Christoph Lofi, Wolf Siberski

Service Oriented Architectures for Open E-Learning Systems: An Overview of the Prolix Project

Abstract

Corporate eLearning imposes profound requirements on learning software systems: Learning has to be specially tailored for each employee to reflect their individual workspace responsibilities, respecting the continuous changes of business challenges. Also, other departments might have considerable interest in current status and effects of ongoing or finished learning processes, for instance human resources might need to keep track of the employee's competences while strategic management might be interested in performance changes of business processes which's assigned employees just received training. All these tasks require an eLearning platform to be highly flexible and integrated with other IT-infrastructure already existing in the company. This paper will showcase the Prolix Project which aims at implementing a system and a generic reference architecture standing that challenge for enabling such integrated and wide-spanning corporate eLearning solutions.

1 Introduction

eLearning is a powerful concept for enabling life-long learning in corporate environments. Advancements in the area like for instance personal Learning Assistants are about to complement or even to completely replace the traditional tutor. The courses can be delivered on a computing device directly at the work space or even at home, personalized to the learners' learning styles and preferences. Additionally, past knowledge, specific learner goals and preferences can be incorporated, providing the learner with personalized and specially tailored learning courses adapted to the requirements of their work area. Also, eLearning can be performed in a more cost-efficient manner than traditional training due to its ubiquitous availability, reduced traveling cost, reduced personnel and tutoring expenses and reusability of content.

In fact, these properties make eLearning very appealing for business and workplace training where employees constantly need to develop new or update existing competences for increasing their productivity or coping with business process changes. From the organizational perspective it is crucial for a company's success to have competent, well-performing human personnel mastering ever changing business tasks. As business process experts continuously optimize the business

processes according to the current business challenges, suitable learning arrangements must be derived to train their employees accordingly and keep their competences up to date.

However, deploying nowadays established eLearning applications in a corporative environment often imposes severe problems. Companies usually posses an existing IT-infrastructure spanning their core and auxiliary departments. An eLearning system thus has to be integrated seamlessly with those existing applications like human resource management, business process management or project management. Also, it has to blend with existing identity and security infrastructure to facilitate an easy and safe access for all eligible employees. Unfortunately, most of the current eLearning applications often come as monolithic software applications with all functionality ranging from learning execution components, collaboration and communication components to identity and security modules bundled in one package. This kind of architecture often fails to meet the demand for flexibility and the need for integration of most businesses where learning and education is an important auxiliary but no core function of daily operations.

To resolve these problems, eLearning infrastructures which are open, flexible and standardized are needed so that an eLearning component can easily blend and cooperate with other, existing applications to allow for a unified, transparent workflow and overcome the majority of businesses integration and flexibility demands.

In this paper, we will showcase the Prolix[1] system and its underlying design principles as an example of a federated system coupling adaptive eLearning with other strategic business tasks like business process management, human resource management and process performance optimization. The Prolix Project demonstrates the use of Web Service Technologies and Service Oriented Architectures to develop a standardized reference architecture (so-called Obelix architecture) for such integrated eLearning application federations as well as implementations of that architecture integrating several established software systems tailored for different workflows.

Therefore, we will provide in section 2 a brief introduction to web services, web service orchestration and enterprise service busses which are the core technologies enabling the Obelix architecture. In section 3, we will introduce the Prolix Learning Life Cycle as a basic showcase of a federated system using the Obelix architecture while in section 4, we give an overview of the architecture itself.

[1] http://www.Prolixproject.org, Prolix is a 48 months research and development integrated project co-funded by the European Commission under the Sixth Framework Program, Priority 2 "Information Society Technologies"

2 Web Service Technologies

In the following sections, we provide a generic overview on Web Service and accompanying technologies. These are used in the core of the Obelix architecture and may also provide salvation to integration and flexibility problems in other areas of eLearning and eScience.

2.1 Web Service Technology

The integration of heterogeneous systems used to be realized by the usage of middleware solutions. Middleware systems facilitated and managed the interaction between distributed applications for decades. Attempts of workflow management systems (WFS) and enterprise application integration systems (EAI) paved the way for the most recent approach: Service-oriented architectures (SOA). These differ from previous IT architectures by loose coupling of distributed components and integration over open standards. SOA connects software through a pool of abstract services that provide a well defined, self-contained functionality of a software module. They provide the opportunity to enhance the functionality of legacy systems by composing them with other functional services. Moreover, this enables the reuse of heterogeneous systems in divergent contexts.

SOAs are usually implemented for three reasons: The ability to save development costs by reusing services, the ability to adapt IT Systems faster to the changing needs of a company and the integration of legacy systems into newer IT landscapes using service adapters. The SOA is an architecture for constructing flexible, adaptive enterprise IT systems. This is achieved by providing loosely coupled business-oriented services that are described and accessed via descriptions of their business functions. They can be published, described and invoked independently of the technology used in the service which enables their use in different contexts

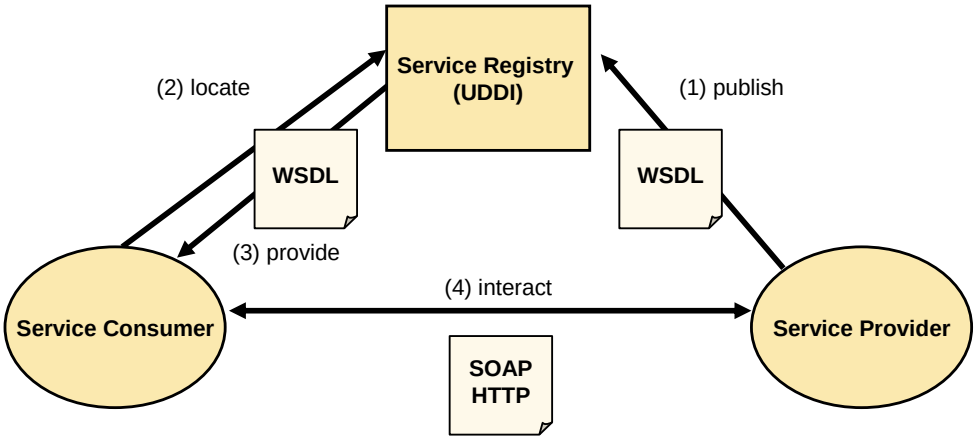

Figure 1: Service Interactions

and ensures adaptability as well as reusability of IT Systems. The idea of services is currently experiencing a climax as standardized web services based on the extended mark-up language (XML)[2] and well-known internet protocols are reaching a current maximum of standardization, reusability and abstraction.

SOA offers enterprises a wide range of advantages, however, they can only profit if their applications are SOA-ready. Today, this implies that they expose Web Services: SOA components are commonly implemented in form of Web Services that a service provider (this may be a company offering this service to others companies for compensation or a department within a company offering the service to other departments) offers via Internet or the corporate Intranet. According to the SOA idea, the service provider is also supposed to publish the Service in a central service registry and to describe functional characteristics of the Web Service, not only its technical interface which is described by the Web Service Description Language (WSDL)[3] as depicted in Figure 1.

In recent years diverse industry initiatives and partnerships have engaged in defining XML languages for integrating distributed systems. In the SOA context protocols for communication, description and discovery of Web Services have established as standards. Figure 2 provides a short overview of the related standards of the web service stack. All the technologies used in the stack are open and free standards which are nowadays widely adopted and supported by all major software and hardware platforms.

2.2 Service Integration deploying an Enterprise Service Bus

Web Services are outstanding tools for offering a certain service in a platform independent, decoupled, well described and flexible manner and provide excellent foundations for distributed, federated software systems. However, the interaction, orchestration and composition of multiple, independent services into one integrated system is not covered explicitly. This section briefly describes approaches for coordinating a system consisting of distributed services and components.

The distributed systems community carries a long history of striving for extensibility: communication protocols, separate communication overlays, messaging formats according to standards and well-defined service interfaces that leave no place for ambiguity.

[2] W3C WSDL (Web Service Description Language), http://www.w3.org/TR/wsdl

[3] W3C SOAP (Simple Object Access Protocol), http://www.w3.org/TR/soap/

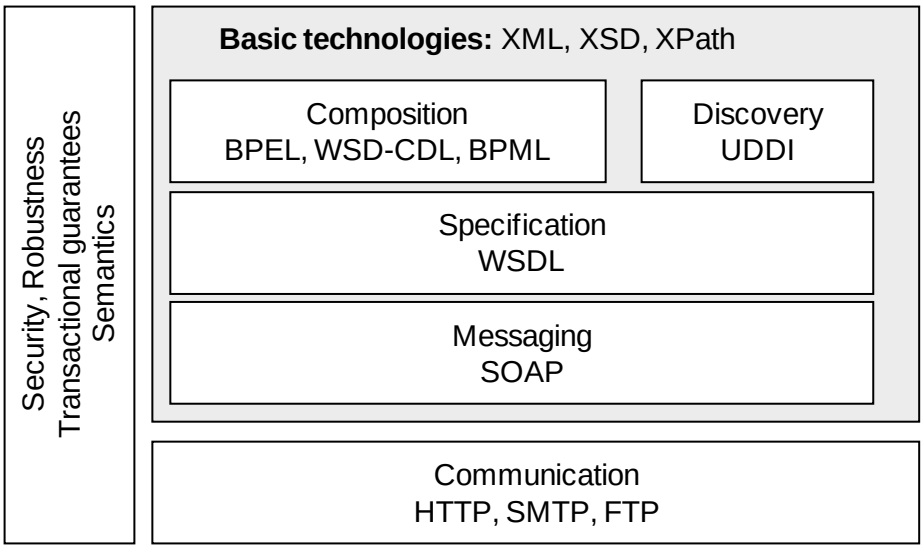

Figure 2: The Web Service Stack

One of the standard approaches in the early-mid 90's was based on the coordination platforms[4,5], taking the coordination aspect away from the components in such a way that components could be developed and handled independently from each other using self-concerned code snippets. In this scenario, distributed components would be handled completely as black boxes. The components themselves needed not to be aware of any interaction that they might have, they only used their input and output ports to interact with the environment (which was the coordination platform), and produced results according to the incoming values.

A more mature approach to the black box components arrived in the late 90's, with the COM-DCOM[6] and CORBA[7] family .The components were still black boxes, but the offered services for each component could be retrieved dynamically, using a single generic interface. While the new service discovery protocol was empowering the work on semantic discovery and orchestration (which the community witnessed in early 2000's), the new approach put the communication aspect back in the component functionality. The components were again made responsible for discovering the services that they required, and a significant programming burden returned back to the component developers.

The golden bullet is now identified somewhere between the two approaches. The components still offer well-defined interfaces, but these interfaces are well-known ahead to a mediator/coordinator, called Enterprise Service Bus (ESB). Being used

[4] CWI Manifold Project, http://www.cwi.nl/projects/manifold/

[5] Magee, J., Dulay N., Eisenbach S., Kramer J.: Specifying Distributed Software Architectures, Proc. of 5th European Software Engineering Conference (ESEC '95), Sitges, September 1995, LNCS 989, (Springer-Verlag), 1995, 137-153

[6] Corba (Common Object Request Broker Architecture), http://www.omg.org/

[7] Microsoft COM/DCOM (Distributed Component Object Model), http://www.microsoft.com/com/

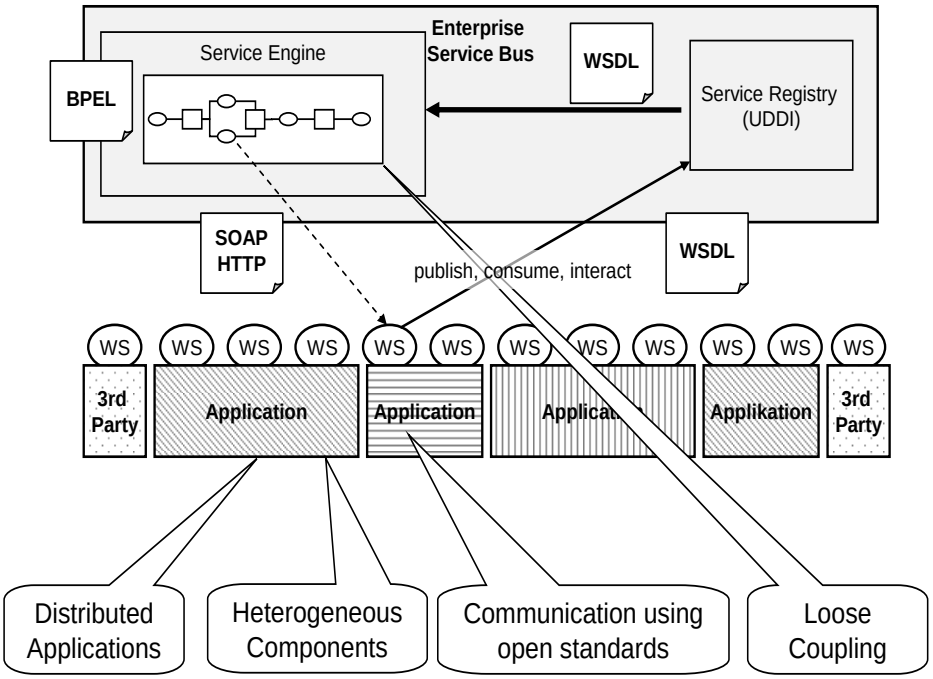

Figure 3: Enterprise Service Bus Architecture

to its full extent, the ESB takes all the components' outgoing messages, uses simple rules for finding the targeted recipients, handles any required transformation (again rule-based), and sends the messages. In this way, the components stay clear of the coordination aspect, and the component developers only bother on well-defining their offered interfaces, and some message transformation rules (if needed). The system architect, having an overview of the whole system, takes care of putting together all the rules, defining any additional transformation, and keeping the ESB up and running. Additionally, security features as message encryption, authorization and authentification can easily facilitated centrally in the ESB rendering insular and closed solution for handling users and access rights obsolete. These features highlight the Enterprise Service Bus approach as an excellent candidate for meeting the requirements for flexibility and openness which are posed by businesses targeting for an integrated eLearning solution.

A schematic illustrating the architecture of service oriented architecture using an ESB is depicted in Figure 3.

3 Overview of the Prolix System

Business process management (BPM) as well as eLearning have become commonplace in modern enterprises. Although most learning activities in companies are initiated from business process changes, both areas re maintained more or less independently in nowadays companies. The goal of Prolix is to remediate this

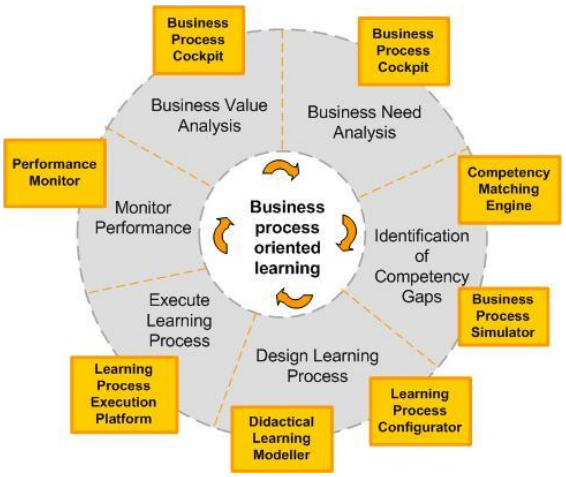

Figure 4: Prolix Lifecycle Overview

shortcoming by integrating BPM and eLearning systems, and supporting the seamless transformation of new skill requirements to learning offers provided by the companies learning management systems (LMS). In this section we first give a short overview of the envisioned integration and then show how LMS features are turned into service-based components and integrated in the whole picture.

One of the core ideas of Prolix is realizing organizational competency management[8] by creating and maintaining competency profiles to employees, tasks and learning material. An employee profile describes the skills and abilities of a staff member. A task profile describes which competencies are needed to perform it. Finally, learning materials are attached with one profile describing the prerequisites for learning, and another one describing the competencies acquired in the learning process

To identify all conceptual and functional interfaces between organizational units and software tools, a complete life cycle – starting with the business need for learning and closing with the actual impact learning made on business performance – has been specified as the Prolix Learning Life Cycle (PLLC, see Figure 4).The innovation in this approach consists of the integrated combination of previously disparate practices and tools from business planning over learning design to execution and performance measurement. Although far from covering all possible application scenarios of Prolix, the PLLC offers a central use case and is used within the project as guideline for specification of the service architecture.

Within the PLLC, changes typically start by a company either quality-checking or redesigning some of its business processes using a BPM application. In both cases, the outcome is the identification of employee *competency gaps*, i.e., skills or com-

[8] Ley, T. (2006). Organizational Competency Management. A Competence Performance Approach. Methods, Empirical Findings and Practical Implications: Graz, Shaker.

petencies which employees need to acquire or improve to conduct their tasks efficiently. As next steps, learning processes which cover the identified gaps are designed or tailored based on existing course material. The final learning design package is handed over to a learning execution platform where employees can register as learners and book the appropriate packages.

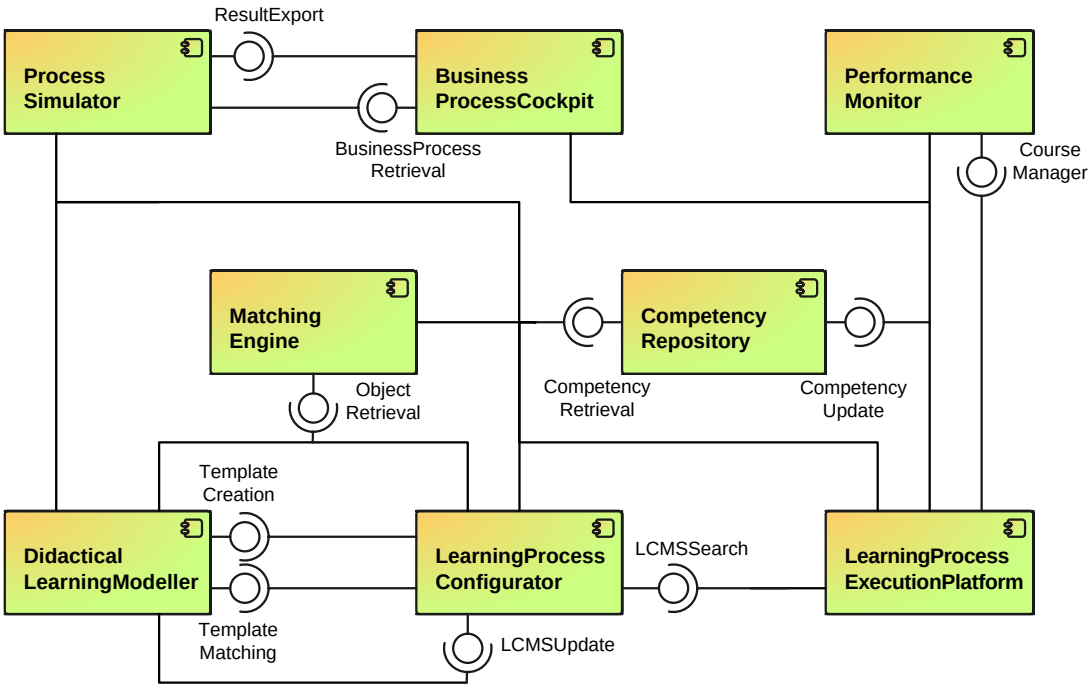

Figure 5: Prolix Architecture

To illustrate how the Prolix service architecture works, we describe in detail the Competency Analyzer (CA), Learning Process Configurator (LPC) and the Learning Process Execution Platform (LPEP) with its interfaces.

Competency Analyzer The CA takes care of competency profile handling. It consists of the competency profile repository, where profiles can be stored, retrieved, and updated, and the matching engine which allows to match objects based on provided and required competencies. For instance, given a competency gap, the matching engine can find the optimal learning processes to close it.

CA Interfaces

Learning Process Configurator The LPC retrieves the competency gaps that need to be covered and configures a personalized learning process for each of the employees that are required to acquire new competencies. In the process of creating the learning processes, the learning preferences of each of the employees, as well as their existing knowledge, are taken into account, so that their educational experience is as effective and as pleasant as possible.

The LPC user, a didactical expert, first retrieves the competency gaps that need to be covered. The LPC application lists the competency gaps per person, and the Competency Matching Engine is queried for each of them to propose learning material that matches the missing competency gaps. The full employees' learning process is configured by combining all necessary learning materials for each person. Finally, the learning process is put into the LPC repository, which stores it in the IMS Learning Design[9] format.

The LPC offers services to retrieve and store Learning Design processes as well as individual learning resources. In Prolix, these services are accessed from the LPEP to retrieve configured courses, as well as from the CA to support requests for suitable learning material.

Learning Process Execution Platform After a learning process is fully configured, the LPEP retrieves it from the LPC repository, stores it in its local repository, and starts the actual training for each user. The learning experience is complemented with testing/evaluation material, specifically targeted for verifying each of the competencies that the learning material provides. At the end of the training, the newly-acquired competencies for each learner and her performance to these competencies (i.e. their grade in the evaluation tests) are saved in the competency repository for future reference.

LPEP interfaces…

4 The Obelix Vision

The Obelix architecture which provide the technological backbone for Prolix allows for

[9] IMS Learning Design: http://www.imsglobal.org/learningdesign/

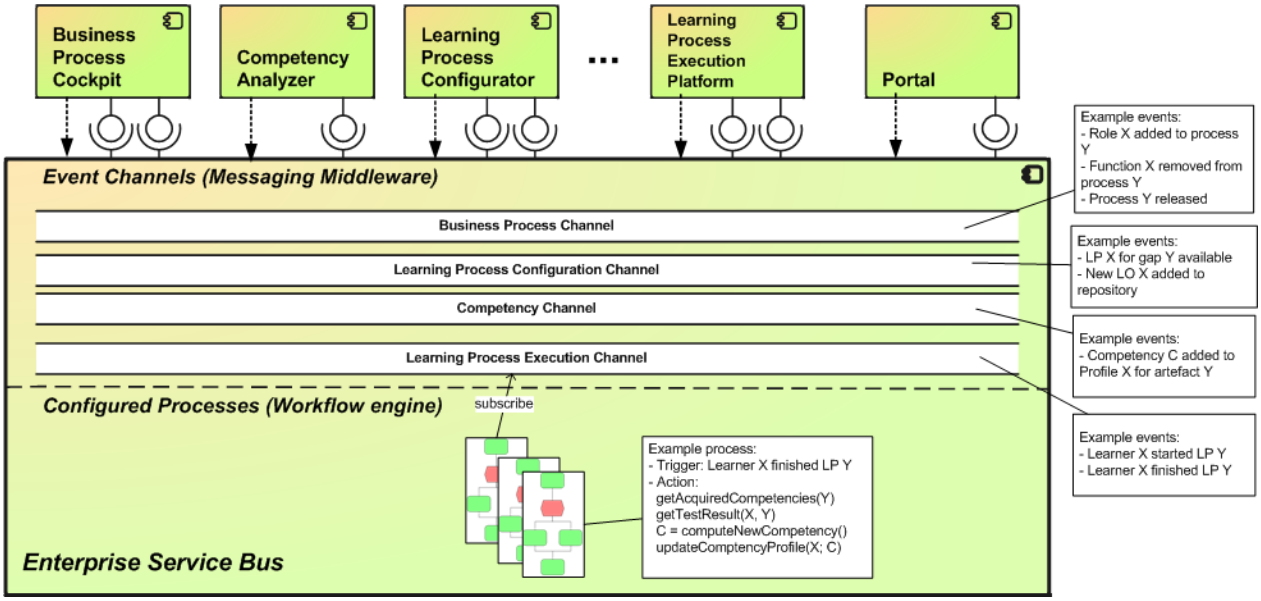

Figure 6: OBELIX Architecture

5 Conclusions

References

The Prolix Project, http://www.Prolixproject.org

Microsoft COM/DCOM (Distributed Component Object Model), http://www.microsoft.com/com/

OMG Corba (Common Object Request Broker Architecture), http://www.omg.org/

CWI Manifold Project, http://www.cwi.nl/projects/manifold/

Magee, J., Dulay N., Eisenbach S., Kramer J.: *Specifying Distributed Software Architectures*, Proc. of 5th European Software Engineering Conference (ESEC '95), Sitges, September 1995, LNCS 989, (Springer-Verlag), 1995, 137-153

W3C WSDL (Web Service Description Language), http://www.w3.org/TR/wsdl

W3C SOAP (Simple Object Access Protocol), http://www.w3.org/TR/soap/

Ley, T. (2006). Organizational Competency Management. A Competence Performance Approach. Methods, Empirical Findings and Practical Implications: Graz, Shaker.

IMS Learning Design: http://www.imsglobal.org/learningdesign/