Maintenance and Exchange of Learning Objects in a Web Services Based e-Learning System

Gottfried Vossen, and Peter Westerkamp
European Research Center for Information Systems, University of Muenster, Germany
vossen@uni-muenster.de
westerkamp@uni-muenster.de

Abstract: *Web services* enable partners to exploit applications via the Internet. Individual services can be composed to build new and more complex ones with additional and more comprehensive functionality. In this paper, we apply the Web service paradigm to electronic learning, and show how to exchange and maintain learning objects is a corresponding e-Learning system. We start from a perception of core *e-Learning activities* as processes, which enables us to break central functionalities of an e-Learning system down into several stand-alone applications; these can then be accessed as Web services. However, building a decentralized system by composing suitably chosen Web services to achieve a functionality similar to that of a traditional e-Learning system leads to a variety of challenges, two of which are discussed in detail: (1) Storing learning content in a distributed fashion, and (2) dynamically exchanging content when necessary or appropriate. The paper also discusses some of the problems arising from storing data on different servers.

Keywords: e-Learning, Web Service, Learning Object, Repository, Distributed Storage

1. Introduction

By moving offline activities online, the emerging paradigm of Web services promises to enable partners to exploit vastly arbitrary applications via the Internet. In a nutshell, a *Web service* is a stand-alone software component that has a unique URI (the *Uniform Resource Identifier* is a unique address), and that operates over the Internet and especially the Web. The basic premise is that Web services have a provider and (hopefully) users or subscribers. Web services can be combined to build new ones with a more comprehensive functionality. The benefits of a Web Services Architecture (WSA) are well-understood in the area of business-to-business (B2B) applications, where companies use it for enterprise application integration; even in business-to-customer (B2C) scenarios, Web services are of growing importance. In this paper, we apply the Web service paradigm to electronic learning and discuss some of the realization problems that arise.

Clearly, Web services need to be interoperable, since individual services typically are restricted and limited in their functionality. Moreover, they have to be independent of the underlying operating systems, they should be usable on every Web service engine regardless of the respective programming language, and they should be able to interact with each other. To achieve these goals, Web services are commonly based on standards; currently most used are the XML-based specifications SOAP (*Simple Object Access Protocol*), UDDI (*Universal Description, Discovery and Integration*), and WSDL (*Web Services Description Language*). Even for the composition of Web services, XML-based languages are being introduced or even used already (e.g., XLANG, WSFL, or BPEL4WS, see Leymann (2001) and Andrews et al. (2003)).

As has been discussed previously, *electronic learning* ("e-Learning") is also taking the shape of a Web service in many applications these days. Here the idea is that learners can, for example, search for content suitable to their needs, book it, pay for it, and finally consume it, all by composing appropriate lookup, payment, and presentation services, resp. The basics of a platform called *LearnServe* providing this are the subject of this paper. LearnServe starts from the perception that a typical learning system is a collection of activities or *processes* (Vossen et al. 2002) that interact with suitably chosen *learner and learning objects* (Vossen and Jaeschke 2002, 2003); these processes can be broken down into suitably chosen components which can then be realized as services individually (Vossen and Westerkamp 2003).

Building a decentralized system by composing Web services to achieve functionality similar to that of a traditional e-Learning system clearly leads to a variety of novel challenges, among them that of *managing the content for the learner*. Indeed, in a distributed system organization learning objects cannot simply be imported into a particular learning management system. Instead, content needs to be stored on distributed servers and be called on demand. This paper will show how these aspects can technically be combined

with recent standardization efforts that aim at content exchangeability and efficient reuse, and how it is even possible to exchange content for courses vastly "on the fly." Our repository for learning object publication and search essentially adapts the UDDI framework also used for commercial Web services (Newcomer 2002) to an e-Learning context. Its main features are that the repository itself contains centralized data *about* learning objects, i.e., all meta-information, while the actual content that it refers to can be arbitrarily distributed. We are thus able to tackle some of the problems arising in the realization of a service platform, including

1. storing learning content in a distributed fashion, and
2. dynamically exchanging content when necessary or appropriate.

Using LearnServe, content can be published and organized for exchange, and content can be accessed in a service-based environment. We also discuss some of the problems arising from storing data on different servers, including quality of content, availability of content, and security problems.

Once a platform such as LearnServe is in place and ready to operate, the usage of Web services will enable the integration of e-Learning functionality directly to business applications (e.g., CRM[1] and ERP[2] systems), since it will become possible to directly interact with applications, processes, and other information sources. This could provide benefits for a number of learners particularly in secondary and tertiary education, who are mostly following a learning-on-demand approach driven by their professional needs. Indeed, in a society where on the one hand it becomes more and more common to change jobs several times during a work life, and service provision based on the Web becomes more and more mature on the other, it is more than feasible to bring these two developments together so that one can benefit from the other, and flexibility for the learner is supported as far as current technological developments allow. We emphasize that the system development reported in this paper is not intended as a replacement for any form of electronic learning scenario. Moreover, even for an on-demand learning application it might not be the only choice available. However, as technology advances towards Internet2, and as Internet access and computing devices become more and more ubiquitous, the flexibility offered by a Web service approach to learning will become attractive for a growing number of people.

The organization of the paper is as follows: We first describe the basics of e-Learning as Web service in Section 2. In Section 3 we present the architecture of LearnServe as a result of the decomposition of learning-related processes, and we discuss the challenges that need to be met for realizing such a platform. In Section 4 we show how content can be organized for exchange, how content publishing can be done, and how content access can be managed in a distributed platform. Section 5 concludes with a discussion of some of the problems that deserve further study.

2. Exploiting web services for electronic learning

In this section, we describe the basic assumptions and ideas behind the creation of an e-Learning system based on the Web services paradigm. Essentially, we need to distinguish the learner (or client) side and the provider side, where the latter includes all the functions of a learning system other than those pertaining to learners. We discuss each side individually in the following subsections. For both we focus on content aspects since we later want to illustrate the handling of content in a distributed system.

As has initially been discussed by Vossen and Westerkamp (2003), in an e-Learning system a variety of features and components can be perceived as processes and consequently be realized as atomic or composite Web services; examples include content authoring, content configuration into classes or courses, learning object management, content updating, learner registration and management, content adaptation, learner profiling and tracking, testing of acquired knowledge, tutoring, virtual classroom setups, organization of chat rooms, and last, not least, the search for and presentation of content itself. Thus, we imagine that the entire functionality of an electronic learning system is decomposed into individual activities or groups of activities which can be implemented independently and offered as services, in such a way that the original functionality can be "reconstructed" through a suitable service composition. Notice that this is an application of the core Web services paradigm as described, for example, by Alonso et al. (2004) or by Newcomer (2002), to the area of e-Learning.

[1] Customer Relationship Management
[2] Enterprise Resource Planning

In such a scenario, all learning objects, classes, and courses may be stored on different servers, and they need to be registered in a central repository together with meta-information. An individual learning object (i.e., content) is not stored in this directory. To "use" a class, the underlying service platform needs to call the desired learning object, which is then accessed by a presentation service and delivered to the learner. The particular Web service used depends on the metadata, which should be provided by the author of the course and which should fit the profile and preferences of the learner (see below). Figure 1 shows the service subsystems that we will later describe in detail. In particular, we make a distinction between three kinds of services:

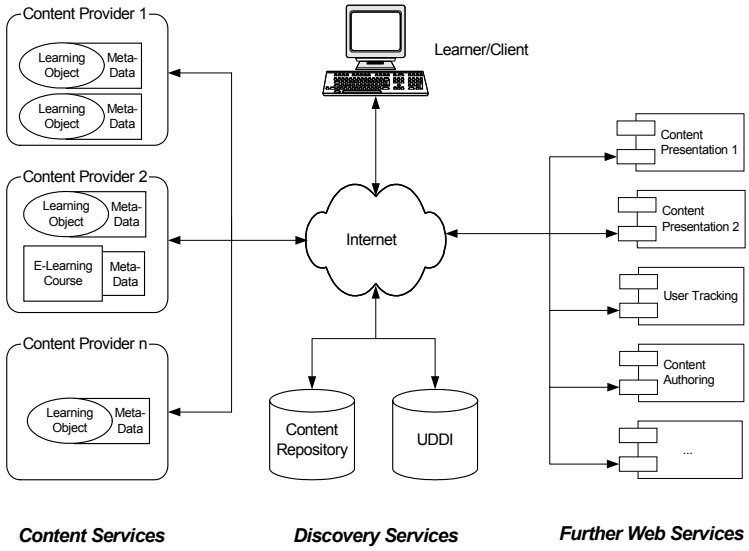

Figure 1: e-Learning as a web service.

1. *Content Services* provide the learning material in form of learning objects, courses or classes.
2. *Discovery Services* are used to search for content (content repository) as well as for additional functionalities that can be added to the system (UDDI).
3. *Further Web Services* can implement a huge variety of functionalities. This can include typical e-Learning activities and third party services that are worth to be consumed by both learners and teachers. These services also encompass payment and certification services.

We mention that the choice of subsystems we discuss here is not exhaustive, and that various additions may be feasible. We also mention that this architecture allows for a variety of implementation choices, i.e., which part of the system is implemented in the central platform that is used by the client and what parts just call upon external or remote Web services.

2.1 Provider side

The provider side is split into different sections that can be handled by individual services. There are *authors* who create learning objects (LO), authors who build courses or classes from such objects, and *trainers* who communicate with learners. Authors creating content do this for a specific group of people or just for an anonymous circle of learners. The first step in building learning material is to create individual learning objects which may afterwards be configured into classes and courses, as we have described in Vossen and Jaeschke (2002, 2003). We assume that authoring tools are made available as appropriate Web services, so that an author can choose between different services to select the one which is best suited for his or her situation. At the end of a content creation session, an author registers the new content in a content repository; in addition, the LOs produced are stored on a selected server of the content provider (see Figure 1).

The creation of classes and courses can even be done by users or persons who are not themselves authors of LOs. To do so, they use existing LOs from other suppliers and combine them into a class or course. This creates a kind of added value by plugging the LOs together and cutting development time. New classes are also stored on a server and registered in the repository, without storing the LOs again, because the latter are reused from the

publisher. Even the action of publishing may be handled by the same Web service as the publication of the LO as described before. The provider side also has to offer Web services to deliver and represent the content to the learner. The presentation of the material depends on the technical requirements of the LO and is not discussed in detail here.

Special services, also on the provider side, need to be available for handling trustworthy actions, such as the collection of payment from a content user or the transfer of royalties to the respective content author, the certification of classes for special exams, or the storage of user profiles. Even the execution of tests and exams or the tracking of a user can be handled by corresponding services. A tracking service is designed to check for completion of assignments, determination of degrees if applicable, and updates to the learning allowances or charges in the account of the learner. Importantly, all these services can be implemented as Web services. As shown in the example of the Petri net (Reisig 1985) in Figure 2, content certification is typically a strictly defined process that includes searching, reviewing, and certifying of specific material. Each of these sub-processes can straightforwardly be implemented as Web services (encircled actions). Afterwards these services can be a composed to a complex Web service "Content certification".

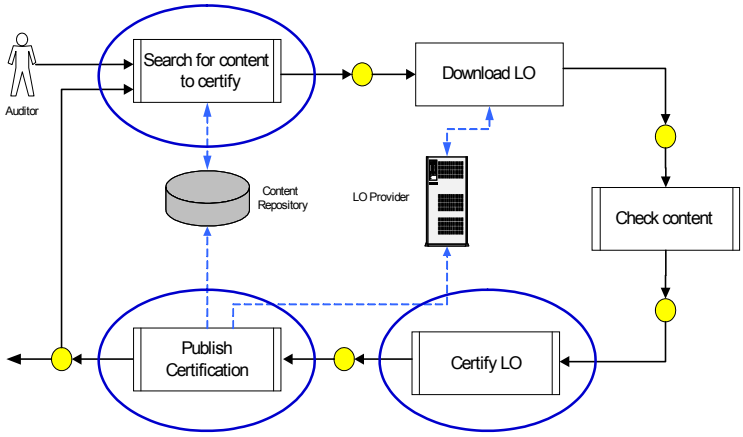

Figure 2: Web services in a content certification process.

Note that the "syntax" of a Petri net as the one shown in Figure 2 is such that rectangles represent activities which may be atomic or composite, and that circles represent "states" of the system (which may be initial, intermediate, or final states) or documents that are consumed or produced. For example, the "Certify LO" activity, which is a composite one (indicated by the double bars), has as input one or more LOs that have been checked already and that are now ready for being certified, and that has as output the certified LO or LO collection. Additional processes of this type have been described by Vossen et al. (2002) or Grüne et al. (2004).

2.2 Learner side

As mentioned earlier, different kinds of learners will want to access and use an e-Learning system with various motivations and perspectives. In a Web-based system the learner ideally just needs a Web browser to use the system, and he or she does not have to bother about what part of the platform is part of the server system and what part is actually a Web service that gets included from a remote site. A personal login ensures that a profile can be created for each user, in order to adapt the system to a user's preferences.

A learner usually logs on to the system with a clear intension of what to learn. Often a learner has already been assigned a course and can start working on the material right away. In our vision of future learning scenarios, the typical learner searches a content directory and "orders" learning objects, classes, or courses that match his or her requirements. Such a search may be driven by needs, prerequisites, budget, client hardware and software, preferences, age of material, author/provider, and the profile of the learner. Upon presentation of search results, a learner can choose and "book" the content he or she wants to use or consume. Suppose the choice is a class which is generally composed of several learning objects that will be presented to the learner in some sequence. The learner just uses the presented material as it would be in a

centralized runtime system of an LMS. In fact, each presentation of material technically consists of a search in the content repository, a call of the learning object, and a call of the corresponding Web service to present the material.

During a learning session, the executing platform has to decide whether or not a learner has passed given test sections and, based on the outcome, whether and which learning objects to present next. If the learner fails such a test, the system has to decide whether to repeat the presentation or to switch to an alternative, where the latter can be done by a dedicated Web service based on the content, the learners' preferences and the authors' prerequisites as mentioned above.

Figure 3 shows a high-level Petri net in the same syntax as before where possible application areas for Web services in the booking of content are again circled. The search for content is done by a search service, which in turn uses another service for querying the user profile and for selecting an appropriate learning object for a learner. Since not every such object can be used without charge, a payment service is also included in the process. At the end, a special service is used for updating the user's profile. Clearly, also the activity of learning can be associated with various services.

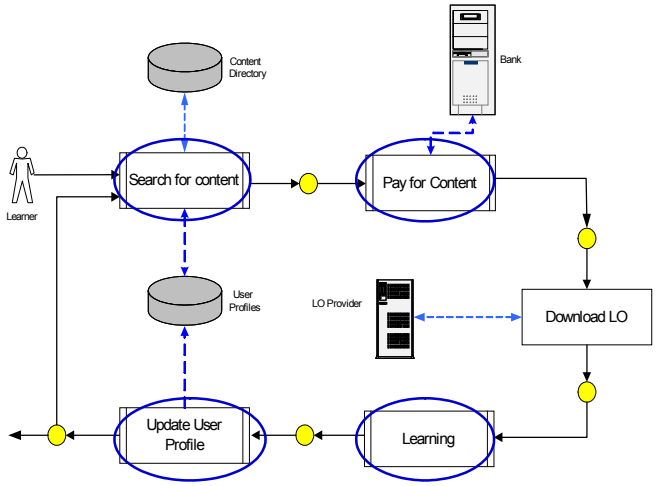

Figure 3: Web services for content booking.

Learners typically need assistance during their work on the material. Help can be provided in an e-Learning system using asynchronous techniques like email or message boards or by using synchronous techniques such as chat with a tutor or video conferencing with virtual classmates. Clearly, all these functionalities can be included in an e-Learning environment by calling upon respective Web services of special vendors, as is currently done in business environments for virtual meetings and video conferencing; details are omitted.

3. LearnServe: Making learning offerings available as services

In this section, we provide an overview of *LearnServe*, a system under development at the University of Muenster; specifically, we show how its functionalities are identified to be built as Web service.

Clearly, it is a crucial design decision which part of a platform should be "outsourced" and hence be included as a Web service from an external provider and which part is not. We mention that a realization of an e-Learning platform as a collection of Web services can basically use an existing service platform and its development tools (e.g., HP Web Services Platform, Microsoft .NET, Sun ONE, BEA WebLogic Enterprise Platform, IBM WebSphere). However, as the discussion about "standards" in this area is far from converging, we are currently experimenting with our own prototypical implementation that grew out of our XLX learning platform (Hüsemann et al. 2002) as well as out of e-Learning workflow studies done in the context of the INCOME Teacher project (Vossen and Jaeschke 2002, 2003; Vossen et al. 2002).

As has been indicated, the idea behind *LearnServe* is to take the functionality of an e-Learning system apart, specify its major

components and activities as processes (cf. Figure 2) that can be executed as workflows, and group the result into atomic and composite Web services, for which UDDI as well as WSDL documents are then prepared. The prototypical implementation independent of a commercial platform, which is discussed next, renders it possible to study various aspects specific to learning environments and scenarios; its overall architecture is shown in Figure 4.

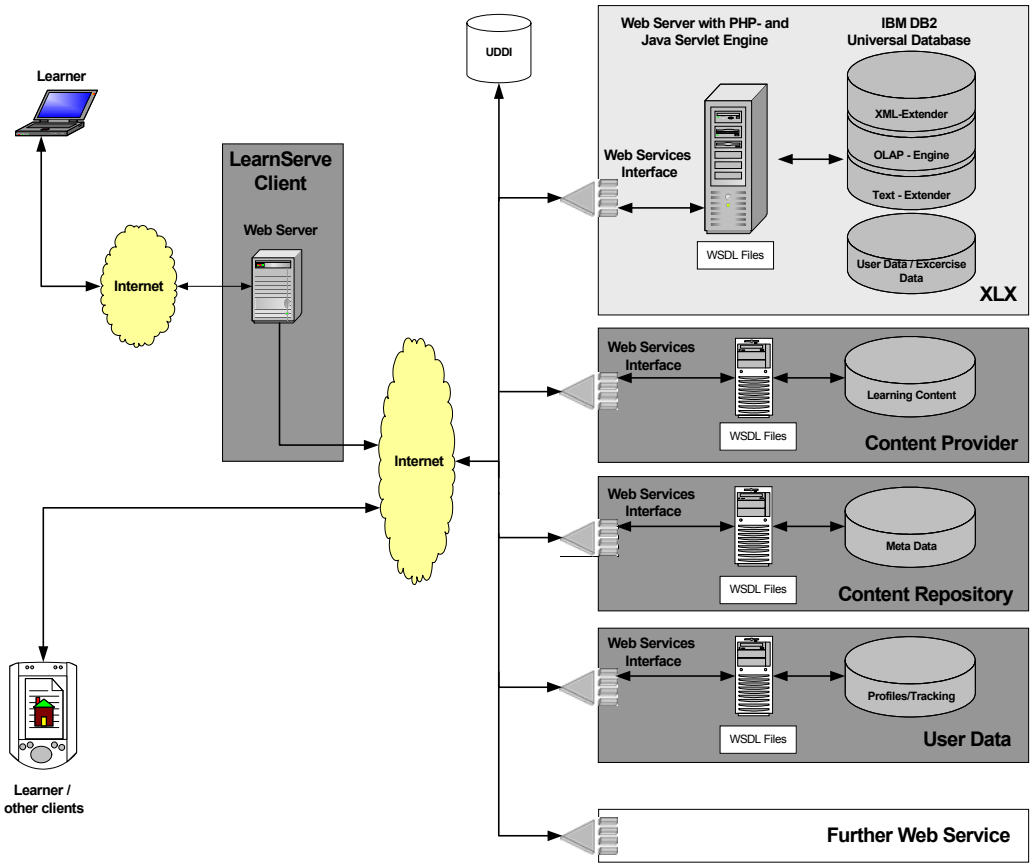

Figure 4: Architecture of LearnServe.

From a logical point of view, LearnServe is divided into two parts: a client software and Web services provided by several vendors or providers. All core components of LearnServe are gray shaded in Figure 4. The LearnServe client is the web-based “access point” for a user that enables her or him to consume and use the learning services. These services are implemented on distributed servers and in particular include authoring, content, exercise, tracking, and discovery services as well as communication services such as email and message boards. Usage of these services is not limited to our own clients, because the implementation of the entire functionality as Web services enables an integration of the e-Learning functionality directly into any business application, in order to interact with applications, processes, and information. The learning Web services can also be used on mobile devices if there is an appropriate client for that device.

As mentioned, the LearnServe system is based on XLX[3] (Hüsemann et al. 2002), which has already been used by different German universities and other customers to train graduate level students in various scientific courses. XLX is implemented using a typical three-tier architecture consisting of a client, an application, and a data layer, resp. Students need an Internet connection and a Web browser to access the system, but no special client software or plug-ins. On the server side, XLX uses an Apache Web server with PHP as well as a Java servlet engine, and stores all necessary data in an IBM DB2 UDB database or in a system with similar functionality. To include third-party systems (such as an XSLT or an XQuery processor) for training purposes, XLX also provides an external interface.

For LearnServe, XLX is currently being enhanced for Web service support (see light

[3] See http://dbms.uni-muenster.de/xlx for more information as well as a guest account.

gray shaded box in Figure 4) to reuse the already implemented exercise and training sections. XLX now serves as a provider of Web services to be used in our LearnServe client. All services are registered in a UDDI directory which provides all necessary information to use the functionality of the remote service within our platform. If one of these internal Web services of XLX is called, the corresponding listener module recognizes this and executes the service. Upon execution, the services are able to communicate with different data sources to read or write the information as needed to (e.g., learning objects or data of a user profile).

Additionally we are implementing services for the usage of learning objects in the LearnServe client. These are based on the one hand on the well established SCORM standardizations and on the other hand on the already mentioned Web services standardizations. Of course this content can be combined with the exercises provided by XLX to build comprehensive courses.

The strict modeling and analysis by cutting the process models in parts delivers a detailed to-do list of what to implement as Web services. In the next section we will focus on the services and requirements in the area of content usage and offering and the potential that service orientation provides.

4. Content management in a web services based system

This section is to show how learning objects are handled in a Web services based system, with both advantages and disadvantages. Several problems newly arise in a distributed system when compared to a centralized learning platform, which require specific solutions.

Content for e-Learning can be composed of miscellaneous items such as text, pictures, videos, animations, diagrams, XML and HTML files, etc. Moreover, content should be designed in a way so that it can be exchanged between different learning management systems, in order to enable efficient reuse. To this end, the *IMS Content Packaging Information Model* (IMS 2001) specifies a technique for storing and exchanging learning object content using XML and a suitable bundling of files and is part of the SCORM standard (ADL 2001). There are three steps in preparing given content for being exchanged: First, a so-called *Manifest* document, which is an XML document, is created; this document is then validated in the second step. Finally, the XML document is bundled with physical files which contain the actual content to form a *package*. Thus, a package is a logical directory and consists at least of two major elements:

- a special XML file (the Manifest, always called "imsmanifest.xml") describing the content organization, metadata and recourses in a package in conjunction with any XML control document that it references, and
- the physical files being described by the Manifest file organized in subdirectories.

This IMS Package represents a unit of (re-) usable content and can be combined or composed with other packages; for an easy exchange and storage it can be incorporated into a single file (using standard archive formats, e.g., zip, jar, or cab), too, which is henceforth called a "*Package Interchange File*" (PIF) as indicated in Figure 5. PIFs can be sent over the Internet or be exchanged via CD-ROM.

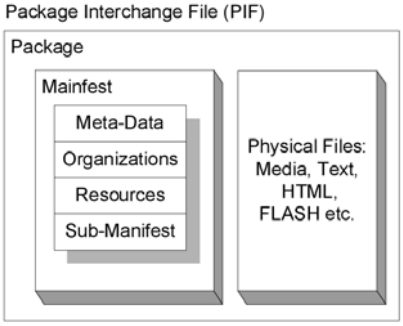

Figure 5: Logical structure of a Package Interchange File.

The Manifest can include an identifier for labeling purpose and a version which refers to the IMS Content Packaging version number. As seen in Figure 5, the Manifest is subdivided into four parts which contain a *meta-data* section, an *organizations* section and a *resources* section as well as an optionally additional *Submanifest*:

- The metadata of the learning content in the *meta-data* section is embedded into a metadata tag and follows the specifications of the IMS Meta-Data Information Model (IMS 2001). The metadata describes the Manifest (and thus the learning object) as a whole.
- The *organization* segment contains information describing the structure of the content in the package, e.g., the table of contents or a custom structure.
- The *resources* reference the actual content. This can be a physical file in the

package or even a reference to another Manifest file. Each resource can be additionally described by metadata following the specification of the IMS Meta-Data Information Model (IMS 2001).

- Any nested *Submanifest* describes the content at the level to which it is scoped, such as a course, instructional object, or other.

The objective of the IMS Content Packaging Information Model is to define a standardized set of structures that can be used to exchange content. These structures provide the basis for standardized data bindings that allow software developers and implementers to create instructional materials that interoperate across authoring tools, LMSs, and runtime environments that have been developed independently. Further information can be added to the structure of a Manifest document, like *IMS Simple Sequencing* (IMS 2003) for defining the behavior and a set of rules for content selection within a course.

Since content cannot be stored in a central place in our distributed environment, we provide a repository that is based on the information provided in the Manifest documents of the PIFs. The repository is made up of several Web services to use its functionality from a remote system, as well as a relational database to store the information about a learning object. The object itself is not stored in the repository; instead, a cross reference is provided in the repository which points to its actual location. For example, this location can be a Web server of the author which has a unique address. The main idea when publishing a learning object is that a Web service can be employed to read the object's metadata of its Manifest document and to store this information in the database provided by the repository.

Notice that this approach has several advantages: For one, calling the described actions can be made very flexible using a dynamic call of Web services, such as the one introduced by Keidl et al. (2002) and described for an e-Learning scenario in Vossen and Westerkamp (2003), by querying the UDDI directory of all available services that are based on the same technical model; in addition, the use of learning objects can become dynamic. Traditional systems have to import learning objects to their own data pool in order to make use of them. This makes learning object availability dependent on the local database system only, at the expense of limiting reuse (in particular in other learning communities) and exchangeability. Using a central content directory enables authors to reduce the workload of creating courses by reusing content from other authors. Because content is stored on distributed servers we have to transfer the dynamic call of Web services to the call of learning objects.

As an example, think of an online course to become a database certified engineer (DCE) as shown below in Figure 6. A learner may first look up the requirements, which are published by the database system vendor, then registers for the course, obtains a to-do list, and starts with the first learning object. The system knows how to assemble learning objects of a particular type, thanks to the authors' class (and course) definition. Let us have a closer look at a "schema tuning" object, which is called after a successful processing of the class for "database administration" as well as after the "query tuning" object. The platform triggers the call of the object by using the definitions the author has made while building up the class. In fact, the author has defined different objects for "schema tuning" to be possibly used in the class. Depending on the preferences of the user and on the (time or cost) allowance, the system selects the object that fits the learner's needs and profile best. Hence, different learners can receive different objects on the same topic if working on the same class, depending on their personal data and preferences. Let us now assume the system has chosen an object for submission to the student. Based on the metadata of the object, a Web service is called that computes the optimal presentation of the material to the learner and delivers the result of the computation. This is not a static call either, but a dynamic one for the presentation of the object. Restrictions and preferences of learner, author, and client trigger the choice of the Web service to present the material.

Building a system of dynamic selection of learning objects has the advantage of an increased flexibility and adaptability. A learner does not even have to notice an exchange of learning objects for the purpose of updating or adapting the course material to his or her needs, or for exchanging content in case of a learning object being offline and hence unavailable. Critical, however, are the facts that reusing learning objects and storing data about them in a central content directory require a use of generally accepted standards. Today, many e-Learning systems do not care much about standards and use their own way of handling content, what makes this material difficult to be reused for a system based on

Web services. This lack of interchangeability can be solved by using Web services to create the content, which are implemented carefully to use corresponding standards; we mention that this is not yet generally the case today. On the other hand, the dynamic call and flexible exchange of learning objects leads to the problem of being able to evaluate and compare them. To use different objects on, say, "schema tuning" in a database course, they must have a comparable and similar content to ensure that all learners can learn the same topics, reach the same level of knowledge, yet do so in different ways. This problem can, for example, be solved "manually" by the author of a course or by a special instance that certifies learning objects. The latter has the advantage that certified content can be used to confer degrees after passing an (online) exam.

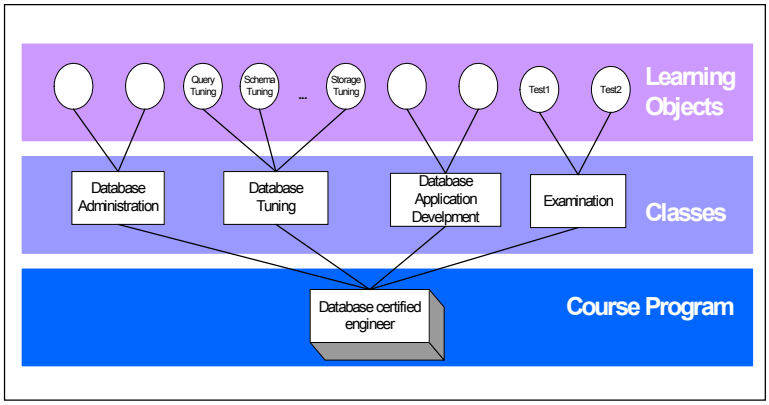

Figure 6: Sample DCE course program.

In addition to the dynamic exchange of learning objects during ongoing learning activities, an author can exchange an object without changing the definitions of the respective course. The main advantage of distributed storage of content is an easy upgrade and correction of content, because the author just has to replace special (commonly just small) parts of the package without changing anything in the repository, if the metadata of the content remains unchanged. Clearly, the repository has to provide a versioning of learning objects since major changes may confuse learners. A disadvantage is that potentially dead links are stored in the database. To resolve this problem, the repository needs to scan stored links regularly to detect registered, but already erased learning objects. The usage of Web services enables developers of an LMS to include the functionality of the repository straightforwardly into their own systems and hence opens them up for a larger diversity of content. This can reduce the authoring of content tremendously, as content can be much easier reused. The learner has a choice of much more content to use - he has now even a choice of similar content of different authors.

Another problem concerns security aspects on the client side. Since learning objects might execute programs on the client's computer, it is difficult to ensure that no attacks will take place to that machine, since normally everybody can build content and provides it for download. Finally, all repositories independent of the type of storage and access properties face the problem of having to verify or at least check the quality of the registered content. To a large extent, this problem can only be solved by human supervisors. The LearnServe repository provides a Web service for reviewers to certify content to allocate LOs of a certain quality to learners.

5. Discussion and conclusions

In this paper, we have described how content can be handled in a Web services based electronic learning platform. Several positive aspects such as a dynamic exchange of learning objects during a learning process as well as an easy updateability of content are provided by distributed storage in connection with a central repository which keeps metadata about the objects. Moreover, a consequent modeling of learner, author, trainer, administrator, and system activities as processes enables us to break down the functionalities of traditional centralized learning platforms into many small components and individual applications that can be "outsourced" and hence be realized as Web services.

We mention that we are not claiming a general replacement of present-day e-Learning systems and scenarios by systems based on

the Web services paradigm. Instead we envision what we have described in this paper as one of several forms in which future learning will take shape. The Web services approach seems particularly suited for learners seeking secondary or tertiary education on demand, due to changing or new job requirements, personal interests or necessities, or other conditions. What has been described in this paper is only a first step in this direction.

We conclude by mentioning several approaches that are related to our work. Sadiq and Orlowska (2001) have introduced an workflow based e-Learning system, where content can be exchanged in the workflow model, and plans can be defined to enable different pathways through the learning content in a centralized system. A similar approach is the COW system under development in France, see Vantroys and Peter (2001), its goal is to develop a workflow based system for e-Learning purposes which is supposed to support personalization and adaptation as well as a sequencing of learning activities. On the other hand, there are several different repositories in the area of e-Learning: iLumina[4] provides a centralized register and references content on distributed servers; however, it is not based on the IMS Content Packaging and on Web services, which makes an integration of content difficult. The LORAX[5] repository provides a *central content management facility* ("The Exchange") to store and publish content. A detailed specification is given how to access this repository via Web services to search for and retrieve learning objects. Learning objects are delivered in the form of PIFs from the central "Exchange". Finally, OLR (Open Learning Repository) follows the idea of not storing content within a repository, but it is not accessible via Web services; for details, see Dhraief et al. (2001).

Among the issues that deserve further studies are on the one hand technical ones, such as the provision of easy-to-use interfaces for Web service composition. Indeed, if a learner is faced with the task of composing a course by herself or himself, this should not have to deal with, say, integrity or plausibility checks; instead, whatever is offered as possible composition parts should indeed be composable. Another study area, from a more conceptual perspective, needs to deal with the provision of pricing models. Here we envision strategies similar to those currently used by telecommunication providers, including flat fees, “call-by-call” fees, or base rates plus fees based on usage. Pricing schemes not only need to consider what learners have to pay, but also, for example, what authors can make in terms of royalties. We expect to report on these issues in th near future.

[4] see http://www.ilumina-dlib.org
[5] see http://www.thelearningfederation.edu.au/

References

Advanced Distributed Learning Initiative (2001). Advanced Distributed Learning Sharable Content Object Reference Model, The SCORM Overview, Version 1.2.

Alonso, G., F. Casati, H. Kuno, V. Machiraju (2004). Web Services - Concepts, Architectures and Applications. Springer-Verlag, Berlin.

Andrews, T., F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, I. Trickovic, S. Weerawarana (2003). Specification: Business Process Execution Language for Web Services Version 1.1, ftp://www6.software.ibm.com/software/developer/library/ws-bpel11.pdf.

Dhraief H., W. Nejdl, B. Wolf, M. Wolpers (2001). Open Learning Repositories and Metadata Modeling. In Proc. International Semantic Web Working Symposium (SWWS), Stanford University, California, USA.

Grüne, M., K. Keferstein, K. Lenz, A. Oberweis, M. von Mevius, G. Vossen (2004). Individualization of E-Learning Processes by Workflow-Based Learning-Object Management. In Proc.\ 7th International Conference on Business Information Systems (BIS), Poznan, Poland, 214-226.

Hüsemann, B., J. Lechtenbörger, G. Vossen, P. Westerkamp (2002). XLX - A Platform for Graduate-Level Exercises. In Proc. 1st International Conference on Computers in Education (ICCE), Auckland, New Zealand, 1262-1266.

IMS Global Learning Consortium, Inc. (2001). IMS Content Packaging Best Practice Guide, Version 1.1.2. August 2001.

IMS Global Learning Consortium, Inc. (2003). IMS Simple Sequencing Best Practice and Implementation Guide, Version 1.0 Final Specification, March 2003.
http://www.imsproject.org/simplesequencing/ssv1p0/imsss_bestv1p0.html

Keidl, M., Kreutz, A., Kemper, A., Kossmann, D. (2002). A Publish & Subscribe Architecture for Distributed Metadata Management. In Proc. 18th IEEE

International Conference on Data Engineering (ICDE), San Jose, CA. IEEE Computer Society, 309-320.

Leymann, F. (2001). Web Services Flow Language (WSFL 1.0). IBM Report.

Newcomer, E. (2002). *Understanding Web Services: XML, WSDL, SOAP, and UDDI*. Addison-Wesley, Reading, MA.

Reisig, W. (1985). *Petri Nets, An Introduction*. EATCS, Monographs on Theoretical Computer Science, W.Brauer, G. Rozenberg, A. Salomaa (Eds.), Springer Verlag, Berlin.

Sadiq, W., M. E. Orlowska (2001). Flex-eL - Managing e-Learning through Worfkflows. In Proc. SAP Research and Applications Congress, February 2001, San Diego, USA.

Vantroys, T., Y. Peter (2001). A WMF-based workflow for e-Learning. In Proc. European Research Seminar on Advances in Distributed Systems (ERSADS), Bertinoro, Italy, May 2001.

Vossen, G., P. Jaeschke (2002). Towards a Uniform and Flexible Data Model for Learning Objects; In Proc. 30th Annual Conference of the International Business School Computing Association (IBSCA), Savannah, Georgia, 99-129.

Vossen, G., P. Jaeschke (2003). Learning Objects as a Uniform Foundation for Database-Driven E-Learning Platforms. In Proc. 7th International Database Engineering and Application Symposium (IDEAS), Hong Kong, China IEEE Computer Society Press, 278—287.

Vossen G., P. Jaeschke , A. Oberweis (2002). Flexible Workflow Management as a Central E-Learning Support Paradigm. In Proc. 1st European Conf. on E-Learning, Uxbridge, UK, 253—267.

Vossen, G., P. Westerkamp (2003). E-Learning as a Web Service (Extended Abstract). In Proc. 7th International Database Engineering and Application Symposium (IDEAS), Hong Kong, China, IEEE Computer Society Press, 242—249.