high during semesters, and – as we observed – has its peaks during examinations. Consequently, the system has to operate absolutely reliable.

3. OSS Adoption problems

Since our OpenUSS project is a pure Open Source approach, we encountered a broad spectrum of different adoption barriers. According to the TOE-framework [7], adoption problems relate to *technological*, *organizational* and *environmental* aspects.

Technologically speaking, we encountered problems concerning our J2EE layer. The employment of Enhydra as servlet container was not a problem, because the product was already mature at that time. Concerning the EJB container, two Open Source products were available:

JOnAS and JBoss². After analyzing both products, we decided to use JOnAS because of its simplicity and the availability of its documentation. In the beginning we had stability problems with JOnAS because this product was immature at that time. But as usual, these problems disappeared with the availability of upgrades. The main problem we encountered on the technical level was the low performance of the system as we use OpenUSS to service mass lectures with more than 1,000 students at one time.

In order to solve this problem, we introduced a multi-server environment with load balancing features (Fig. 2).

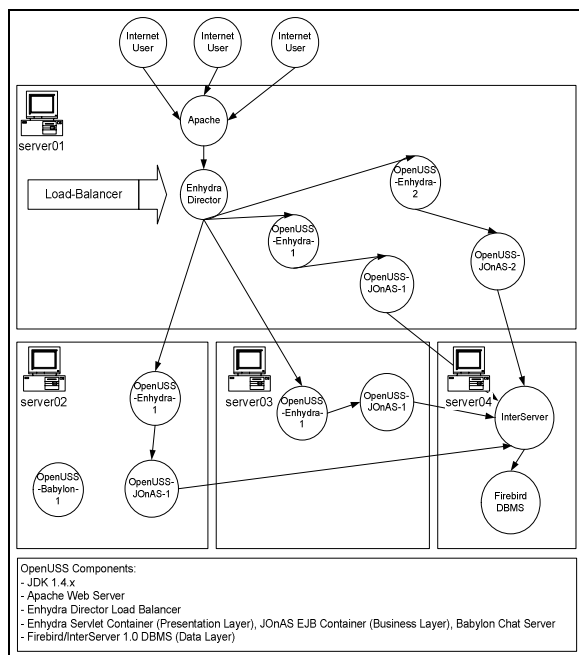


Fig. 2: OpenUSS Physical Architecture

We decided to deploy OpenUSS over four distributed Intel-based servers. In particular, three servers provide presentation and business layers of OpenUSS, whereas one server acts as dedicated database machine. Fig. 2 shows the distribution scheme of OpenUSS [8]. In the front we use a load balancer to distribute the incoming requests among available presentation layer components. Subsequently, they forward the requests to the corresponding business layer components which carry out business processes (e. g. document download, mailing lists). In many cases the business layer components read or write data to the database to collect the necessary result. Afterwards, the business layer compo-

nents send the calculated result back to the user through the same way. In this physical architecture we clustered the presentation and business layer components, so that failure of those components can be balanced by others. Still, we have *single point of failures* remaining in some other components in this architecture. For example, in case that the database server will fail, the whole system will also fail. The remedy of this situation is to use clustered databases. Using an Open Source product like C-JDBC (Clustered JDBC) we can simply add a cluster of databases without having to change the system itself [9]. This extension will make the architecture a lot more robust, since each server node has the same functions and failure in a single node will not affect the system as a whole. By using distribution techniques, we are able to meet the different periodical workloads (rightsizing). For instance, during semesters it is necessary to provide high capacity for serving 1,000 users simultaneously, whereas during semester break, there is just a single server necessary to cope with some basic load.

By choosing J2EE as technological base, prerequisite skills and specialised knowledge for effective software development are indispensable. Due to the low level of diffusion of profound knowledge on J2EE software engineering, only few developers were available to help building or maintaining OpenUSS, even at university. This would be different if we would have chosen PHP, which is very popular for Open Source projects, in particular for the domain of e-learning. In our project context, we teach our students J2EE programming in order to develop the human resources necessary for system development and maintenance. This was a significant effort to eliminate the lack of skills as the dominant organizational barrier in OpenUSS adoption.

Environmental barriers evolved from the supplier side of our project. We used a variety of Open Source tools, which we evaluated by hands-on testing procedures. Of course, this process was very time consuming and represents the main cost driver in our project. According to our experience, this area of research deserves further scientific and practical attention. For instance, Open Source intermediaries like SourceForge.net could supply more instruments to support structured evaluation techniques.

² JOnAS and JBoss can be found at <http://www.objectweb.org> and <http://www.jboss.org> respectively.

4. User Activity and Acceptance

In order to track the diffusion process, we set up different instruments to observe user activities and to analyse the acceptance. To draw a complete picture, we decided to use unobtrusive and obtrusive methods of data collection. A valuable data source are server based log files. These web logs provide all elementary user events as a history of system usage and can be analysed by using web log analysing tools. Since web logs just report empirical behaviour [10], we also decided to use dedicated, obtrusive means of usability testing for the presentation front end of OpenUSS³ [11] [12].

Within our web log analysis, we analysed user activity for our winter semester 2002-2003 (six months). During that time, there were 8,000 users available on the LMS. An analysis of the user activity is significant since we have to observe the user needs carefully to prevent bottlenecks. Functionality and performance represent the most important bottleneck types on the system. In the face of limited resources, it is useless to invest a lot of time on a functionality, which users will not use anyway. This also applies to the performance of the system which plays an important role for user acceptance.

Despite the fact that there are numerous Open Source software products which are able to analyse server based log files, it was necessary to develop suitable analytical software. This was necessary since log file entries of J2EE software packages are very cryptic and need an intense integration of metadata (e. g. the name of a lecture instead of its index number). With this software we are able to create standard reports like the top 10 page views (shown in Fig. 3).

Through web log analysis we found that the most used function in OpenUSS is the download component for Lecture Materials with 170,000 page views (see Fig. 3). This implies that most students use the LMS to obtain their lecture materials. Typically, lecture materials are digital documents stored as Microsoft PowerPoint or Adobe PDF file formats. This business process is critical, since it implies significant loads for the LMS due to the high degree of network traffic. The second function is the Discussion Forum component, besides the login page of the Foundation component, with more than 25,000 page

views. These results enabled us to fine tune the performance for both functions to increase user acceptance.

Furthermore we found interesting results by looking at the average number of login requests by weekday (see Fig. 4). Starting on Sunday, the requests to system login increase because the students have to prepare their lectures for the coming week. On Friday, LMS usage decreases dramatically as the weekend gets closer.

The singular user activity analysis is not sufficient to evaluate user acceptance. Therefore we performed a usability test on representative users based on usability engineering techniques [13]. There are two roles of users available on the OpenUSS system: lecturers and students. The test is done on the latter. In the beginning, it is important to construct the representative user profile. Our study focused on male and female student users between 20 and 30 years old. Basic knowledge about computer and internet is available. Altogether, there were 73 users which are subdivided into three groups (see Tab. 1).

Test Group	Usability Methods
Test group A: first time users of OpenUSS (N=48 persons)	Scenario Based Testing
Test group B: first time users of OpenUSS (N=5 persons)	Scenario Based Testing combined with Think Aloud Method
Test group C: experienced users of OpenUSS (N=20 persons)	Questionnaire

Tab. 1: Test Groups and Methods

The testers from the test group A and B received some tasks to be accomplished within a given time line. An example task looks like: "You are at the start page of OpenUSS (<http://www.openuss.org>), please apply for a login und log in afterwards. This task is finished as soon as you are located at your personalised home page." The time restriction for this task was 300 seconds. The result of this task is depicted in Fig. 5.

³ These studies were carried out by students in scope of their Bachelor or Master thesis at the University of Muenster, Department of Information Systems.

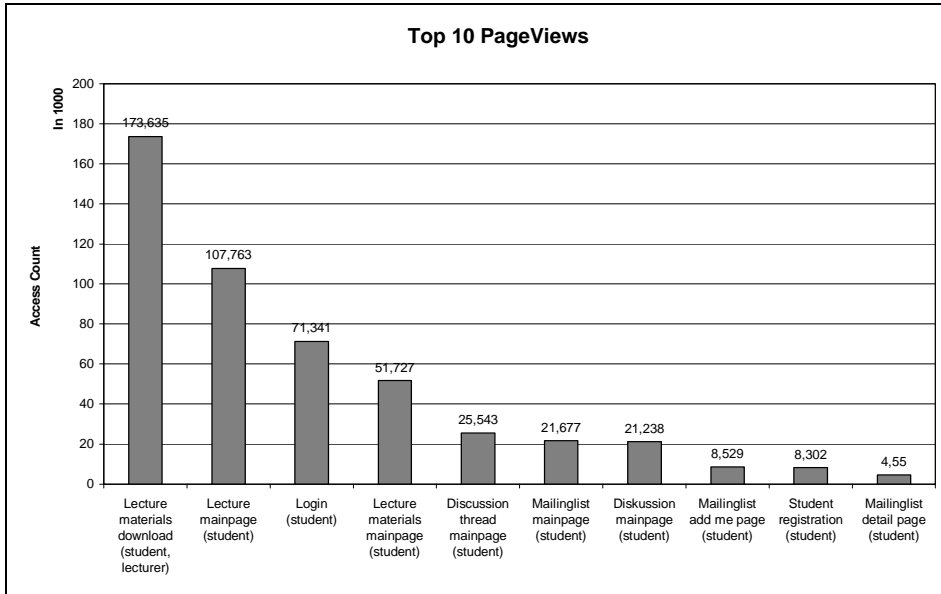


Fig. 3: Top 10 Page Views

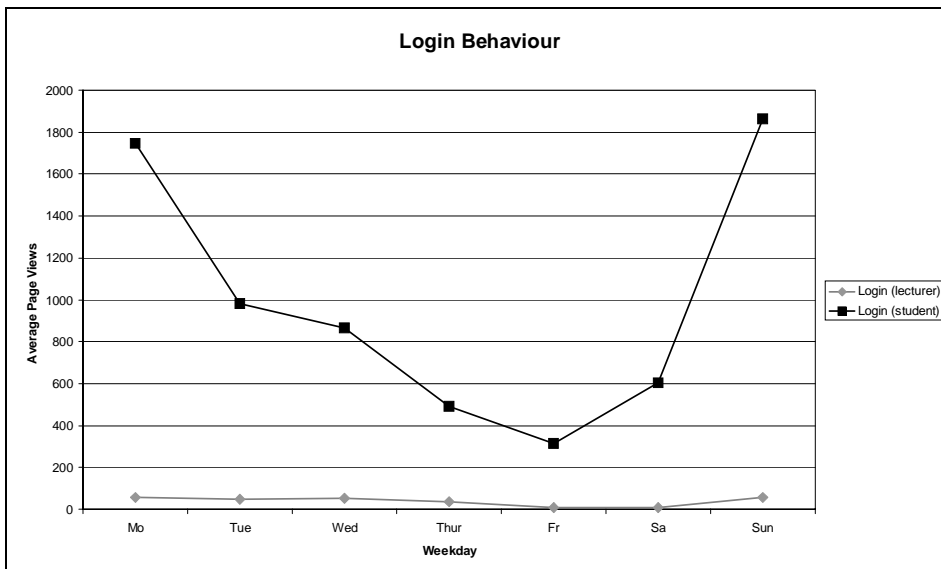


Fig. 4: Login Behaviour by Weekday (avg.)

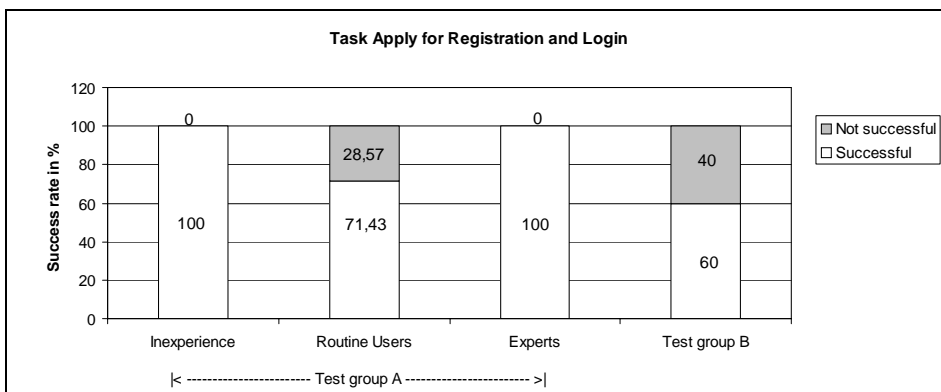


Fig. 5: Assignment to Apply for Registration and Login

It is interesting to see that almost 30 % of the routine users of computer and internet from the test group A were not able to complete the task within the given time restriction. In contrast, all inexperienced users from the same group could finish the task without any difficulty.

Another approach was carried out on the test group C, which has to answer a set of questions based on their subjective experience with the system. The questionnaire offered a scale from 0 (bad) to 9 (excellent). **Tab. 2** shows the average result of the questionnaire, which divides into five categories. Each category contains one or more questions.

Category	Average Result (0 = min, 9 = max)
General impression	$\mu = 5.5$
Display view	$\mu = 6$
System messages and menu expressions	$\mu = 4$
Ease of use	$\mu = 4.5$
System performance	$\mu = 6$

Tab. 2: Average Result of the Questionnaire

It is obvious to see that the ease of use (help text, testing on new functions, execution of tasks and operation on the user interface) and the system messages need an intense rework. On the other side, it is discernible that the system rates above average for the categories general impression, display view and performance. This confirms our basic technological hypothesis that the chosen J2EE technology is adequate for the implementation of mission critical systems with high server loads.

5. Summary

During the development process of OpenUSS, we learned that it is possible to create and operate a robust LMS using pure Open Source tools and environment. In particular, building and deploying a component-based LMS with J2EE is an excellent way to manage system complexity. Since we used J2EE as core technology, we had to develop the system from scratch. There were only few J2EE components available within the Open Source, such that we were even forced to develop basic services like mailing lists and login procedures.

Other universities adopted OpenUSS and customised it to meet their needs (e. g. language translation). Although an Open Source solution

like OpenUSS can offer significant financial advantages compared to commercial products [14], the rate of the platform diffusion is rather low. In our experience the most critical success factor for adoption is the presence of profound skills in J2EE software engineering and hence the ability to manage J2EE complexity. To lower this barrier we created a new infrastructure project to alleviate the development of J2EE components in particular for OpenUSS (see Enterprise Java Open Source Architecture – EJOSA at <http://ejosa.sourceforge.net>) [15].

After the development part we had to deploy the system within a production environment. One should not underestimate this task as it took a long period to optimise the system environment to cope with the massive load. Therefore it is certainly worthwhile to offer an ASP solution to other institutions, which do not have the capacity and knowledge in maintaining a production environment.

At the end of the software lifecycle it is necessary to evaluate the LMS to analyse user acceptance and activities. From the evaluation's result it is obvious that it is not easy to build a user interface for LMS, which is acceptable to all users of the system. Clearly arranged navigation and function bars help to improve user acceptance as many of students and lecturers are still new with the environment of e-learning.

The result of those studies will be absorbed into the next lifecycle of system development and will contribute to the continuous improvement of the whole system.

6. Acknowledgements

We would like to thank Kerstin Müller and Henning Eiben for their scientific and practical engagement on the OpenUSS development project [11] [12].

7. References

- [1] B. L. Dewanto, "Learning Java Programming Language with Open Source Products and Technologies," <http://edu.netbeans.org/support/oss.html> (current July 2003).
- [2] G. T. Heineman and W. T. Council, *Component-Based Software Engineering – Putting the Pieces Together*, Addison-Wesley, 2001.
- [3] R. Monson-Haefel, *Enterprise JavaBeans, Second Edition*, O'Reilly, 2000.

- [4] W.-G. Bleek, I. Jackewitz, and B. Pape, Matching Needs Application Service Providing for Asynchronous Learning Networks, in: Proc. of the 36th Hawaii International Conference on System Sciences (HICSS'03), IEEE, 2003, p. 76.
- [5] ObjectWeb.org, "JOnAS and Enhydra pass OpenUSS entrance exam," http://www.-objectweb.org/wws/d_read/marketing/-public/SS_OpenUSS_2p.pdf (current January 2004).
- [6] TheServerSide.com, "Application Server Matrix," <http://www.theserverside.com/-reviews/matrix.jsp> (current July 2003).
- [7] R. Depietro, E. Wiarda and M. Fleischer, The Context for Change: Organization, Technology and Environment, in Tornatzky, L. G. and M. Fleischer, The processes of technological innovation. Lexington, Mass.: Lexington Books, 1990, pp. 151-175.
- [8] H. L. Grob, Informationsverarbeitung in der Hochschullehre, Working Paper, Computer Assisted Learning and Computer Assisted Teaching, No. 25, Muenster, 2003.
- [9] E. Cecchet, J. Marguerite, W. Zwaenepoel, "RAIDb: Redundant Array of Inexpensive Databases", 2003, <http://c-jdbc.objectweb.-org/current/doc/RR-C-JDBC.pdf> (current January 2004).
- [10] F. Bensberg, Web Log Mining als Instrument der Marketingforschung, Gabler, Wiesbaden, 2001.
- [11] H. Eiben, Analyse des Benutzerverhaltens von E-Learning-Plattformen – dargestellt anhand des Systems OpenUSS, Bachelor thesis, Muenster, 2003.
- [12] K. Müller, Empirische Analyse der Nutzerfreundlichkeit von Learning Management-Systemen – dargestellt anhand OpenUSS, Master thesis, Muenster, 2004.
- [13] J. Nielsen, Usability Engineering, Morgan Kaufmann, San Francisco 1994.
- [14] F. Bensberg and B. L. Dewanto, "TCO VO-FI for eLearning Platforms," Poster Abstracts of the 25th International Conference on Information Technology Interfaces (ITI 03), SRCE, Zagreb (Croatia), 2003, pp. 9-12.
- [15] B. L. Dewanto, "Enterprise Java Open Source Architecture, User Manual", 2003, <http://prdownloads.sourceforge.net/ejosa/ejosa1.3.5-doc.pdf?download> (current January 2004).