PROPOSAL FOR THE DESIGN AND IMPLEMENTATION OF A MODERN SYSTEM ARCHITECTURE AND INTEGRATION INFRASTRUCTURE IN CONTEXT OF E-LEARNING AND EXCHANGE OF RELEVANT DATA

Sebastian Pätzold, Sabine Rathmayer, Stephan Graf (Universität München)

Abstract: This paper discusses approaches for system integration in the context of eLearning applications, Learning Management Systems, and Campus Management Systems. Three current projects (elecTUM, CampusSource Engine Integration Platform, and e-Framework) are discussed. Finally a Service Oriented Architecture (SOA) for these integration tasks and an abstraction layer of Campus Management Systems is outlined.

Keywords: integration, eLearning, service, SOA, ESB

Introduction

The usage of eLearning methods and tools is getting more and more important at universities. This is because of the rapidly changing situation there today. Students demand a higher quality and efficiency in teaching. One example is the centralized storage of documents and resources for courses, another one is the demand for continuous self-assessments to control the personal learning progress. Furthermore the usage of collaboration and communication tools like forums or chats is an important factor. Another major driver of this development is the establishment of the Bologna Process in the European Union. The restructuring of existing programs of study to Bachelor and Master degrees have often caused a dramatic raise in exams per semester. To solve this problem more and more eTests are used as these reduce the workload of examiner significantly.

The points mentioned above provide very good arguments to integrate eLearning supported by a Learning Management System (LMS) in academic everyday life. However this is a major project with various pitfalls. We have extensive knowledge with these current eLearning topics, as a major part of our work in the last years in the elecTUM project concerned the establishment of the central commercial LMS CLIX from the vendor imc AG at the Technische Universität München, one of the largest technical universities in Germany. In the course of this still ongoing project it has become more and more obvious that commercial as well as open-source LMS are lacking next-generation features (e.g. adaptability and customization by users). We are designing and pioneering new approaches to solve these shortcomings. One major concern is the design of a system-architecture to allow for an easier integration in existing campus systems and applications.

For the success of an LMS it is vitally important that it is integrated with other campus applications. In our experience the use of a central LMS is only tolerable for lecturers if data which was previously entered in other systems is also available in the LMS automatically. Furthermore actions initiated in one system have to affect others as well. Everything else is not understandable and acceptable from a user's point of view and hinders the introduction of eLearning at a university. Especially the exchange of data between an LMS and Campus Management Systems is important. These days Campus Management Systems are providing a large number of functionalities regarding activities and data administration within Higher Education Institutions (HEIs). This includes managing the complete student life cycle with accompanying data. As eLearning is becoming an integral part it is obvious that it uses nearly the same basic data as other applications with little domain-specific variations and extensions. This includes among others:

- personal information about students, lecturers and employees course data,
- exam data,
- information concerning the curriculum and program of study student grades.

Not only the basic data is nearly identical, but also a set of actions is shared among an LMS and Campus Management System. Examples include the initiation of course booking or adding a grade to a student's record after an exam. Considering this still incomplete listing of both data and actions it is evident that a broad overlapping between a Campus Management System and a common LMS exists.

In order to give learners as well as lecturers a better user experience, and to avoid the necessity to re-enter the same data in a variety of systems or inconsistencies in data sets because of actions which are carried out on one system but not the other, it is obvious that a strong demand for integration and exchange of information between applications exists.

This paper gives an overview about the current state of integration between LMSs and Campus Management Systems. For that three different approaches and projects are discussed. Afterwards problems are highlighted and discussed. The last section presents a modern system-architecture and integration infrastructure based on a Service Oriented Architecture (SOA) using an Enterprise Service Bus (ESB) which can help creating more flexible integration and system coupling solutions.

Existing approaches, projects, and initiatives

As system integration and coupling between eLearning systems and Campus Management Systems are a very prominent topic, naturally, various projects exist which address the identified problems. These are presented in this section.

Application-To-Application integration

Most projects today focused on the integration via interfaces of one LMS with a specific Campus Management System. This approach is often described as Application-to-Application integration (A2A). Most of the time these projects are quite focused, e.g. the feature set is reduced to the transmission of course data from a course management tool. In order to achieve a communication between the mentioned systems, appropriate interfaces have to exist in both systems. Common interface or communication technologies used in this context include web services (e.g. SOAP, XML-RPC or REST) and message exchange via a Message Oriented Middleware (MOM, e.g. Java Messaging Service (JMS), CORBA, BizTalk).

In the course of the project elecTUM conducted at the Technische Universität München, we designed an extension to the JMS based ERP interface included in the LMS CLIX cooperatively with the software vendor imc AG. This interface is usable in CLIX 7.0 and later. The features include the synchronization of courses, course types, booking data, as well as grades as a result of eTests and regular exams. See figure 1 for an overview of synchronized data and communication directions:

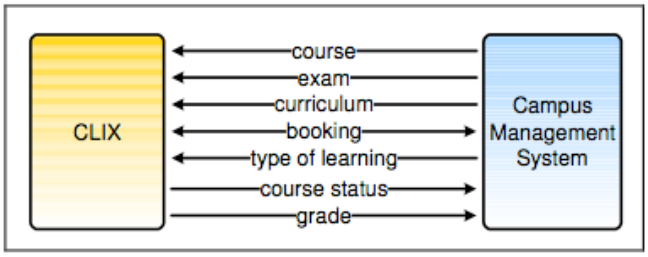

Figure 1. Overview of synchronized data

In order to avoid inconsistencies in the data set in the involved applications the Campus Management System is treated as the master system. As an example course data is only editable in the course management component of the Campus Management System. Upon saving the changes, CLIX is notified via a message and the data set is updated. Additional editing of the data in CLIX is restricted to parameters which are only relevant in the context of eLearning not included in the general lecture data. For most communication types listed in Figure 1 the Campus Management System is the initiating the communication. However, sometimes also the exact opposite is true. This is the case when a student books as well as finishes a course using CLIX, or receives a grade in an eTest. In these scenarios the LMS has the role of providing this data to the Campus Management

System which in return checks the submitted data and triggers the relevant logic, e.g. the check if a student fulfills all prerequisites required to book a certain course.

At the time of the design of this interface it was thought that the course management system LSF from HIS GmbH would be introduced at the Technische Universität München to complete the already productive exam and student management components POS respectively SOS from the same vendor. However, after careful review of this system an introduction was deferred. Recently the Technische Universität München decided to commission all existing applications and replace them with a unified system from another vendor. Furthermore in order to use the interface designed, a correspondence in the Campus Management Systems is needed. LSF, POS, and SOS offer some web services and command line batch utilities for searching, reading, and updating data. However, those were not useable with the implemented CLIX interface based on JMS. Adaptation was needed which proved difficult to achieve as these components had to be designed and implemented as an individual customization. Also long-term support for those custom implementations seems doubtful.

In order to use the CLIX ERP interface nevertheless, a connector was implemented to make use of the data stored in the Course Management System UnivIS. This system is used at the Technische Universität München for administering and presenting course, department, and person data via a web interface without further features like booking of courses etc. UnivIS offers only little means to access the data other than using the described web interface. The only other method available is an XML export which can be triggered via the web interface or calling a shell script. In case of the connector to CLIX the latter method is used. The extracted XML data is converted to CLIX XML messages using XSL transformations which are then submitted to CLIX using the ERP interface and JMS. Currently, the curriculum and courses for one semester are transmitted to CLIX. This is done once at the beginning of each semester. Continuous synchronizations of changed data in UnivIS are not possible at the moment.

e-Framework

The e-Framework project is an initiative initiated by UK's Joint Information Systems Committee (JISC) and Australia's Department of Education, Science and Training (DEST). It aims at providing technical interoperability in the research and education application domain [1]. This is done through initiating a planning process for a service oriented approach in a community of experts.

Another problem addressed is the establishment of a shared vocabulary to ease collaboration among partners. The most important terms in this context are:

- Services,
- Service usage models (SUMs).

Tasks relevant in an interoperable environment are abstracted into services. This is beneficial if identical tasks are used in different applications or an integration of systems can be achieved by their usage. Services are identified by developing SUMs. These SUMs describe the relationship of services and their interactions with each other in a specific domain. As a main resource for developers these help to plan and implement interoperable applications by designing the software according to these specifications.

Summarized, this project aims at providing a knowledge base as a strategic resource to foster interoperability in an academic context and as a foundation for integration projects. It is not a product useable out of the box.

Campus Source Engine Integration Platform (CSE)

The CampusSource Engine Integration Platform is a project by the Campus Source Initiative NRW currently in development. The project aims at developing a custom middleware for the integration of various LMSs with Campus Management Systems. The project has integrated several LMSs so far, namely CLIX, ILIAS, and EWS II.

The integration platform currently focuses on connecting these LMSs with the course management component LSF of the Campus Management System suite from the software vendor HIS GmbH. According to [2] the following data objects are transferred:

- Course data,
- Data concerning booking and cancellation of courses.

LSF istreated in this scenario as the master system. This essentially means that all data which is generated in the LMS and relevant to the Campus Management System needs approval.

In future developments the integration with other systems should be implemented [3]. For example synchronization with the component for the management of exams, grades and programs of study POS-GX of the HIS GmbH is currently planned. Other systems which will be integrated are Moodle and OpenUSS.

As a technical infrastructure the JBoss Application Server is used [3] which offers management interfaces, good scalability, and several other services out-of-the-box. Connectors to the various systems mentioned above are implemented using technologies like web services and Java Message Services. These standards are used intentionally to build a future-proof system. The project team has released the source code under the GNU General Public License recently.

Discussion of existing approaches, projects, and initiatives

In this section we would like to highlight positive aspects and drawbacks of the presented solutions. First of all the pattern of Application-To-Application integration was discussed. This approach is straightforward and may very well be a fast solution if all communication partners are known and rarely changed. Furthermore the number of applications in the scenario has to be small. However, this is often not the case, as the deployed systems change eventually and new applications are introduced in the integration scenario. A severe drawback of the application-to-application approach is the rapidly increasing number of needed interfaces in case of the rising number of interconnected applications. Each application needs to have a specially designed or adopted interface to communicate with a target system which itself needs such a point of access. In case of n interconnected systems $n^2 - n$ interfaces are needed. Of course this increases the cost for maintenance, as each change in one application's interface has to be accounted for in all other interconnected interfaces. Additionally, interfaces may very well be not realizable because of economic considerations, lack of support of software vendors in case of commercial products, or missing skills of employees responsible for an application.

Next UK's Joint Information Systems Committee (JISC) and Australia's Department of Education, Science and Training (DEST) initiative e-Framework was presented. The effort is very promising as it aims at providing insights on how to provide interoperability in an academic context. The project outcome is a knowledge base and community of experts which is a rich and highly valuable resource providing insights in planning and implementing an interoperable IT infrastructure at education institutions. However, the e-Framework is not a product which can be readily used to interconnect applications and systems.

Last but not least as a product the CampusSource Engine Integration Platform (CSE) was highlighted. This projects aims at creating an adoptable integration solution for academic needs. However, currently it is focused only on supporting Campus Management Systems of the German vendor HIS GmbH. In the past the development was focused on concrete integration projects with some German universities. The source code was only recently open sourced under GNU General Public License (GPL). In the author's opinion it is crucial that CSE gains attention from other developers which help to further develop the platform and help broaden the portfolio of integrated applications. It has yet to be shown that the used concepts are generic enough and barriers acceptable for this to happen.

A modern system architecture for integration

In order to solve the shortcomings which we as a project team experienced in the past it is apparent that a new approach is needed. As there are certain aspects which were outlined especially in the Application-To-Application integration section (e.g. the expected - yet unrealized - usage of LSF as a Campus Management Systems), this approach has to be designed concerning following requirements:

- The new approach needs to be highly generic and provide a maximum of abstraction so that a changing system environment can be dealt with efficiently and avoid maintainability issues;

- The integration of systems without the requirement to design special interfaces is needed as such a customization is expensive and maintainability is an issue. A prerequisite for this to happen is the usage of all kind of interfaces offered by existing applications, e.g. existing web services; file import/export interfaces; database interfaces etc.;
- Interfaces to applications should be specified according to functional aspects. Technical or system specifics should be hidden to increase the durability and usability in an evolving system environment.
- Implemented integration services should be reusable by all potential users according to defined policies as it is obvious that many other, yet unforeseen, use case for these exist.

Furthermore advantages of other projects like e-Framework and CSE should be considered. These are:

- The abstraction of tasks into interoperable and commonly shared services (e-Framework) as a way to enhance reusability and abstraction of a system environment.
- The usage of a dedicated middleware (CSE) to allow an easy integration of applications without adapted interfaces, to increase maintainability, to introduce central management interfaces, and to build a highly scalable solution.

According to these principles a specific architecture is presented in the following section.

Service Oriented Architecture

Service Oriented Architecture (SOA) is a buzzword these days. In the context of an integration scenario which is discussed here, it can prove most useful.

According to [4] a SOA is defined in the following way:

> *"Service Oriented Architecture (SOA) is a computer systems architectural style for creating and using business processes, packaged as services, throughout their lifecycle. SOA also defines and provisions the IT infrastructure to allow different applications to exchange data and participate in business processes."*

An overview of a possible layout of a SOA is presented in Figure 2.

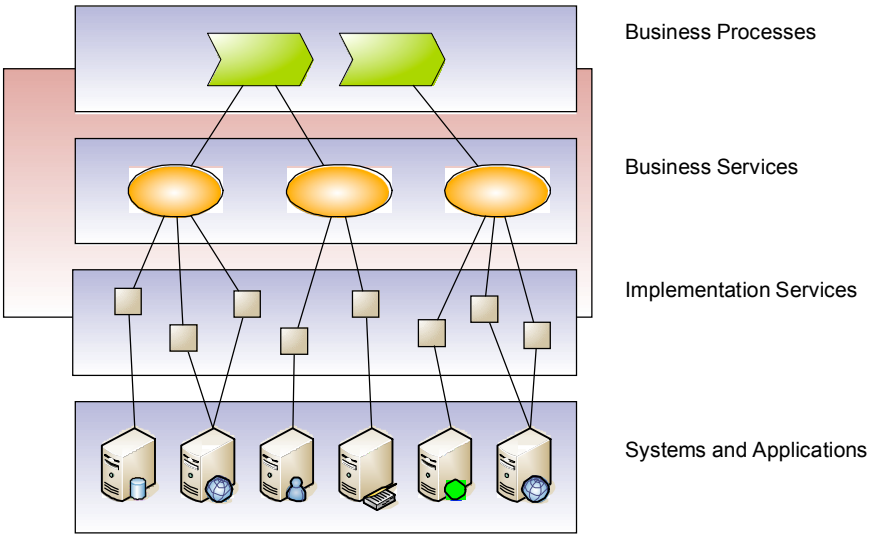

Figure 2. Overview of a SOA

Most of the different layers seen in the figure are presented in the following sections. To illustrate the concept of SOA in the context of eLearning and universities a common example is used. The exchange and synchronization of lecture data of a provider (e.g. the Campus Management Systems) with other systems or applications (e.g. an LMS, faculty portals) is such an example. Here the situation at the Technische Universität München as described previously is used to illustrate this example.Business Processes.

One of the most important terms in the context of SOA is the business process. A business process is a sequence of tasks which are needed to achieve a certain business goal. Essentially these processes run the business and create value. Business processes can be combined from various sub-processes.

In an academic context a valid business process is the exchange of lecture data with all relevant systems. This satisfies the business goal to have accurate, identical, and possibly quality assured lecture data available in all systems ready to use by students and employees.

Business Services

As mentioned above business processes are a mere sequence of interconnected tasks. These are often called business service (though sometimes the term is also used to describe a technical front-end used to trigger a business process). Business services thus are vital for business processes to actually do something.

Business service interfaces have to be designed according to functional aspects. This is necessary to create durable interface, increase usability as well as reusability. That is why often standard data formats can be utilized here with great effect. Usable standards can be defined by examining which data is needed in a certain application domain. In a second step business objects are derived from this work. As an example the project elecTUM defined valid business objects for eLearning which were later used for the implementation of the CLIX ERP interface.

In the section above the exchange and synchronization of lecture data was introduced as a business process. However, as it is also an action it is a valid business service as well. This "course announcement service" is tasked with accepting lecture data via a well-defined interface and forwards it to other systems which use this data to create a new course or do an update on existing entries. A possible interface might be designed according to the Course Description Metadata (CDM) format which specifies an XML based exchange format for lectures. It is used worldwide, thus can be considered as a pseudo-standard.

In Figure 3 an overview of this business service and its communications paths is given. The figure depicts the situation at the Technische Universität München with its applications and systems as mentioned previously.

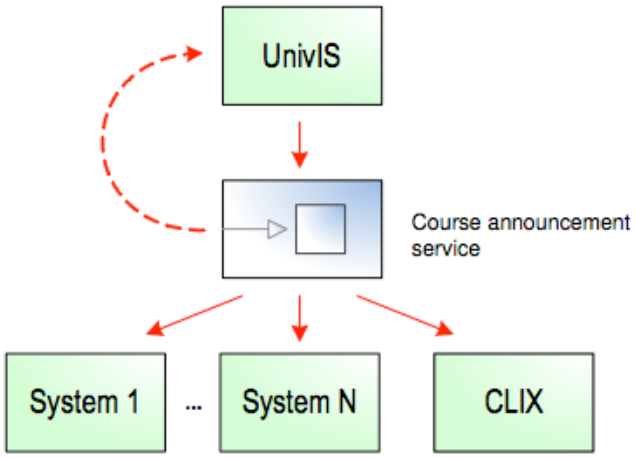

Figure 3. Business service "Course announcement"

Implementation Services

Implementation services close the gap between business service and a concrete service provider (application, system). This type of service converts such a component to a usable unit in the context of SOA. Of course this is a rather complex task in the case of so-called "legacy applications" which simply were not designed to work with other systems in the way needed. Most of the time these systems offer interfaces which are however not readily useable in an SOA environment.

As an example UnivIS at the Technand has to be converted to CDM to be usable as input for the course announcement service described above. For this to happen the UnivIS export script is triggered periodically and the output files are stored at a known location. A so-called file gateway observed this location for new data, reads the files, and forwards them to the UnivIS implementation service. As a first processing step, the data has to be split in several messages containing only one lecture each as UnivIS aggregates all courses in a single file. In a second step, the lectures are converted to CDM using an XSL transformation. These transformed messages are then forwarded to the course announcement service.

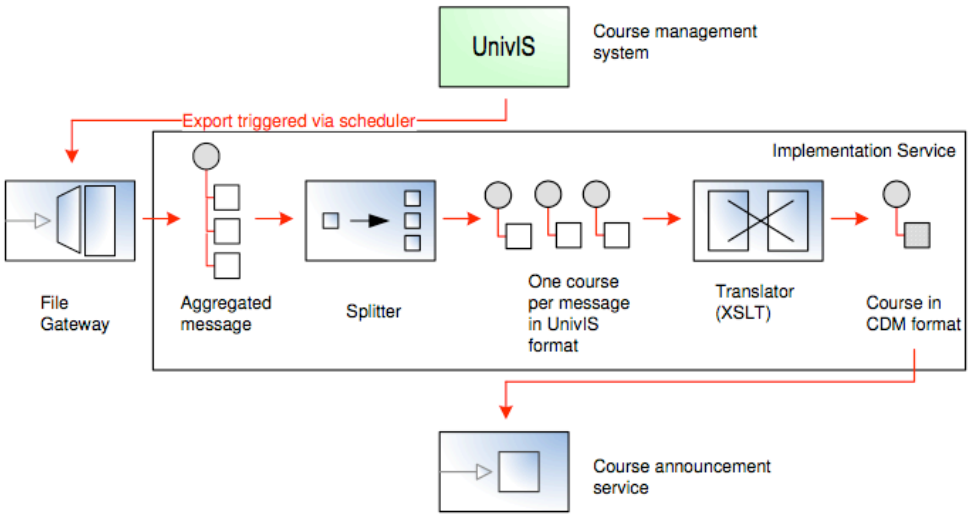

Figure 4. Implementation service for UnivIS

Of course several other services are needed to connect to the target systems interested in lecture data. These are not depicted here, but these are implemented in a similar fashion.

Enterprise Service Bus

The Enterprise Service Bus (ESB, see also [5]) is a middleware pattern often used in the context of SOA. It is a type of Message Oriented Middleware (MOM) used as integration network in a heterogeneous and highly distributed environment. The ESB picks up the role of a mediator between applications with the need to communicate.

There is no common definition of an ESB. That is why it is often interpreted as a set of capabilities, though there is great variance in different ESB implementations. Common capabilities include:

- Message processing for the secure delivery of messages to consumers. This is the inherited classic middleware functionality;
- Data transformation including enhancement of messages with additional data and transformation of messages in other formats;
- Protocol transformation for enabling communication between service provider and consumers using different protocols (e.g. JMS, HTTP, IIOP);
- Routing of messages according to policies, evaluation of rules or content;
- Transaction management to aggregate different work steps to an atomic action which is completely executed successfully or not at all;
- Mechanism for ensuring security including establishing identities of providers and consumers, guarantee message confidentiality as well as integrity;
- Management utilities to control the status of the ESB and the message processing as well as triggering certain actions manually;
- *Process choreography* and *service orchestration* respectively for aggregating business processes and services to a greater whole in a coordinated fashion.

These functionalities are themselves abstracted into services which may be recombined to adapt to specific scenarios. For this to happen it is necessary that a common input and output format is defined.

That is why an ESB works with a normalized internal message format often resembling an email with metadata in a header and a payload in a body as well as various attachments.

In the example mentioned an ESB can be used very well. Business as well as implementation services can be implemented in that way. In particular the UnivIS implementation service uses numerous functionalities. First of all a protocol transformation is applied (file system to internal message format). The splitter is implemented as a special application of a generic routing service, while the translation of the UnivIS data in CDM is implemented by a translation service.

Conclusion

Currently system integration in an academic context is still an issue. All existing projects have made progress so far, but a breakthrough due to several shortcomings is still missing. The proposed architecture might be another brick to build a solid basis for the much needed integration at Higher Education Institutions.

However, much work has still to be done. As a first step all data objects have to be identified which need to be made available to other systems and applications. From this foundation a service catalogue can be extracted which then can be implemented step-by-step. Especially the e-Framework is an interesting resource in this case. The goal is a portable abstraction layer of the Campus Management. Once this has been achieved, lean and efficient applications can be built; data as well as functional redundancy can be reduced. For example a new generation of eLearning tools could become possible, optimized for eLearning tasks and stripped of currently necessary, but redundant, functions, like student management, grade books etc.

References

1. e-Framework Contributors (2006). The e-Framework for Education and Research. A Briefing Paper. [Available at: http://www.e-framework.org/portals/9/docs/papers/briefing060802.pdf, accessed: 23.12.2007].
2. Hüvelmeyer, Josef; Christof Pohl; Manfred Postel (2007). *Konzeption des Integrierten Informationsmanagements von His-Lsf/Pos und Weiteren It-Systemen mit der Campussource Engine.* [Available at: http://www.mz.uni-dortmund.de/aktuelles/publikationen/cse_broschuere.pdf, accessed: 23.12.2007].
3. Pohl, Christof (2007). *Funktionen und Struktur der Campussource Engine. Komponenten, Entwicklungs-Roadmap, Gpl-Veröffentlichung*. [Available at: http://www.campussource.de/events/e0712unidortmund/docs/cse_funktionen_pohl.pdf, accessed: 02.12.2007].
4. Wikipedia Contributors (2008). *Service-Oriented Architecture.* [Available at: http://en.wikipedia.org/w/index.php?title=service-oriented_architecture&oldid=190903634, accessed: 20.01.2008].
5. Chappell, Dave (2004). *Enterprise Service Bus*. O'Reilly.

Authors:
Sebastian Pätzold / Dr. rer. nat. Sabine Rathmayer / Stephan Graf
Fakultät für Informatik, I10
(LRR) Technische Universität
München
85747 Garching bei München, Germany

E-mails:
sebastian.paetzold@tum.de
sabine.rathmayer@tum.de
stephan.graf@tum.de