The Emerald Research Register for this journal is available at
www.emeraldinsight.com/researchregister

The current issue and full text archive of this journal is available at
www.emeraldinsight.com/0969-6474.htm

Aspect-oriented re-engineering of e-learning courseware

Victor Pankratius and Wolffried Stucky
AIFB Institute, University of Karlsruhe, Karlsruhe, Germany, and
Gottfried Vossen
ERCIS, University of Münster, Münster, Germany

Abstract

Purpose – This paper proposes solutions to problems related to the maintenance and update of already existing e-learning courseware.

Design/methodology/approach – A structured approach in form of a reference model for the re-engineering of existing educational material is presented. In this context, concepts already established in the area of aspect-oriented programming are applied to deal with crosscutting concerns in e-learning material. Finally, a software product line approach is proposed for the creation of new courseware using re-engineered components.

Findings – It turns out that some aspects of the methodology developed for aspect-oriented programming can also be used to restructure the existing e-learning material in such a way that maintenance is eased and redundancy is significantly reduced. In addition, software product lines for e-learning material provide a global framework for coordinating the re-engineering and reuse of components.

Practical implications – The advantage of the proposed approach is that existing e-learning standards and systems do not have to be modified or adapted.

Originality/value – Usually, courseware evolves during a longer period of time and its development does not start every time from zero. There is a high incentive for re-engineering of existing courseware, since it constitutes in many cases the competitive advantage of companies or universities. However, up to now, little attention is paid to the maintenance and the efficient update of e-learning material which is already there.

Keywords Learning, Computer based learning

Paper type Technical paper

Introduction

During the last years, internet-based e-learning has achieved a widespread usage due to better access to educational material, decreased cost and instant delivery. Current learning environments give learners the flexibility to learn anytime and virtually anywhere. In general, a typical learning environment is implemented at the moment in the form of a learning management system (LMS) (Adelsberger *et al.*, 2002) which stores relevant learning content and which is able to manage learners and their profiles, record their progress, schedule events, and ease collaboration. A learning content management system (LCMS) additionally provides functionality to edit and modify educational content. However, we will not distinguish further between LMS and LCMS here, since there is a significant overlap of functionality between current LMS and LCMS systems, and we will collectively use the acronym L(C)MS (Pankratius *et al.*, 2004). Educational content itself is usually handled in units of the so-called learning objects (LOs) which basically represent packages to organize, encapsulate, and exchange learning material between different L(C)MS systems. There are several

The Learning Organization
Vol. 12 No. 5, 2005
pp. 457-470

0969-6474
DOI 10.1108/09696470510611401

efforts in the field to standardize the data formats of LOs, however, there is no single dominating standard.

The maintenance and update of learning material and LOs in the presence of progress is a recurring and persistent problem in electronic learning, partly because of the abundance of specifications that have emerged in recent years. Furthermore, systematic approaches are needed in order to prevent learning material from becoming outdated or inconsistent. In particular, one important problem occurring in practical situations is that learning material which conceptually belongs together is scattered across several LOs. This leads to difficulties when such material has to be updated, since changes may have to be made in several independent locations. In this paper, we show a possible solution by taking a software engineering perspective on the problem of updating learning material for electronic platforms, and show how a re-engineering approach can help. Specifically, we show how concepts already established in the area of aspect-oriented programming (AOP) can aid in handling cross-cutting concerns, and we present a reference model for the re-engineering of existing material to ease maintainability.

As an example, consider the situation where an LMS contains, among other material, an online course that deals with databases, and one that deals with software engineering. Very often, database and software engineering courses both teach modeling techniques. In practice it is likely that identical material on modeling techniques (e.g. assessment questions for online tests, slides, etc.) occurs in an LO of the database course as well as in an LO of the software engineering course. This complicates the situation during updates, since the same changes may have to be applied to both LOs. In a situation with many LOs, manual updating is hardly feasible, especially if content has been imported from various sources. However, there are no comprehensive and systematic proposals around so far. Our approach, to be developed in this paper, is to reuse concepts from the area of software re-engineering and AOP to perceive the problem as a process which can even be cast into the framework of a reference model.

The paper is organized as follows. We will first present some details on current standards for LOs and some key concepts of AOP, which will be applied in the context of LOs. Subsequently, we present a re-engineering process model, which shows how aspects can be systematically regained from existing LOs. Finally, we describe how maintenance tasks can be eased during the creation of new LOs by using principles originally developed for software product lines.

Current e-learning standards for LOs

In this section, we briefly sketch some of the standards and recommendations for LOs, with an emphasis on instructional management system (IMS) content packaging. It is important to understand that conceptually there are three basic layers of formats: for packaging information for transport purposes, for meta information about the content, and for the actual content. With respect to what will be described in this section, examples of these formats are: for a transport format XML; for a metadata format IEEE LOM; for content formats ASCII text, XML, HTML, PDF, or PowerPoint.

The Aviation Industry CBT Committee – AICC (2004) defines guidelines for the implementation of computer-based training in a general way so that they can be used in multiple industries. The Dublin Core initiative defines a widely accepted

specification of metadata for the content of digital libraries, but which is also used in e-learning. The standard defines a set of 15 descriptors related to content, intellectual property and document instances. The descriptors are often embedded directly into HTML (via meta-tags), XML, or RDF documents. Further information is given in Dublin Core Metadata Initiative (2004, http://dublincore.org/).

The Learning Technology Standards Committee (LTSC, 2004, http://ltsc.ieee.org/) of the Institute of Electrical and Electronics Engineers (IEEE) has several working groups covering different aspects of e-learning systems, like learning object metadata (LOM), architecture of learning systems, digital rights management, or computer managed instruction. In LOM, a LO is loosely defined as "any entity – digital or non-digital – that may be used for learning, education, or training" (IEEE Computer Society, 2002). The basic metadata structure describes LOs using nine categories of attributes: general, lifecycle, meta-metadata, technical, educational, rights, relation, annotation and classification. One significant problem of the LOM standard is that it allows different levels of "conformity", which can lead to interoperability problems in practical situations.

The committees of the IMS project are developing standards related to learning resources metadata, content packaging and sequencing, question and test interoperability, or enterprise integration, to name just a few (IMS Global Learning Consortium, Inc., 2004). IMS incorporates and extends aspects of Dublin Core. The XML-based IMS content packaging specification deals with "content resource aggregation, course organization, and meta-data" (IMS Global Learning Consortium, Inc, 2003). A package is defined as a standalone unit of reusable content, which has instructional relevance and "can be delivered independently, as an entire course or as a collection of courses". Technically, a package is a directory that includes a manifest, which is an XML element describing the package itself, any control documents (e.g. DTDs) (Harold and Means, 2002), and the actual physical resources (Figure 1). Inside the manifest, a metadata section contains data describing the manifest, an organizations section defines possible course outlines by allowing a multiple membership of resources in different organizations, and a resources section gives details on the needed resources, which can be any type of assets, e.g. web pages, media files, or text files. It is also possible to define resources that act as a container for other resources. Furthermore, a sub-manifest section allows zero or more (optional) nested manifests.

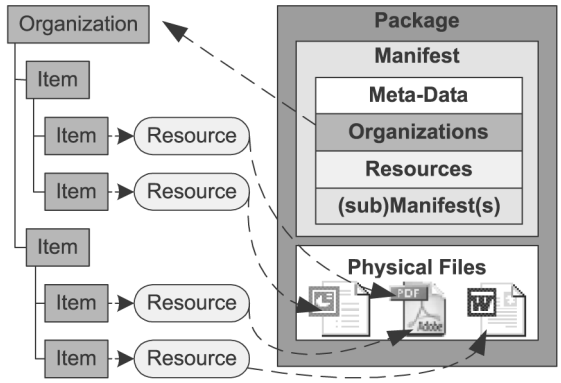

Figure 1.
IMS content packaging

Finally, the philosophy of the advanced distributed learning (ADL) sharable content object reference model (SCORM, 2004, www.adlnet.orgwww.adlnet.org) is to integrate the best of all standards into a single one. Indeed, the proposed content aggregation model (CAM) is based on metadata specifications from IEEE LOM, content structure derived from AICC, content packaging from IMS and sequencing information from IMS.

Aspect-oriented programming concepts

The concepts of AOP are aimed at improving the structure of a system, which may be defined, for example, by a hierarchy of classes when object-oriented programming (OOP) techniques are used. During the development of the hierarchy of classes in the design phase, developers are keen to neatly localize all related concerns in a class and produce a clean hierarchy of classes. However, there may be situations in which concerns and their respective code cannot be assigned to a single class, no matter what decomposition criteria are chosen. At this point, such concerns become the so-called cross-cutting concerns since the code that does not "fit" into a given class hierarchy will have to be broken into pieces that cross multiple classes. In the literature, this phenomenon is called "tangled code" (Kiczales *et al.*, 1997), because although it logically belongs together, it is distributed over several locations. For example, failure handling or debugging code is often implemented for individual classes, so that one cannot find a single coherent program module where all the failure handling or synchronization is done. Thus, updates and changes of such code can be problematic.

AOP tries to mitigate this problem by providing constructs called aspects (Kiczales *et al.*, 1997) which basically modularize and isolate the code of such crosscutting concerns from the rest of the code. Furthermore, an additional phase during the creation of an executable program is introduced, in which the right parts of the aspects are "woven" into their corresponding, predefined program locations. This is done using an aspect weaver which takes the aspects and the program code without crosscutting concerns as input, and which produces regular, executable source code as output. The locations in which aspects will be included in the source code are defined by the so-called join points. Aspect languages are usually used to extend base programming languages to specify aspects and join points.

For example, in AspectJ (Kiczales *et al.*, 2001), an aspect language for Java, an aspect consists of a pointcut which is a collection of join points, and a corresponding section of advice code. A pointcut may contain for example, one joinpoint which matches a call to a particular method name. It can be specified that the corresponding advice code is executed before or after that method is called. Let us suppose that a developer wants to check if some classes work correctly. Without aspects he would add debugging code to each class, which prints the contents of variables before and after the execution. Using AOP concepts in AspectJ he can create a debugging aspect with advice code that prints contents of variables, along with a joinpoint which specifies that it should be executed before and after some specified methods. The interesting point is that even though the aspect weaver makes the necessary code inclusions during the program translation, from a developer's point of view the original program is left untouched, i.e. it remains executable as if aspects would not exist. Therefore, the code maintenance is simplified for the aspects, because coherent code is now found in one location, as well as for the rest of the program which does not contain tangled code parts. For more details on AOP refer to Rashid (2004).

Improving LO maintenance using AOP concepts

The application of AOP concepts to LOs has recently been proposed in Pankratius (2005). As an example, Figure 2 shows a sample organization of the metadata of two LOs in IMS format. As can be seen, online tests can be a cross-cutting concern since their resources are distributed across LOs. In addition, in practical situations it is even possible that resources for online tests contained in one LO are duplicated to other LOs using "copy & paste" functionality, thus introducing redundant parts.

Conceptually, "online testing" may also be a module in its own right. The proposal of Pankratius (2005) is to create an aspect for cross-cutting LO-concerns like testing and use an XML-based aspect weaver to include the respective aspect parts into a LO just before it is delivered to an LMS. Since the IMS format is based on XML, implicit join points can be specified by XPath expressions (Harold and Means, 2002), which identify particular nodes in the XML document. Then, the weaver can be instructed to replace those nodes, or insert XML code before or after them. The advantages are that aspects allow easier maintenance, for example, for all online tests, since the related data will be found in one location, which simplifies editing or removal of redundant parts. In addition, the LOs without the testing aspect will still conform to the IMS specifications and remain "executable" in state-of-the-art LMS. Moreover, the weaving process takes aspects and regular LOs as input, and produces extended, regular LOs as output. Thus, existing LMS do not have to be modified.

We describe next how to attack the problem of creating aspects from already existing LOs by using a re-engineering approach.

Re-engineering educational material

Clearly, the abundance of standards makes interoperability, exchange, and migration of LOs between different L(C)MS very difficult. From the perspective of the competitors which offer e-learning material, there are also few incentives to make the content really interoperable, since they try to build up market entry barriers for the competition, and at the same time exit barriers for current customers, with the result that customers will likely be bound to a certain learning platform. This analysis suggests that it is questionable if the standardization efforts for e-learning will ever succeed. Furthermore, current standards seem to unconsciously assume that once the educational material is

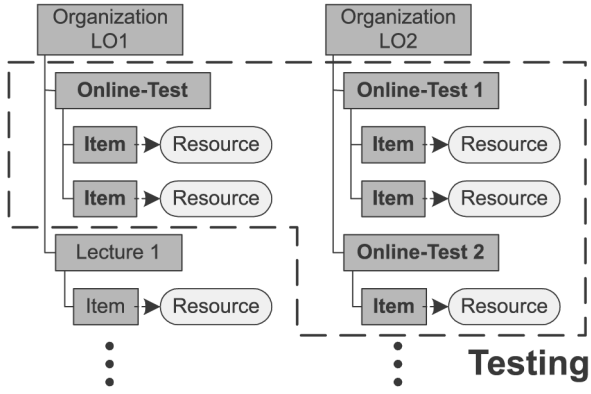

Figure 2.
Example for a crosscutting concern in LOs

"packaged", it will be reused forever without change, which is not at all realistic (the reader might think of a change in curriculum, new online exercises, corrected mistakes, etc.). Another critical point is that current standards are tailored towards the usage of LOs on PCs and neglect developments in M-Learning (M-learning, 2004), which use mobile devices like PDAs or cell phones to deliver e-learning material.

The previous arguments leave not much choice, but to pursue a re-engineering approach for educational material (Figure 3). In Chikofsky and Cross(1990), re-engineering is defined to be "the examination and alteration of a subject system to reconstitute it in a new form and the subsequent implementation of the new form". Furthermore, the authors make clear that for re-engineering, reverse engineering is needed first to obtain some abstract description (Lu *et al.*, 2002), which is then followed by a forward engineering phase that restructures the description. Re-engineering is done when the original system does not meet new requirements, and it is too expensive or impossible to create a new system from scratch. In the context of e-learning, we argue that there is a high incentive for re-engineering of material, because it is unlikely that anybody will start developing lectures, online exercises, and other material (which usually evolved during a long period of time) from scratch, rather than try to recover as much as possible. In our context the re-engineering process will be used to extract aspects from existing LOs and produce regular LOs which do not contain the extracted aspect parts. An aspect (e.g. for online tests) may contain extracted parts from different LOs along with the necessary information to integrate them back. It should be noted that an aspect is on a higher abstraction level than a re-engineered LO, since it cannot be used as a standalone component and since it has to be woven into LOs.

In the next section, we present a process model for re-engineering processes in e-learning, which extends and adapts already proposed models for re-engineering software components (Caldiera and Basili, 1991; Arnold and Frakes, 1992).

A reference model for e-learning re-engineering processes

In this section, we argue that the process of updating learning material, creating new material from old one, or creating aspects can be made well-structured, and we propose a systematic approach in form of a reference model for re-engineering processes in e-learning (Figure 4) (Pankratius and Vossen, 2005).

In general, a domain expert starts by specifying a reusability model, which is used in a component identification and a component qualification phase. The steps in the component identification phase are designed to be executed automatically, so that no human intervention is needed. This automation is a key point in the entire process, since it contributes significantly to the reduction of time and costs to analyze, revise, and extract educational components of interest. The following component qualification

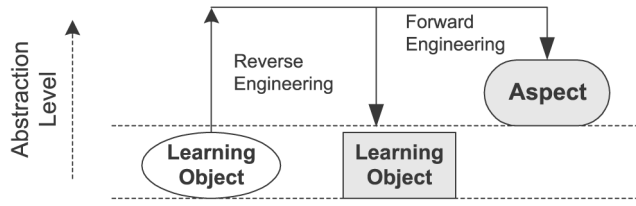

Figure 3.
Re-engineering of educational material includes reverse and forward engineering

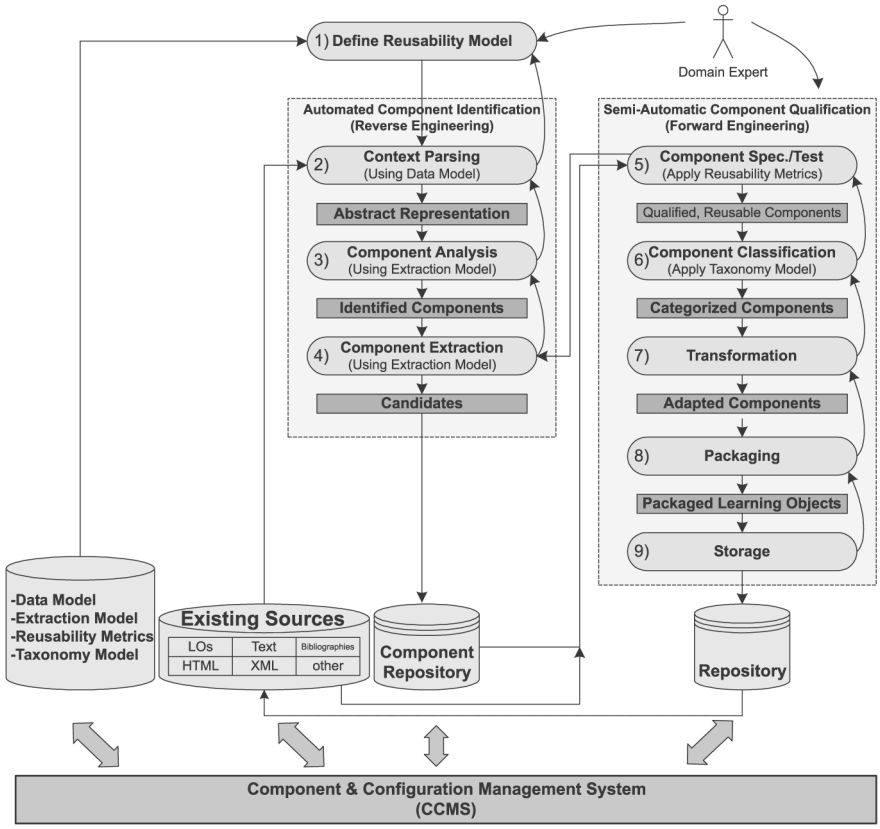

Figure 4.
A reference model for re-engineering processes in e-learning

phase is executed in a semi-automated way and requires interaction with a domain expert. In our view, this part cannot be fully automated, since extracted components from the previous phase have to be validated by human experts. In addition, the extracted components may be changed manually. Therefore, our philosophy is to involve the domain expert and provide as much automated support as possible. The reference model consists of nine steps, which we present next in greater detail. The processing of the respective steps is controlled by a component and configuration management system (CCMS) which coordinates the workflow (Leymann and Roller, 1999; Pankratius and Stucky, 2005) and keeps track of the changes and configurations.

Defining a reusability model

In order to be able to automate some parts of the re-engineering process, a domain expert has to specify a formal reusability model. In our context, such a model is composed of a data model of the sources which contain the material to be extracted, an extraction model which specifies what is to be extracted, a reusability metrics model which is used to calculate metrics based on the data and extraction model, and a taxonomy model, which, based on the data model, is used for the classification of extracted components.

The data model of the sources specifies how the data are organized, and is indispensable for the subsequent steps. As an example, one may consider the previously discussed XML-based standards for LOs, where the corresponding document type definitions (DTDs) or XML schema definitions (XSDs) (Harold and Means, 2002) describe how valid LOs look like, using context-free grammars. Context-free grammars can also be used to specify the structure of other types of material, like, say, BibteX bibliographies which may represent reading lists.

For each particular data model, an extraction model is needed to specify what parts are interesting for extraction. The specifications of the extraction model may range from simple keywords, file types, to complex patterns. If applicable to the underlying data model, queries could also be used for the specification of the extraction model (e.g. using XSLT or XQuery for XML data (Harold and Means, 2002). As discussed earlier, we point out that e-learning material usually not only consists of the educational content itself but also of metadata. In addition, there might be specifications about the presentation of the material on a screen, specifications of possible organizations (e.g. course outlines), assessment items (used to construct online tests), response processing information, or scoring information. From this point of view, there are many different types of data that can be re-engineered and reused. The domain engineer has to delimit in the corresponding extraction models what will later be extracted as aspect and what else as remaining LO. We assume the domain expert to be aware of these nuances and that he or she will define the extraction model accordingly.

Next, a domain expert has to define a reusability metrics model based on the data and extraction model, which is used to measure how "easy" an extracted component can be reused or transformed. Such measures can be used to estimate the required effort and to produce rankings for the extracted "candidate" components. For example, content file types of educational content could be used to produce a metric, since binary file types may require some specific authoring tools, while the manipulation of other types that are based on XML could be automated (e.g. by using XSLT). In addition, other metrics like cyclomatic complexity (McCabe, 1976; Caldiera and Basili, 1991) or coupling/cohesion (Patel *et al.*, 1992) can be adapted to measure the complexity of LOs by taking the links and dependencies between other LOs into consideration, which can be found in the metadata. The model may contain several metrics, which can influence each other. However, the metrics need to be aggregated finally into a single one to simplify the ranking process of candidate components. In addition, a threshold for the aggregated metric is defined to indicate, from an economic point of view, if it makes sense to re-engineer a component or not (Caldiera and Basili, 1991; Sneed, 1991).

Finally, a taxonomy model (e.g. an ontology) (Staab and Studer, 2004), also defined by the domain expert, may ease the work and produce a better overview when many components are extracted. For example, keywords found in the content or in the taxonomic paths of the corresponding metadata can be used for categorization.

Context parsing

The specified data model or grammar from the previous phase is used in this phase to parse the source (file) from which data is to be extracted. In a sub-process called lexical analysis, the source consisting of a stream of characters is grouped into indivisible language units called tokens, which are passed to the parser. In general, the parser constructs a tree (called abstract syntax tree) in line with the specified grammar

(Aho *et al.*, 1985). This tree is an abstract representation of the source and can be used for analysis in the next phase. For example, an XML document of an XML-based e-learning standard can be represented as a tree using the document object model (DOM) (Harold and Means, 2002). As mentioned in the previous subsection, different types of data can be extracted, which requires a flexible adaptation of the parser. As described earlier, in general there are several layers of embedded formats (e.g. for transport and metadata) which have to be crossed in order to get to the actual content.

Component analysis and extraction

Using the abstract representation of the source created by the parser in the preceding phase (i.e. the tree), the extraction model can be applied to find the components of interest which are to be extracted. In the case of aspects, the extraction model simultaneously represents a specification of join points, since it contains the information of the location of pieces which will be extracted.

The pieces identified are forwarded to the phase of component extraction. In this phase, the components that matched the specifications of the extraction model are copied from the sources and stored individually in a component repository. This "distillation process" produces candidates that are of interest for re-engineering. However, some unwanted components might also be stored in the repository because they accidentally match the specifications. During the next phases, it has to be determined in a semiautomatic way which components are of interest, and for those components which are of interest, if it makes sense to re-engineer them rather than start developing from scratch.

Component specification and test

This phase requires intensive interaction with the domain expert, who is presented at the beginning with a list of candidates obtained by the automated component identification. For each candidate, the metrics model is used to calculate an aggregated score, which gives a first hint if it is easy to re-engineer a candidate. Based on these candidates, the domain expert is then able to create the final components that will later be aspects or new LOs, along with their specification. During this process, the domain expert can also aggregate different candidates to form a new component. For example, different pieces of aspects, like extracted parts of online tests from various LOs, can be aggregated to obtain one single aspect which constitutes one module, e.g. for all online tests. Similarly, different parts of LOs can be used to produce a new LO. Based on the extraction models and the change history, the CCMS stores the information from which sources and which locations the parts of the newly created components come from.

After the specification process, the domain expert has to test if components are still valid with respect to the data model of their source, which can be checked automatically. The output of this phase consists of components which are qualified for reuse, but which are not yet readily deployable. Note that the reference model allows a domain expert to go back to the first step and redefine the specifications, if they turn out to be inadequate.

Component classification and transformation

In the previous phase many reusable components may be produced. Therefore, it is desirable to categorize them in order to help domain experts keep track of the components.

The taxonomy model is used to deduce automatically to which category a component belongs. There may be several types of categories, i.e. if a component is an aspect or a LO, the possible application domain, etc. Subsequently, the domain expert checks the results and makes corrections, if necessary.

There are situations in which components are not intended to be reused right away after the classification phase, because they have to be updated (e.g. content of lectures). These updates represent transformations that can be performed manually (e.g. modify PowerPoint slides) or, if possible, automatically with tool support (change colors and copyright year in all HTML pages of a lecture). If the reusability metrics and thresholds are specified correctly, the transformation effort should be minimal. If it is not the case, the model should be updated. The adapted, transformed components are passed over to the next phase.

Packaging and storage

The re-engineered components need to be embedded "backwards" into a metadata and transport format, so that they can be redistributed again. This phase supports the domain expert performing this task. If necessary, input from the domain expert is requested, for example, to add metadata. In essence, this phase transforms the re-engineered components into regular LOs or completed aspects. For aspects, the join point information has to be added from the CCMS. For LOs, it is important to notice that this phase not only allows storing LOs using common standardized formats but also in other ways, like web services (Vossen and Westerkamp, 2003; Pankratius *et al.*, 2004; Vossen and Westerkamp, 2004). In addition, LOs can be adapted to work in various environments for which the current standards do not provide a satisfactory specification yet, e.g. mobile devices.

In the final phase, the packaged LOs as well as the corresponding aspects are stored to the (web-) repository or location where they are to be used. However, during this phase other deployment tasks may also be executed. For example, if a LO is deployed as a web service, additional WSDL files may have to be delivered, and appropriate entries may be inserted into UDDI registries (Casati and Dayal, 2002), so that the service can be found by others.

A product line strategy for reuse of educational material

In the previous sections we have motivated why re-engineering of educational material makes sense, and we have presented the details of a reference model needed for re-engineering.

From a global point of view, a strategic model for the reuse of educational material is needed which embeds our re-engineering approach (Figure 5). The systematic re-engineering of educational material discussed so far (Layer 3) provides a base for further reuse. In general, such a reuse may be *ad hoc* (Layer 2), i.e. it may be uncoordinated and not planned in advance. Although a systematic approach to the recovery of educational material may reduce development costs for new material, other potential cost savings are not realized due to the lack of global coordination. A product line engineering approach for e-learning (Layer 1) attempts to fill these gaps.

In general, a software product line (sometimes also called a product family) is "a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from

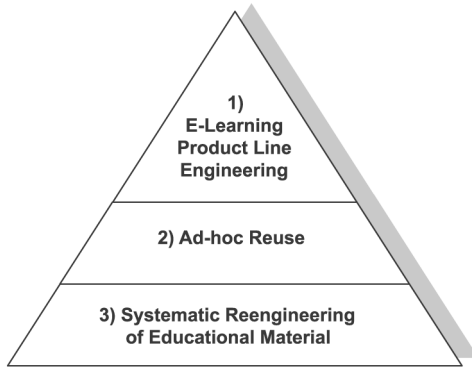

Figure 5.
Strategic layers of re-engineering and reuse for e-learning material

a common set of core assets in a prescribed way" (Clements and Northrop, 2001). Software product lines borrow ideas from traditional manufacturing product lines to reduce development costs (Bockle *et al.*, 2004). Basically, a product (i.e. a software system) is constructed by assembling a set of predefined components (i.e. core software assets) drawn from an asset base, using predefined variation mechanisms. Such a construction method is particularly applicable for software if the products share some commonalities. Thus, the development focus is shifted from programming to assembly.

Three essential activities then play an important role in software product lines (Northrop, 2002): core asset development, product development, and management. The core asset development aims to provide the basic components for further products. They adhere to predefined constraints and requirements that are valid for the entire product line (Kang *et al.*, 2002). The definitions of core assets have far-reaching implications for the product line scope, because they determine the structure of the assembly process and the types of products that can be eventually built. The core assets can be created from scratch, bought from external vendors, or extracted from current software systems using re-engineering methods. During product development, the core assets are assembled to create a particular system according to the requirements of that system. During the development of both core assets and products, management (of the organization) should be actively involved in the process. Management has to provide not only technical, but also organizational support to ensure that every person receives the appropriate resources at the right time.

The proactive approach of software product lines which plan and coordinate re-engineering and reuse, can also be pursued in the e-learning context to manage diversity (Figure 6). The figure depicts the reuse processes for e-learning material from a global perspective: the family engineering dimension includes management processes which define an overall strategy for a product line of e-learning courses by specifying requirements common to several LOs. For example, such common requirements may be specified for all lectures in a database course, all courses in a university curriculum, or all curricula of several universities participating in an exchange program. The domain engineering dimension is concerned with core asset development and focuses on the creation and storage of LOs and aspects in a repository. Our proposed re-engineering process model (Figure 4) is a critical part of this dimension, since the repository of LOs and aspects represents a starting point for the assembly of e-learning courses in the product line.

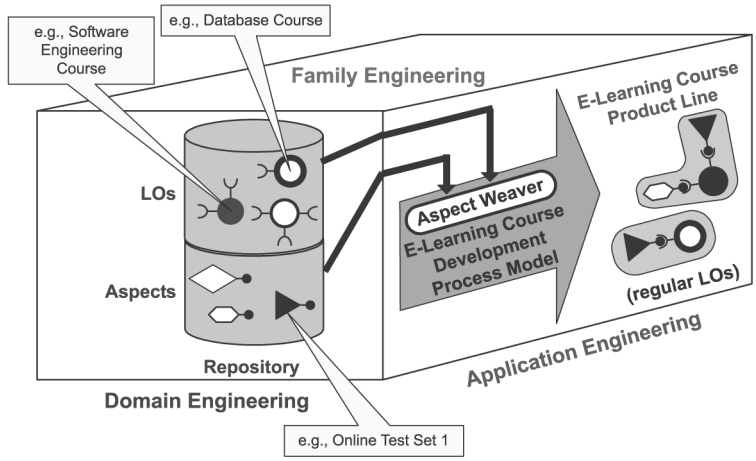

Figure 6.
A product line approach for e-learning

The difference to traditional reuse approaches (Jacobson *et al.*, 1997) is that the LOs must conform to the global specifications made in the family engineering dimension, and that aspects represent the commonalities of LOs in the context of product lines. Finally, the application engineering dimension is concerned with e-learning product development, i.e. the assembly of LOs and aspects from the repository to final courses in an e-learning course product line. An important tool in this dimension is the discussed aspect weaver, which integrates the code of aspects into appropriate LOs. This way, not only the maintenance of material but also the reuse is simplified since in the product line the potential variability of the produced LOs is reduced. Another advantage is that more than one aspect can be woven into one LO, for example, to add customized material and the corresponding pricing information as metadata. Using the weaver, it is technically possible to achieve a better level of customization for predefined target groups of learners, for example, to calculate individual discounts. Finally, it is also conceivable to weave in metadata for the Semantic web in order to improve the retrieval and categorization of LOs.

Conclusion

In electronic learning, the maintenance of learning material in educational as well as in enterprise environments has been underestimated and vastly overlooked in the past. To attack this problem, the absence of a single dominating standard for the exchange of educational content, as well as the missing economic incentives for interoperability, leave not much choice but to pursue a re-engineering approach. Building on our own experiences, we have presented in this paper an approach which uses concepts already established in the area of AOP to improve the maintenance of crosscutting concerns in learning objects. In addition, a reference model was presented which systematically describes the steps needed to re-engineer existing learning material and extract aspects. Finally, a reuse strategy of aspects and Learning Objects based on software product lines was introduced in the e-learning context to improve the management of common requirements and coordinate reuse.

References

Adelsberger, H., Collis, B. and Pawlowski, J. (Eds) (2002), *Handbook on Information Technologies for Education and Training*, Springer-Verlag, Berlin.

Aho, A.V., Sethi, R. and Ullman, J.D. (1985), *Compilers: Principles, Techniques, and Tools*, Addison-Wesley, Reading, MA.

AICC (Aviation Industry CBT Committee) (2004), available at: www.aicc.org/

Arnold, R.S. and Frakes, W. (1992) in Arnold, R.S. (Ed.), *Software Reuse and Re-engineering*, (1993), IEEE Computer Society, Washington, DC.

Bockle, G., Clements, P., McGregor, J., Muthig, D. and Schmid, K. (2004), "Calculating ROI for software product lines", *IEEE Software*, Vol. 21 No. 3, pp. 23-31.

Caldiera, G. and Basili, V.R. (1991), "Identifying and qualifying reusable software components", *IEEE Computer*, Vol. 24 No. 2, pp. 61-70.

Casati, F. and Dayal, U. (Eds) (2002), "Special issue on web services", *IEEE Bulletin of the Technical Committee on Data Engineering*, Vol. 25.

Chikofsky, E. and Cross, J. II (1990), "Reverse engineering and design recovery: a taxonomy", *IEEE Software*, Vol. 7 No. 1, pp. 13-17.

Clements, P. and Northrop, L.M. (2001), *Software Product Lines: Practices and Patterns*, Addison-Wesley, Reading, MA.

Harold, E.R. and Means, W.S. (2002), *XML in a Nutshell*, 2nd ed., O'Reilly & Associates.

IEEE Computer Society (2002), *IEEE Standard for Learning Object Metadata*, IEEE Std P1484.12.1™-2002, available at: http://standards.ieee.org/

IMS Global Learning Consortium, Inc. (2003), *IMS Content Packaging Best Practice Guide, Version 1.1.3*, June, available at: www.imsglobal.org/content/packaging /index.cfm

IMS Global Learning Consortium, Inc. (2004), *IMS Global Learning Consortium, Inc. – Specifications*, August, available at: www.imsglobal.org/specifications.cfm

Jacobson, I., Griss, M. and Jonsson, P. (1997), *Software Reuse: Architecture, Process and Organization for Business Success*, ACM Press/Addison-Wesley, New York, NY/Reading, MA.

Kang, K., Lee, J. and Donohoe, P. (2002), "Feature-oriented product line engineering", *IEEE Software*, Vol. 19 No. 4, pp. 58-65.

Kiczales, G. *et al.* (1997), "Aspect-oriented programming", *Proc. 11th European Conf. on Object-Oriented Programming (ECOOP)*, pp. 220-42.

Kiczales, G. *et al.*, (2001), "Getting started with ASPECTJ", *Comm. ACM*, Vol. 44 No. 10, pp. 59-65.

Leymann, F. and Roller, D. (1999), *Production Workflow: Concepts and Techniques*, Prentice-Hall, Englewood Cliffs, NJ.

Lu, C., Chu, W., Chang, C., Chung, Y., Liu, X. and Yang, H. (2002), "Reverse engineering", in Chang, S.K. (Ed.), *Handbook of Software Engineering and Knowledge Engineering*, World Scientific, Singapore, pp. 447-66.

M-learning: learning in the palm of your hand (2004), M-learning: learning in the palm of your hand, August, available at: www.m-learning.org/

McCabe, T. (1976), "A complexity measure", *IEEE Transactions on Software Engineering*, Vol. SE-2 No. 4, pp. 308-20.

Northrop, L. (2002), "SEI's software product line tenets", *IEEE Software*, Vol. 19 No. 4, pp. 32-40.

Pankratius, V. (2005), "Aspect-oriented learning objects", *Proc. 4th IASTED Int. Conf. on Web-based Education (WBE2005)*, ACTA Press, Grindelwald.

Pankratius, V. and Stucky, W. (2005), "A formal foundation for workflow composition, workflow view definition, and workflow normalization based on petri nets", *The Second Asia-Pacific Conference on Conceptual Modelling (APCCM2005)*, CRPIT, Vol. 43; will also be available at the ACM Digital Library, Newcastle.

Pankratius, V. and Vossen, G. (2005), "Re-engineering of educational material: a systematic approach", *International Journal of Knowledge and Learning (IJKL)*, Vol. 1 No. 3, Inderscience Publishers.

Pankratius, V., Sandel, O. and Stucky, W. (2004), "Retrieving content with agents in web service e-learning systems", *Proc. of the Symp. on Professional Practice in AI, First IFIP Conf. on Artificial Intelligence Appl. and Innovations (AIAI)*, Toulouse.

Patel, S., Chu, W. and Baxter, R. (1992), "A measure for composite module cohesion", *Proceedings of the 14th International Conference on Software Engineering*, ACM Press, New York, NY, pp. 38-48.

Rashid, A. (2004), *Aspect-Oriented Database Systems*, Springer, Berlin.

Sneed, H.M. (1991), "Economics of software re-engineering", *Journal of Software Maintenance: Research and Practice*, Vol. 3 No. 3, pp. 163-82.

Staab, S. and Studer, R. (Eds) (2004), *Handbook on Ontologies*, Springer-Verlag, New York, NY.

Vossen, G. and Westerkamp, P. (2003), "E-learning as a web service", paper presented at 7th International Conference on Database Engineering and Applications (IDEAS), IEEE Computer Society Press, Hong Kong, pp. 242-9.

Vossen, G. and Westerkamp, P. (2004), "Intelligent content discovery within e-learning web services", *Proc. 1st Int. Conf. on Knowledge Management (ICKM)*, Singapore, pp. 365-75.

Further reading

Aalst, W.V.D. and Hee, K. (2002), *Workflow Management*, MIT Press, Cambridge, MA.

Arnold, R.S. (Ed.) (1993), *Software Re-engineering*, IEEE Computer Society, Washington, DC.

Kodaganallur, V. (2004), "Incorporating language processing into java applications: a javacc tutorial", *IEEE Software*, Vol. 21 No. 4, pp. 70-7.

Virtual Global University (2005), available at: www.vg-u.de