[LHRS08] *Lasse Lehmann, Tomas Hildebrandt, Christoph Rensing, Ralf Steinmetz;* **Capture, Management and Utilization of Lifecycle Information for Learning Resources.** In: IEEE Transactions on Learning Technologies, vol. 1, no. 1, March 2008. S.75-87

Capture, Management, and Utilization of Lifecycle Information for Learning Resources

Lasse Lehmann, Tomas Hildebrandt, Christoph Rensing, and Ralf Steinmetz, *Fellow, IEEE*

Abstract—Throughout their lifecycle, Learning Resources undergo a multitude of processes by being created, used, provided, or reused. However, in order to be reusable, a Learning Resource often has to be adapted for a new context of use. This in turn implies multiple Reauthoring processes being performed on a Learning Resource. During each of these processes, different types of information emerge. When retained, this information can be helpful for the retrieval, authoring, use, or reuse of Learning Resources thereafter. In this paper, the lifecycle of Learning Resources along with the information generated herein is analyzed and a distributed architecture proposed that allows for the capture, processing, management, and utilization of the information mentioned in a generic way. Three steps have been conducted to implement the proposed framework. First evaluation results are promising.

Index Terms—Context metadata, lifecycle metadata, relations, information storage, information search and retrieval.

1 Introduction

It is a widely accepted fact that Learning Resources must be reused in order to be efficient. A Learning Resource is a digital resource used for learning (see [40] for details on our definition of Learning Resources). During the last few years, many efforts were made to facilitate the reuse of Learning Resources. Learning Object Repositories like ARIADNE [1], [12] and MERLOT [29] were built and filled with content, tools like ALOCOM [45], the Content Sharing Module Editor [32], or the ResourceCenter [20] were developed, and frameworks and data models were designed [30], [44] to ease the reuse and repurposing of Learning Resources or parts thereof. One major finding of the research done in this area was that the reuse of a Learning Resource as it is, i.e., unchanged or not adapted, is seldom possible [48]. Learning Resources are created mostly within a specific context and with high granularity. From a didactic point of view, this clearly makes sense [2]. However, the reuse of these resources is quite difficult. Usually, it is inevitable to edit the Learning Resources, e.g., in order to overcome the so-called mosaic effect [22]. Mostly, it is necessary to change or remove parts of them, add parts of other Learning Resources, update them, or adapt the Learning Resources according to new requirements in order to reuse them [13], [40]. All these actions are subsumed under the concept of Reauthoring, which is defined, described, and classified in [40]. Through the course of Reauthoring processes, a multitude of information about the resources involved in these processes emerges. In particular, the relations that result from Reauthoring processes have not been considered by most existing approaches yet. The adaptation or Reauthoring of a Learning Resource is usually seen as the beginning of a new lifecycle for a new Learning Resource, which makes sense. However, in most cases, the relations that connect both instances are not taken into account. Nevertheless, there are many situations whereby such additional information is helpful.

It is a fact that users, especially authors, do not want to create or edit metadata and information about their Learning Resources themselves [13]. There are several approaches and frameworks that try to automatically generate metadata for Learning Resources (e.g., [28]). The efforts in this area show that it is important to automatically generate information about Learning Resources to an extent as great as possible. Therefore, it is important that the information that emerges during the processes mentioned above is captured without a user having to actively interact or interfere with the capturing process itself. As information emerges following the actions taken by a user or author, we need an instance that captures the information in the background. Nevertheless, the captured information has to be organized and stored in order to be processible. Most existing systems do not support the capture and storage of lifecycle information sufficiently, and if so, the information gets stuck at system borders. Most authoring tools especially proprietary ones like MS Word or MS PowerPoint do not care about the capture of lifecycle information at all. In some cases, repositories or marketplaces count the number of downloads or views for ranking purposes. However, this information is kept within these systems. Most Learning Management System capture information about the consumption of Learning Resources by learners, but the possibilities to utilize this information in other systems are quite sparse. In this paper, lifecycle information is analyzed, and a system for its capture, management, and utilization beyond system borders is proposed.

In Section 2, we present the approaches and applications that deal with related issues, which show how significant research interest in this area is. In Section 3, our definition of a Learning Resource's lifecycle is presented and the

• *The authors are with the Multimedia Communications Laboratory, Department of Electrical Engineering and Information Technology, Technische Universität Darmstadt, Merckstrasse 25, 64283 Darmstadt, Germany. E-mail: {Lasse.Lehmann, Tomas.Hildebrandt, Christoph.Rensing, Ralf.Steinmetz}@kom.tu-darmstadt.de.*

Manuscript received 20 Mar. 2008; revised 4 July 2008; accepted 16 July 2008; published online 28 July 2008.
For information on obtaining reprints of this article, please send e-mail to: tlt@computer.org, and reference IEEECS Log Number TLTSI-2008-03-0022.
Digital Object Identifier no. 10.1109/TLT.2008.9.

information that is generated when a Learning Resource proceeds through this lifecycle is analyzed and identified. Section 4 depicts the different possibilities for the utilization of lifecycle information with corresponding sample scenarios. Section 5 addresses the storage of the lifecycle information and proposes an extension of the well-known Learning Object Metadata (LOM) standard [21], which stores lifecycle information with Learning Resources in an integrative way. Additionally, a schema for the storage of lifecycle information in a scenario where no common standards are used is proposed. In Section 6, a comprehensive architecture for the capture, management, and utilization of lifecycle information—the LIS.KOM framework—is described, while Section 7 covers the three different steps we took in order to implement our approach for different levels of generalization, their current implementation state, as well as evaluation plans and first results. Section 8 concludes this paper and gives an outlook on future work.

2 Related Work

Concerning the capture of information during the lifecycle of a Learning Resource, there is a similar approach called *Ecological Approach* [5], [27]. This work constitutes that information should be gathered during the actual use of a Learning Resource and not during explicit labeling phases, i.e., phases where metadata are created manually, only. However, the authors focus mainly on one phase of the lifecycle—the usage or learning phase—to gather information about the learner in order to support their approach of learner modeling. Information being captured about learners includes information about the interaction between a learner and a document like access patterns, numbers of keystrokes or dwell time, learner feedback, as well as information about the technical context of use like applications and hardware used by the learner. The learner model generated from this information is then attached to the Learning Resource and linked to other learner models attached to Learning Resources. The collected instances of learner models are then inspected for patterns. The amount of information and patterns is growing over time—therefore "ecological"—and can be processed to, e.g., recommend content to learners, support the formation of groups of learners, and provide study aids or a learning sequence planner. Other phases of the lifecycle except for the actual learning phase as well as the relations between Learning Resources emerging from these other phases are not taken into account.

The utilization of *contextual lifecycle information* in a wider sense has been conducted for several years now in known applications like eBay or Amazon, as well as in many works in information retrieval research, e.g., [16], however, there is hardly any work that concentrates on the capture of information when it emerges.

Attention Metadata or *Contextualized Attention Metadata* (CAM) [6] contain information about the attention a user pays to different Learning Resources via different applications. Users' activities are captured in so-called *feeds* with one feed per system. Feeds contain items. Items depict the attention a user pays to a specific document. An item has events, which themselves encompass session information as well as information about single actions taken on the document [47]. In existing approaches, CAM is, for instance, gathered and utilized to gain information about a user's experience [36] or used for the retrieval of Learning Resources [39]. In [46], Attention Metadata are even used to combine learning with knowledge management. For capturing and managing Attention Metadata, an approach similar to ours involving plug-ins and a central instance is used. While initially only the actual usage of an object was considered, the authors in [31] propose to take also into account the creation phase as well as reused components. Nevertheless, we have not been able to find a system where this is actually done. Systems where CAM is captured at the moment include the FireFox browser, MSN Messenger, and MS PowerPoint. The capture of CAM in PowerPoint encompasses events like saving or opening a document, editing durations and times of last access, as well as query information for searches with the ALOCOM plug-in for PowerPoint [45]. The CAM framework [35] is actually very closely related to our work. With respect to context information, the underlying scheme is very well designed. However, the scheme is designed for information to be collected in a user centric way and not for the storage and processing of document centric information. A transition is in most cases possible if enough information is stored, though. However, things get quite complicated with CAM when it comes to the storage and processing of relations. While the mapping of a relation between two documents could be done with the CAM scheme, it is not suited for an outright capture of more granular relations. In order to achieve this, additional information about elements within the documents a relation points to is needed (see Section 5.2). Though the information needed could theoretically be stored using CAM, it generates too much overhead from our point of view. Therefore, it seems to be not suited for our approach that focuses on relations between Learning Resources. Nevertheless, it might be worth to enable an export of lifecycle information captured with our system to the CAM scheme to be compatible with other applications that mine attention metadata in the future.

Semantic Desktops are meant to support Personal Information Management. Examples of Semantic Desktops are Gnowsis [43] or Haystack [23]. With Semantic Desktops, users can organize their personal information in a semantically rich way. *Social Semantic Desktops* like NEPOMUK [18] provide the functionality of semantic desktops in a community scenario by connecting the Semantic Desktops of several users. A Semantic Desktop is a complex application, which merges several technologies and approaches. For us especially, the approaches in *desktop search* and *usage and context mining* are of interest. In [9], the utilization of context information for semantic desktop search is described. The context information involved here is captured from emails, the file system structure, and the browser cache. Thus, documents sent via emails can be related to emails, which provide context information about them. Additionally, semantic information is gotten from folder hierarchies, since most people organize their folders by topics. Capturing is done mainly by monitoring file system events like the deletion or creation of files [3]. Beagle++ [8], a desktop search engine, enriches the usual full text index with contextual information like, e.g., numbers of accesses or hits

for documents to rank the search results. Even social aspects for recommendation are taken into account. Nevertheless, the information captured in these approaches is mainly at the usage or access level. Relations resulting from authoring, reuse, or reauthoring processes are not considered.

A system in which relations between Learning Resources are used is the *HyLOS* system [15]. HyLOS is mainly a Learning Management System and relations created semiautomatically or manually by users are used to provide additional links to learners in order to enable a constructivist learning style. The method used to store relation information is based mainly on the Dublin Core extension for the LOM relation category [42]. The relations used in HyLOS are on a higher semantic level than the relations taken into account by our approach. One of the core features the HyLOS system provides is relation enrichment. An existing set of relations is processed to generate new relations by using a certain rule set. This is quite similar to the enrichment component within the LIS we propose in Section 6.

The *TenDAX* system [19] is a collaborative text editing environment, which captures information that emerges during the creation of documents. This is done by storing every transaction carried out on the document within a special database model. Relations are captured by storing information about copy and paste actions between documents. The information captured is used to provide an alternative solution for the inflexible standard file system. The system also provides a networked visualization of the information gathered. However, there is only one type of relation captured this way. Since the system relies on the TenDAX editor as opposed to standard office tools, the approach does not seem to be very integrative or suited for common use.

Mueller [34] proposes an extended versioning system for structured and unstructured documents. To achieve consistent *management of change*, a special XML model for the documents in question has been proposed. With this model, relations between documents and parts of documents can be depicted and captured. However, the approach does not aim to collect lifecycle information but offers an explicit system for a better versioning of documents.

Besides the closely related approaches in the area of Technology Enhanced Learning, there are also interesting approaches in other areas. In software engineering, Requirements Traceability deals with related issues [17]. Here, the lifecycle of requirements of software from the specification to the implementation and evaluation has to be monitored. The different phases of a requirement's lifecycle are modeled and the information needed for its tracing has to be gathered. From a methodogical point of view, this is very much related to our approach, although the specific techniques for capture, utilization, and management of information are quite different.

3 Lifecycle Information

According to our understanding, lifecycle information is a special kind of metadata, which, in contrast to the common notion of metadata, is not related to a specific object, i.e., Learning Resource, but emerges from a certain process.

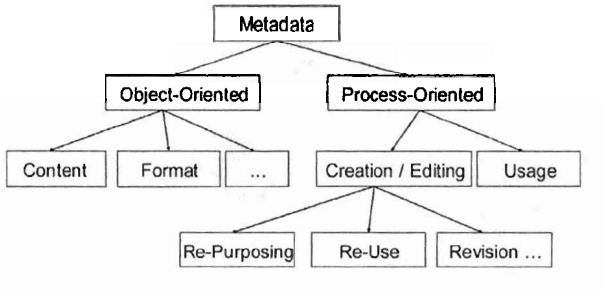

Fig. 1. Schematic metadata taxonomy.

Fig. 1 shows a schematic taxonomy, which depicts this issue exemplary. Object-oriented metadata are located in the left main branch. It describes the content of a Learning Resource like, e.g., the description or keywords, the format, the didactic implications, or legal issues as known from standards like LOM [21]. Lifecycle information covers the right main branch of the tree and emerges from processes like interaction of users, learners or authors with, or their actions on a Learning Resource. In order to identify the information that emerges during processes, the processes themselves have to be identified. In order to determine these processes and the information that emerges, we have modeled the lifecycle of Learning Resources. Starting from there, we have identified and studied two general types of lifecycle information, which occur in the different stages of the lifecycle: relation information and context information.

3.1 The Lifecycle of Learning Resources

There are existing works that propose models for the lifecycle of Learning Resources. In [10], a rather general model with several phases has been proposed. Here, labeling, i.e., the creation of metadata for a Learning Resource, is an explicit phase in the lifecycle. Cardinaels has modified this lifecycle to allow the generation of metadata in parallel with all other phases [7]. To enable the collection of metadata during the whole lifecycle of a document is a basic condition for our approach to work. Like Learning Resources themselves, models are always designed to serve a certain purpose. The purpose of the model we have developed is to identify the processes that cause the emergence of lifecycle information.

Fig. 2 shows our model of the lifecycle of a Learning Resource. Learning Resources are originally created with authoring tools (Authoring Phase), before being supplied to customers, teachers, or learners, e.g., in a learning object repository or marketplace (Provision Phase). Finally, they are used and utilized, which in the case of Learning Resources typically takes place in a Learning Management System (Learning Phase). However, for the majority, the Learning Resources available within a repository do not fit the special needs of customers, teachers, authors, or learners who search for Learning Resources. In order to be reusable, the Learning Resources have to be adapted to the new context of use. Thus, the Reauthoring Phase has been introduced to the lifecycle model (see [40] for a detailed discussion). By using Reauthoring tools, existing Learning Resources can be unitized, adapted, updated, and reaggregated. Parts can be added to or removed from a Learning

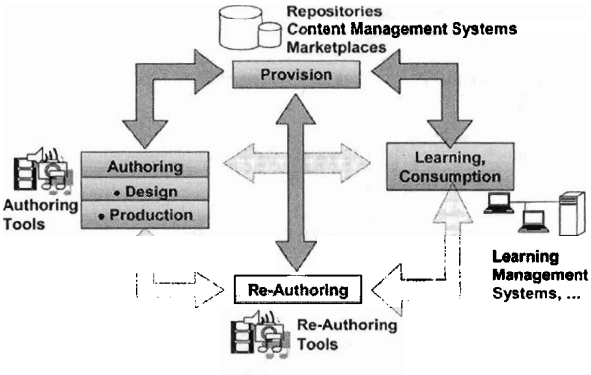

Fig. 2. Learning resource lifecycle model.

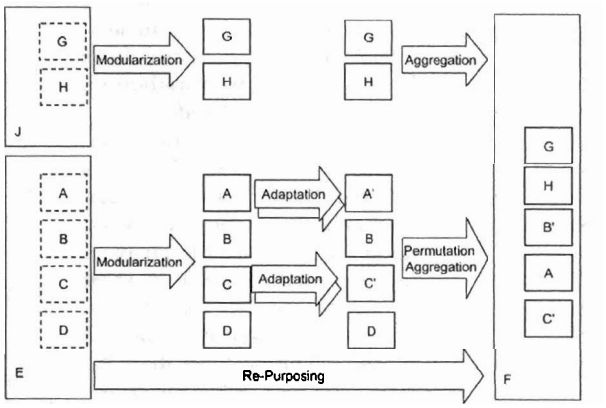

Fig. 3. Repurposing process [40].

Resource or parts of different Learning Resources can be joined to form a new one. In Fig. 3, an example of this whole process has been depicted. First, the existing Learning Resource E is disassembled. Thereby, four new Learning Resources are generated (A, B, C, and D). In this scenario, two of the resources generated are adapted for a new context (Adaptation) before the parts are put into a new order (Permutation) and with two other Learning Resources (G and H), put together to form Learning Resource E' (Aggregation). This process is called Repurposing, because the original Learning Resource has been changed to suit a new purpose. Repurposing is a special kind of Reauthoring [40]. We distinguish two kinds of information generated during the lifecycle of a Learning Resource: Relation and Context information. In the following sections, we will study both concepts and analyze which kinds of information occur in which phase of the lifecycle.

3.2 Context Information

While relation information always connects two or more Learning Resources, context information is restricted to one Learning Resource. Since our approach focuses on the Learning Resource itself, context information—as we understand it—represents the contexts a Learning Resource goes through during its lifecycle. Context information is generated implicitly, mainly during the usage or retrieval of a Learning Resource. Thus, it is usually generated during the Provision and Learning Phase. An example of context information is the number of views a Learning Resource gets in a repository or market place. Accordingly, the number of downloads or the number of times a Learning Resource is sold could present valuable information, e.g., as input for ranking methods. In a community-based scenario, ratings, comments, or feedback messages are also context information. During the Learning Phase, several kinds of context information can be collected, especially if learning takes place in a Learning Management System. The time needed for a learner to study a Learning Resource, the assessment statistics, or the number of students who viewed or even failed the assessment of a Learning Resource are only a few of the many different types of information that can be collected. Naturally, even in the authoring or Reauthoring phase, there is context information to capture, e.g., the time it took for a Learning Resource to be edited, which editor was used or who edited it. Even feedback from learners or other authors related to a Learning Resource, in our opinion, is context information. Many of the related approaches described in Section 2 concentrate on the collection of context information. However, there is a difference between user-centric and document-centric collection of context information. In our case, information collected for specific Learning Resources is document-centric. Most of the related work, however, collects user-centric context information. With the Attention Metadata specification [35], [6], for example, the attention a user gives to various objects like documents, e-mails, or web pages can be tracked. Such information is often used to model user or learner behavior.

Especially in the case of context information, information should be collected only if there is a clear utilization scenario, for which the information can be used, mainly because there is such a sheer amount of information capturable. In our approach, we concentrate on context information that can be applied in a concrete utilization scenario and supports us in processing the relation information we collect (see Section 4).

3.3 Relation Information

In contrast to context information, relation information does not correspond to only one Learning Resource. A relation always connects two Learning Resources or instances of Learning Resources. Regarding the lifecycle shown in Fig. 2, relation information emerges mostly in the Authoring and Reauthoring phases, as these are the two phases in which the content of Learning Resources is actually changed. In some cases, relations are also built in other phases, e.g., when a Learning Resource is downloaded from a repository and thus a new version or instance of this Learning Resource is created. We have identified a set of relation types generated during the Authoring and Reauthoring phases. These relation types are described in the following, before they are correlated to certain (Re-) authoring actions.

Aggregation relations or "part of" relations are derived from the composition of several Learning Resources resulting in a new Learning Resource. Each of the resources composed has a "part of" relation to the latter.

Provision relations result mostly from reuse processes. Two Learning Resources with the same level of aggregation

TABLE 1
Reauthoring Processes and Implicated Relation Types

Re-Authoring	User Action	Generated Relation Types
Update	Update Learning Resource	Version
Correction	Correct errors	Version
Modularization	Decomposition	Aggregation / Sequence
Adaptation	e.g. Translation, Printability, Accessibility [48]	Variant, Reduction
Aggregation	Aggregation	Reduction / Permutation / Aggregation / Sequence / Provision

are connected by "providesElementTo" if one Learning Resource contains an element, which is part of another Learning Resource. Basically, this is similar to an aggregation relation; however, the difference is that instances connected by this relation reside on the same aggregation level.

Sequence relations exist between Learning Resources with a particular sequential order. Two consecutive Learning Resources are connected either by a predecessor or successor relation, respectively.

Permutation relations connect two Learning Resources, which consist of the same modules, while these modules have a different sequential order.

A reduction/extension relationship occurs, when parts are removed from a Learning Resource. In this case, the two versions of the Learning Resource are connected by an "isReductionOf" or "isExtensionOf" relation.

Requirement relations are created when the processing of a Learning Resource requires the processing of a second Learning Resource.

Version relations relate two instances of a Learning Resource in the same version history to one another. These relations occur only if a Learning Resource is changed slightly, for example, in order to correct errors or to update facts (see [40] for details).

Variant relations persist between different variants of a Learning Resource. Variants are branches in the version history of a Learning Resource. They result mostly from adaptation processes, like translations, layout adaptations, or changes of the target group or the design of a Learning Resource. Here, Learning Resources are adapted to different contexts of use. In [40] and [48], Reauthoring processes and adaptation processes are defined. This includes the 15 different adaptation processes used most often in practice. Each of these Reauthoring and adaptation processes implies a specific type of relation information being generated. Table 1 gives an overview of these processes along with the information that emerges when these processes occur.

Updates and corrections are Reauthoring processes, which are performed very often, implying the emergence of version relations. Examples of user actions that imply such relations include the revision of content, when new circumstances like the introduction of the Euro currency are given, or the correction of errors. Typically, these actions do not change the learning objective or target group of a Learning Resource in contrast to the adaptation processes mentioned.

The modularization of a Learning Resource implies that it is decomposed into a number of modules. Modules, by definition, are Learning Resources themselves. Decomposition accounts for the generation of relation information, like aggregation relations between the decomposed Learning Resource and each of its modules, as well as sequential relations between consecutive modules.

The adaptation of a Learning Resource implies—as mentioned above—a variant relationship between a source and its targeted Learning Resource in the adaptation process. A variant relation implies a larger change in a Learning Resource than a version relationship does. A variant has its own version history. Version and variant relationships are fuzzy by nature and quite generic. Therefore, the kind of adaptation that has led to the emergence of a variant should be taken into account in order to differentiate them. The actual type of adaptation can be captured easily during the Reauthoring process. It just has to be stored properly (see Section 5).

Finally, the aggregation process implies different types of relation information, like aggregation, provision, reduction, or sequence relations. In fact, most of these relations occur throughout the entire Reauthoring process, but as aggregation is often the last step of the whole process, the relations mentioned cannot be captured until the final aggregation has been performed.

4 Utilization of Lifecycle Information

There are several possibilities for utilizing lifecycle information. The identification of new ways to support authors, learners, providers, or just users of Learning Resources by using lifecycle information is an ongoing process. There are two main areas where we want to utilize the information captured: The retrieval and (Re-) authoring of Learning Resources. However, even during the usage phase of a Learning Resource, there are imaginable situations when lifecycle information can be used effectively. We have developed different utilization scenarios to determine which information should be captured for what kind of utilization and partly implemented them. These scenarios are described in the following.

In order to support the retrieval of Learning Resources, there are different possibilities. First, we have to consider the ranking of search results. By using context information such as the number of views, downloads, or purchases in a repository or a marketplace, a search result set of Learning Resources can be ranked. There are several approaches, which already do this (like, for example, [39]). We believe that the ranking can be improved, if relation information is taken into account. By counting, e.g., the aggregation or provision relations of a Learning Resource, the number of reuses can be determined. In addition, this information can be weighted based on information about the author reusing the Learning Resource. If a Learning Resource is reused by

its creator, the information is probably less significant than if the Learning Resource is reused by a different author. This information can be used to further improve existing ranking approaches. Furthermore, different popularity rankings can be rendered for a result set, e.g., the most often viewed, reused or downloaded ranking, or combinations thereof. In [26], we show a prototypical implementation of this.

The provision of intelligent recommendations is a big field of research and is used in Digital Libraries as well as commercial systems like Amazon or eBay. Most approaches make use of collaborative filtering (like, e.g., [16]) and bibliographic citations, which can be considered as specific type of relation. We think that in this area lifecycle information can also be helpful. For instance, to augment a search result or recommend related resources to a user or author. Especially, relation information is helpful here. Despite not matching search terms, Learning Resources can be added to a result set, if they have a specific relation to the Learning Resources within the result set. For example, all Learning Resources with a requirement relation can be displayed to cover the basics of a certain topic, while sequence relations might indicate a certain degree of affinity between Learning Resources. A translation of a Learning Resource would not be in a result set unless the variant relation connecting the two is taken into account. Aggregation relations that we capture in the ResourceCenter [20] can serve as a concrete example: Due to the modular structure of courses created with the ResourceCenter, it is easily possible to capture aggregation relations between a course and the associated sections. These relations are stored for every section as part of one or more courses. If an author searches the ResourceCenter for sections, he plans to reuse, the aggregation relations of the sections that match the search query are traversed, and links are added to the search result, which point to other sections that are used within the same courses. The same is applicable for sequence or variant relations.

The possibility to browse via relations is closely related to the augmentation of a result set. Here, the user may browse through the relations captured during the lifecycle of a Learning Resource instead of directly searching for Learning Resources. In this case, the user does not need to provide specific search terms. The only thing he would need is an anchor from which to start browsing. Thus, the user would find Learning Resources even if they use a different vocabulary to describe similar facts. Certainly, the approaches mentioned above can be combined so that a user achieves a ranked and augmented search result from which he can pick a Learning Resource, which he wants to start browsing from. Again [26] shows a prototypical implementation of this approach.

Besides the retrieval, there are several possibilities to support the authoring of Learning Resources with lifecycle information. Authors can be notified when their Learning Resources are reused. Thus, this makes them aware of changes or revisions to their resources. On the other hand, if an author reuses a Learning Resource of another author, he can be notified about the changes that have been made to the original Learning Resource. This mechanism can also be used to provide feedback to an author, especially combined with context information such as the number of downloads or reuses. In this way, an author can be notified, e.g., about the popularity of his resources. On the other hand, weaknesses in his Learning Resources can also be indicated. If an above-average number of students fail a test after using a certain Learning Resource, a revision of that resource may be necessary and the author should be notified about this. Another type of context information for which this is relevant might be time duration. If students need significantly more time for one part of a Learning Resource than another, this perhaps could be split into two parts. The same applies for slides in a presentation: If one requires significantly more time than the others, it might indicate that the contents of this slide have to be split and distributed over two slides. Relation information can also be helpful in supporting the management of local documents. Users often have a hard time finding reused objects on their computer [37]. For example, if a user opens a presentation and he knows that some of the slides have been reused, he might not remember where the original presentation of the reused slides is stored. With the help of relation information (provision relations), the user can find the corresponding presentation. Context sensitive relation information could be visualized within a presentation to show related slides or assets to a user.

Apart from the scenarios where we want to utilize lifecycle information, there are several approaches in ongoing research that utilize kinds of lifecycle information or contextualized metadata differently and should not be left unmentioned. Most of them have already been implemented or are currently in work. A few examples of ongoing work in research for the utilization of lifecycle information are given in the following:

- Extending information about learner behavior for efficient learner modeling [27], [36];
- Supporting authors in authoring by aggregation and recommending Learning Resources with the help of aggregation context [31];
- Collecting and providing feedback from learners and other authors to the authors of Learning Resources [41]; and
- Update and consolidate existing metadata with context information [3].

5 Storage of Lifecycle Information

When lifecycle information is captured, it needs to be stored somewhere. This section describes a concept for the storage of both lifecycle and context information. First, an extension for the LOM metadata standard is proposed. Section 5.2 covers the storage of lifecycle information in scenarios where the LOM extension cannot be applied.

5.1 LOM Extension

When Learning Resources that follow established standards like LOM [21] are involved, it makes sense to extend these existing standards for the storage of lifecycle information. Since the LOM standard is a widely used standard for metadata, we decided to use it as the basis for the storage of relation-based lifecycle information. LOM consists of nine categories with about 60 fields. The categories of interest for

TABLE 2
Relation Types and Their Vocabulary

Relation Type	Vocabulary of the LOM Extension
Aggregation	hasPart / isPartOf
Provision	providesElementTo/ containsElementOf
Sequence	isPredecessorOf / isSuccessorOf
Permutation	isPermutationOf
Reduction/ Extension	isReductionOf / isExtensionOf
Requirement	requires / isRequiredBy
Version	hasVersion / isVersionOf
Variant	hasVariant / isVariantOf

the storage of lifecycle information are category 2 ("Lifecycle"), category 3 ("Meta-Metadata"), or category 7 ("Relation"). Category 2 reflects the current status of the respective Learning Resource in terms of versioning. It provides fields for the depiction of the completion status, the contributors (persons or organizations), their roles and entities, as well as contribution dates. Since we try to store mainly relation information within LOM, this category provides—except for the name—nothing of use for our purpose, at least not without changing the standard too much. Category 3 handles information about the metadata record itself and how it evolves over time. It provides, among others, roughly the same fields as category 2 and is therefore not suited to store relations between Learning Resources (or metadata records).

The category that matches our interests best is category 7: *Relation*. In this category—by default—the storage of relation information is covered. It may consist of an arbitrary number of relation fields, each containing the ID of the related resource and the type of relation. However, the Dublin Core vocabulary [42] used to express the relationships in the LOM relation category is not sufficient to fulfill the needs specified in Section 3. Therefore, we have developed our own vocabulary, which enables us to express our types of relations. Table 2 shows how the different kinds of relations are named.

As mentioned in Section 3.2, version and especially variant relations need to be typed. Therefore, we need to figure out a way to uniquely map aspects of change without affecting the underlying LOM standard too much. The LOM standard itself is very rich and covers many aspects of a Learning Resource. Therefore, we are able to relate these aspects and yet stay independent from the content of the Learning Resource itself. For this purpose, the relation category was extended by a field named Changes. This field exists for every relation and consists of a pointer and a value. The pointer points to the LOM field that was changed by the process that led to the existence of the relation and the value depicts the old value of that field. Thus, when a Learning Resource is translated from German to English, the value of the LOM field *General.Language* would change from "de" to "en." The Changes field would then hold the pointer to *General.Language* as well as the old value: "de." Thus, it is possible to reconstruct which relation is implied for which changes. In turn, it is also possible to include changes in the metadata of the original Learning Resource in order to determine which changes have been done to it. While it is especially helpful to type variant relations, other kinds of relations can also be typed. Fig. 4 shows an excerpt of a relation element, which depicts a reduction relation resulting from a change in semantic density and the learning duration of a Learning Resource. The changes field is not mandatory for the relation to be comprehensive. It would be possible to compare the metadata records of related Learning Resources to find out what type of change occurred. But using the "changes" field is, on the one hand, faster, and on the other hand, only one metadata record is needed to determine the type of relation. A relation itself has significance without knowing the application or context it was generated in. That means that if it is known that Learning Resource A is a part of Learning Resource B, it is not relevant for the interpretation of the relation to know which application was used to create this relation. Therefore, it makes sense to store relation information within the metadata close to the Learning Resource itself. Context information in contrast is highly dependent on the system it emerges in. This means that, for example, the number of views or queries for a Learning Resource in one marketplace might have a different meaning in another marketplace or repository, due to the number of users, the number of Learning Resources provided, or the target group. Furthermore, the sheer amount of context information that can be captured makes it impractical to store within LOM. Therefore, context information is stored in an independent format in a central instance. The schema for the storage of context information includes an identifier for the Learning Resource, the type of information—like sold, bought, downloaded, viewed, and so forth—and finally an identifier for the system the information was captured in. Thus, it is possible to weigh the captured context information accordingly.

```
<relation>
  <kind>
    <source>http://www.contentsharing.com/relation</source>
    <value>isreductionof</value>
  </kind>
  <resource>
    <identifier>
      <catalog>content_sharing</catalog>
      <entry>"modul1-uuid"</entry>
    </identifier>
  </resource>
  <changes>
    <date/>
    <categorie>educational</category>
    <dataelement>typicalLearningTime</dataelement>
    <oldvalue>PT30M</oldvalue>
  </changes>
  <changes>
    <date/>
    <categorie>educational</category>
    <dataelement>semanticalDensity</dataelement>
    <oldvalue>medium</oldvalue>
  </changes>
</relation>
```

Fig. 4. LOM extension excerpt.

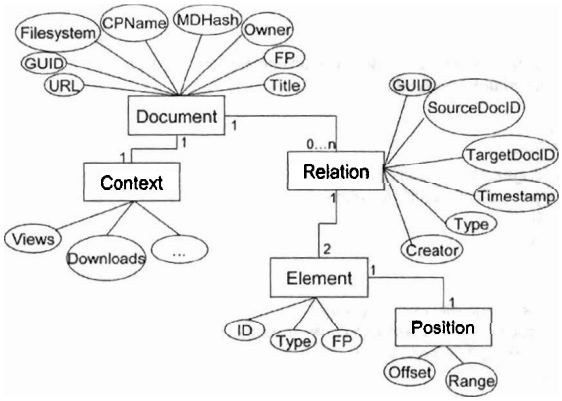

Fig. 5. Lifecycle metadata schema.

5.2 Lifecycle Metadata Schema

In case Learning Resources do not follow the common standards, alternative solutions are needed for the storage of lifecycle information. CAM [6] is a good means to store context information, but for the storage of relations, there is no existing specification or schema we can make use of. Therefore, we have developed a scheme to store relation and context information for each document. The main purpose of this scheme is the exchange of lifecycle metadata. Fig. 5 shows the schema we use for the storage of lifecycle information we collect. The focus is on relation information here. Since we store information, document-centric *document* is the central instance. Each document possesses a GUID stored in its properties and in the scheme. Besides that, information like the URL, owner, title, and a (high similarity near distance) fingerprint is stored. Each document can possess an arbitrary number of *relations*. A relation has a GUID and a pointer to the GUID of the target document of the relation (the source document is the document the relation belongs to). Furthermore the relation type, its creator, and a timestamp is stored. Additionally, we need to store information about the *element* (e.g., slide, paragraph, and so forth) a relation points to. Specifically, in a scenario where Learning Resources are involved, which do not possess an open, modular data model (like, e.g., PowerPoint presentations), this additional information is needed. An element may, e.g., be a page, a paragraph, a slide, or a media object like a picture, video, or audio file. It has a type, ID, and Fingerprint. Optionally, it can be extended by a position object depicting the offset and range of a certain paragraph in a text document.

Context information is stored under the context element. This section is still a work in progress since we want to capture context information only in cases where it helps in the utilization of relation information. For the storage of context information, it might be reasonable to implement the CAM specification (or parts of it) since it provides good means to store this kind of information. Additionally, this would lessen the effort of a later on "LIS.KOM-to-CAM" transformation. When using XML as format for lifecycle metadata, it is possible to integrate lifecycle metadata in the documents themselves. Especially, the new XML-based document formats like ODT [38] or Office Open XML [33] allow for this.

6 The LIS.KOM Framework

From the different types of systems that cover the different phases of a Learning Resource's lifecycle, like authoring tools, repositories, or Learning Management Systems, there are few in which lifecycle information is captured; and if it is captured, the information remains in these systems and gets stuck at system borders. Examples of information already being captured by existing systems include usage and assessment information in Learning Management Systems or numbers of downloads or purchases in repositories. However, if Learning Resources are transported via system borders, this information gets lost. This is due to the lack of a standardized format for its storage and a suitable architecture for the management of this kind of information. The goal of the proposed architecture is therefore to enable the storage, capture, and utilization of lifecycle information beyond system borders. To achieve this, we designed the LIS.KOM (Lifecycle Information System) framework with a central component called LIS.KOM Server and distributed clients. The components of the LIS.KOM framework, which we describe in the following, are shown in Fig. 6.

6.1 LIS.KOM Server

The LIS.KOM Server has been designed as an online central application and is interconnected to LIS.KOM Clients via Web services, i.e., clients use Web services running on the server to send information collected for the locally available Learning Resources to the server or to retrieve it from there. We decided to use a centralized architecture, because this allows us to easily connect to the server and to update information. The LIS.KOM Server integrates all data sources and manages incoming data. Here, the information gathered is processed, evaluated, and stored in a database. The server collects all information that is related to any version of a Learning Resource. From these individual facts, a lifecycle record for each Learning Resource is built. This lifecycle record contains a representation of all instances (versions and variants) of a Learning Resource, relations between these instances, and context information for each instance. The context and relation information stored separately is enriched by making use of a special rule set (e.g., if A isPartOf B and B isPartOf C, then A isPartOf C, and so forth). The enriched record of a Learning Resource can then be retrieved by the LIS.KOM Clients for utilization. In a productive system, the ownership of collected lifecycle information as well as security issues like protection against fraud and misuse would be an issue and should be handled by the LIS.KOM Server. However, these challenges are not in the scope of this paper.

6.2 LIS.KOM Clients

A LIS.KOM Client wraps up the local functionality needed for the management of lifecycle information. It implements the Web service APIs provided by the LIS.KOM Server to retrieve lifecycle information from it or put the locally collected information onto it. Additionally, it serves as cache for the information captured locally and is thus operable even in an offline scenario. For this purpose, the locally captured information is preprocessed in a way similar to the processing on the LIS.KOM Server, so that it can be utilized locally in an offline scenario. When a connection to the server is established again, the local cache

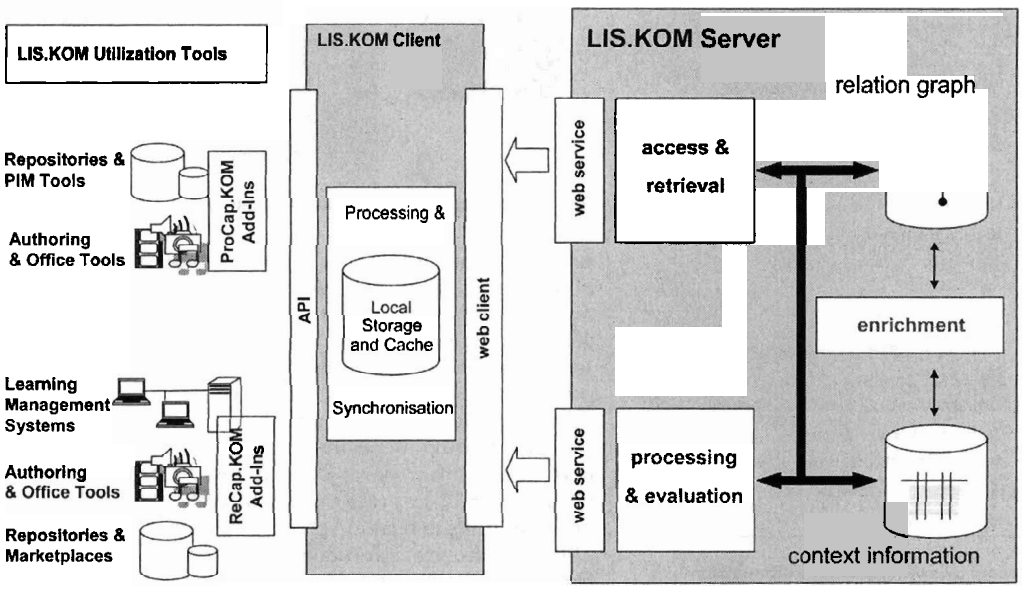

Fig. 6. LIS.KOM framework.

is synchronized with the server via the Web service API. In addition, the LIS.KOM Client provides an interface for ReCap.KOM and ProCap.KOM add-ins, which are responsible for capture and utilization of information within different applications. Alternatively, stand-alone utilization tools can implement this interface. As a fallback solution—in case an online connection is never established—the lifecycle information may be attached as metadata to the Learning Resource; the Resource itself then serves as the transfer medium.

6.3 Capture and Utilization Add-Ins

There are two different kinds of add-ins connected to the LIS.KOM Client. ReCap.KOM add-ins are responsible for the capture of lifecycle information. This happens from within the respective applications where information needs to be captured. Fig. 7 shows the basic components of such an add-in. Although the actual implementation is different for each application, the basic components are always similar. The *event dispatcher* wraps up and handles all sources of events triggered by relevant user actions. It usually uses a combination of application-specific events (e.g., provided by the API of the application), mouse, and keyboard hooks, as well as Clipboard events. A synchronization module makes sure that the state of captured information is synchronous to the application state. This usually includes the handling of undo- and redo-actions performed by the user. Since it cannot always be assumed that every user who edits a document has the respective add-in installed, the information, specifically the relations stored for a document, has to be verified. This is done by the verification module. Naturally, a capture add-in implements the API provided by the LIS.KOM Client.

ProCap.KOM add-ins are plugged into applications where lifecycle information can be of particular use. This applies, for example, to applications used for the retrieval of Learning Resources, where context and relation information can be combined to provide better search results. The add-ins retrieve information about the Learning Resources from the LIS.KOM Client and present it in a helpful way. This, for instance, may be done by providing links to closely related Learning Resources or ranking a search result based on the context information collected (see Section 4). For each application, in which lifecycle information should be utilized, such an add-in is needed. The nature of these add-ins strongly depends on the application and the way in which the information is actually utilized. In some cases, it makes sense to have both types of add-ins plugged into an application. While searching in a repository, the information captured about Learning Resources contained in the repository is especially helpful, therefore making a ProCap.KOM add-in mandatory for its utilization. Additionally, there is information being generated while using a repository as well: The selection and access of Learning Resources will increase respective counters and provide additional information on the relative significance of certain Learning Resource instances. In addition to utilization add-ins, basically every application interested in the utilization of

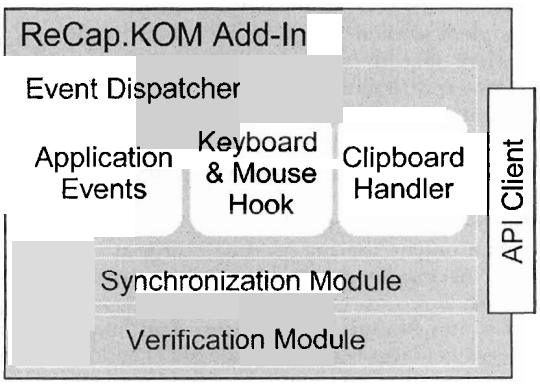

Fig. 7. ReCap.KOM add-in.

Introduction (Network Calculus)

Description:	Introduction to Network Calculus
Keywords:	Network Calculus
Creation date:	Sun Feb 19 17:51:42 CET 2006
Last modified:	Sun Feb 19 17:51:42 CET 2006
Mime type:	text/xml
Authors:	Nico d Heureuse
Views:	130
Downloads:	25
Reuses:	1
Prev. Version:	7f965fa08253f57001048a2e69ef0000
Used by:	Network Calculus - Course Communication Networks - Course
Uses:	Cars, slow - Animation Cars, fluid - Animation
Variant (Translation):	Einleitung (Netzwerkkalkül) - Section
Successor:	Outline - Section

Fig. 8. Learning Resource detail view with relation information.

lifecycle information can do so by using the given API of the LIS.KOM Client.

7 Implementation and Evaluation

We have implemented our approach in three different scenarios by conducting three steps from specific to generic. After we have described these steps and the current state of the implementation, we depict first results and further plans for an evaluation of our approach.

7.1 Implementation Steps

The first proof of concept was done in the ResourceCenter [13]. Due to its nature as authoring by aggregation environment, the capture of lifecycle information is an easy and intuitive task here. Capture and utilization are both carried out within the ResourceCenter and thus there is no need to traverse system borders. The authoring processes that provide lifecycle information are very specific and determinable and we have full access to the source code of the ResourceCenter in order to carry out capturing. We provide details about this implementation in [26]. Fig. 8 shows an example of the utilization of information collected in the ResourceCenter. Here, the detail view of a Learning Resource is enriched with links to related resources.

The second implementation step was carried out in the course of the Content Sharing project [11]. Here, in contrast to the ResourceCenter, different systems are involved, i.e., the information captured must transfer system borders. In order to enable this, the architecture presented in Section 6 has been partly implemented. Fig. 9 shows the components we have implemented in this scenario.

Capture add-ins and a LIS.KOM Client have been integrated into the Content Sharing Module Editor as well as in the Content Sharing Repository. A utilization add-in within the repository helps users in finding, searching, and retrieving the Learning Resources they want. And finally, the LIS.KOM Server has also been implemented. The local caching features of the LIS.KOM Clients are used to transfer the captured data via the Content Sharing Repository to the

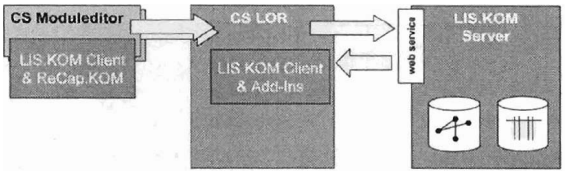

Fig. 9. Components implemented in Content Sharing.

server. The Module Editor is a combination of modularization, aggregation, and adaptation tool, which supports several adaptation processes. Hence, the ReCap.KOM add-in theoretically captures all the relation information described in Section 2.2. Although this is only possible if the processes are performed accordingly. At the moment, it only captures relation information. The add-in integrated into the Content Sharing Repository captures context information like views or purchases of Learning Resources, while the utilization add-in processes all the relations captured and provides links to related Learning Resources. For the sake of utilizing lifecycle information, this implementation is far from complete but served as a proof of concept.

An example of the use of this system is given as follows: A Learning Resource is downloaded from the Content Sharing Repository; meanwhile, the capture add-in of the repository counts the view and the download as context information for this resource. It is opened in the Module Editor and modularized into smaller Learning Resources. The ReCap.KOM add-in in the Module Editor captures the aggregation and sequence relations between the existing resources and stores it with the metadata of the Learning Resources in the local cache (or sends it to the LIS.KOM Server). Some of the submodules are adapted, which results in relation information being captured as well, and even newly created Learning Resources are added. Finally, the Learning Resources are aggregated and the corresponding information is captured. The author can decide which of the created Learning Resources to upload to the repository. He may upload any created or adapted module or he may choose only to upload the final Learning Resource. The relation information is transported to the repository either via the metadata of the Learning Resources or via the LIS.KOM Server. In the current implementation, the former is the case. In the repository, the LIS.KOM client gets information from the metadata or the LIS.KOM Server, respectively, and provides it to the utilization add-in, which generates links or recommendations. Before this is done, the information is filtered to show links to resources only, which truly exist in the repository. This prototypical implementation carried out in the course of the Content Sharing project is generic regarding the transfer of information and its underlying architecture but very specific for the processes that enable the emergence of lifecycle information and the software components used. The Reauthoring processes conducted with the Module Editor are still very specific and determinable. Additionally, we have full access to the source code of these applications for capturing.

Thus, the third and most challenging step is the implementation of our approach for generic tools with generic authoring processes like, e.g., office tools. The wider a tool is distributed and used, the greater the value of add-ins for this application. Thus, we decided to implement add-ins for generic office tools like Microsoft's Word and

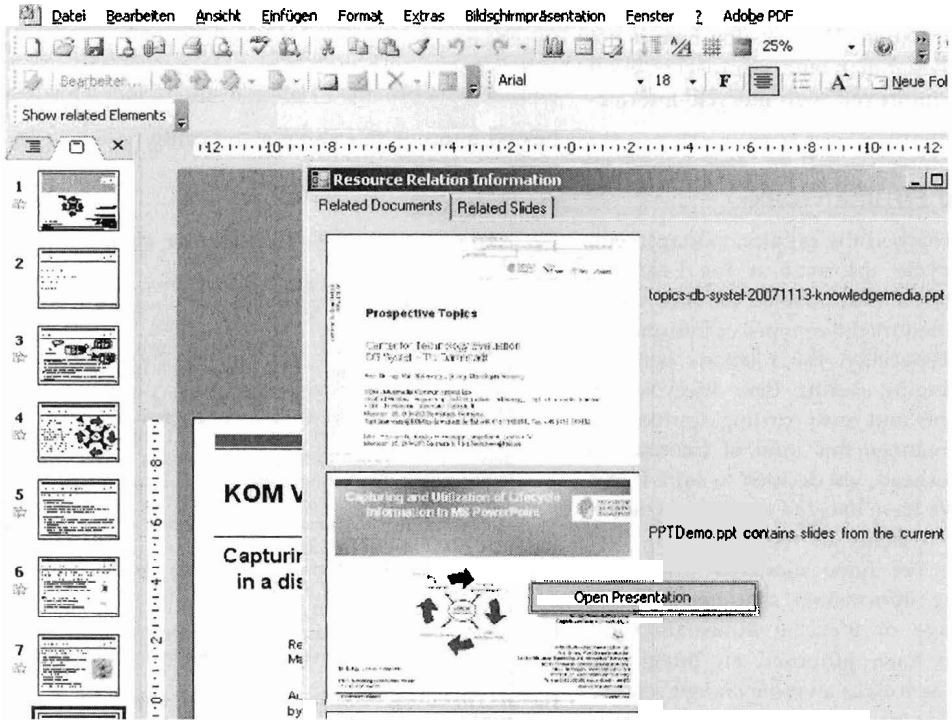

Fig. 10. ProCap.KOM utilization add-in in PowerPoint.

PowerPoint. For this case, we have used the independent storage format described in Section 5.2 because we could not expect office documents to implement the LOM standard. Nevertheless, many Learning Resources are still created using these tools. We have implemented capture add-ins as well as prototypical utilization add-ins for both PowerPoint and Word. Both are realized as .net add-ins using C# [14]. Capturing runs completely in the background, so that normal work with the applications is not affected. Most users should not even notice that an add-in is there. The information is stored in the local cache of the LIS.KOM Client and then sent to the LIS.KOM Server. Currently, we are able to capture provision and aggregation relations for assets, slides, shapes, and text within PowerPoint and Word, between both and from most other sources like websites or PDF documents. Additionally, we capture variant relations between different documents. For the utilization of collected information in PowerPoint, related presentations can be shown and opened directly from within a PowerPoint document. Fig. 10 shows the utilization add-in that provides information about and access to presentations that are related to the one that is actually opened.

7.2 Evaluation Plans and First Results

We have conducted a first small evaluation for the third implementation step. We installed ReCap.KOM add-ins for PowerPoint and deployed LIS.KOM clients on the computers of four test persons and let them do their usual work with PowerPoint. The evaluation was conducted as offline scenario, i.e., without connection to a LIS.KOM Server. It gave us two important insights:

1. There actually is a significant amount of reuse when users create PowerPoint presentations.
2. Relations can be captured with a high validity.

With 29 different documents opened in about four weeks, we were able to collect 58 provision relations resulting from the reuse of slides and 23 relations resulting from the reuse of external files like images. The overall validity of the captured relations was about 85 percent. However, we identified several possibilities to further improve the validity of the capture. Furthermore, we observed that a verification of captured relations is needed when the scenario is not closed, i.e., when users without respective add-ins edit involved documents. For details about this evaluation, we refer to [25]. There are three main questions we want to answer with our ongoing and planned evaluations:

1. Is the information we collect valid?
2. Is the information we collect significant?
3. Do the means of utilization we provide help users?

The validity of information is evaluated like described before. Our goal is to evaluate the add-ins in an open scenario where the test persons use the respective applications as usual. Thus, we will get additional insights in the reuse behavior of users. The remaining two questions are closely connected. We plan to evaluate the significance of lifecycle information to improve the retrieval of documents as well as to support authoring. At least for the latter we will have to conduct a user driven evaluation where users judge how helpful the captured information and the means of utilization actually is. To evaluate if our approach can help improve the retrieval of Learning Resources, we plan

to extend existing approaches for ranking and recommendation with lifecycle information. The question here is not whether our approach is better than other approaches but if other approaches can be improved with lifecycle information we collect.

8 Conclusions and Future Work

In this paper, we have proposed the capture, management, and utilization of lifecycle information for Learning Resources. We have analyzed and modeled the lifecycle of Learning Resources and identified two types of information that can be captured. Especially, the relations between Learning Resources emerging during their lifecycle are neglected by many systems and most existing approaches. In order to be able to manage this kind of information properly and without overhead, we decided to store it in a document-centric way. We have decided to capture context information only when it helps in the processing and utilization of relations. We have modeled utilization scenarios to identify the information that needs to be captured. For the storage of lifecycle information for Learning Resources, we have proposed an integrative approach based on LOM as well as an independent schema for Learning Resources that are not compliant to standards like LOM. We have introduced the LIS.KOM framework—a system that supports capture, management, and utilization of lifecycle information, which enables us to collect and use information in all phases of a Learning Resource's lifecycle. The three implementation steps have shown that our approach is valid in different scenarios and for different levels of generalization. Although, in most cases, the lifecycle information that we collect cannot be used to directly improve their learning experience, it helps learners implicitly in different ways. On the one hand, with lifecycle information, we can support authors in the creation and reauthoring of Learning Resources by facilitating access to and reuse of related documents. Notifications about changes and updates as well as feedback can help authors as well. In supporting authors, we indirectly support learners of the respective Learning Resources. On the other hand, improvements of the retrieval and better recommendations of Learning Resources can also lead to an improved learning experience. Here, relation information could even be used to support learners directly, e.g., by providing them links to related resources.

We have conducted first evaluations with promising results. Further evaluations, including a user-driven evaluation, are planned. The generic, plug-in-based architecture enables us and other developers to extend the number of supported applications easily. In the future, one major step besides evaluation will be the development of further add-ins for the LIS.KOM framework. Additionally, with the growing availability of information about Learning Resources, new ways of utilization will emerge, which must be implemented to support learners, authors, or providers of Learning Resources in different ways.

Acknowledgments

The authors would like to thank the reviewers of this journal, who provided elaborate and detailed feedback that helped improve this paper significantly.

References

[1] ARIADNE, *ARIADNE—Foundation for the European Knowledge Pool*, 2008.
[2] P. Baumgartner, "The ROI Paradox," *Proc. Gesellschaft für Medien in den Wissenschaften Conf.*, Keynote Presentation, http://www.c3-initiative.info/peter/2004/09/12, 2004.
[3] J. Belizki, S. Costache, and W. Nejdl, "Application Independent Metadata Generation," *Proc. First Int'l Workshop Contextualized Attention Metadata: Collecting, Managing and Exploiting of Rich Usage Information (CAMA '06)*, pp. 33-36, 2006.
[4] J. Broisin, P. Vidal, M. Meire, and E. Duval, "Bridging the Gap between Learning Management Systems and Learning Object Repositories: Exploiting Learning Context Information," *Proc. Advanced Industrial Conf. Telecomm./Service Assurance with Partial and Intermittent Resources Conf./E-Learning on Telecomm. Workshop (AICT/SAPIR/ELETE)*, 2005.
[5] C. Brooks and G. McCalla, "Towards Flexible Learning Object Metadata," *Proc. Int'l J. Continuing Eng. Education and Lifelong Learning*, vol. 16, nos. 1/2, 2006.
[6] CAMs, *Conceptual Base Scheme*, http://ariadne.cs.kuleuven.ac.be/empirical/attention/CAM%20schema_Document_v1.5.pdf, 2007.
[7] K. Cardinaels, "A Dynamic Learning Object Life Cycle and Its Implications for Automatic Metadata Generation," PhD thesis, Katholieke Universiteit Leuven, 2007.
[8] P. Chirita, S. Costache, W. Nejdl, and R. Paiu, "Beagle++: Semantically Enhanced Searching and Ranking on the Desktop," *The Semantic Web: Research and Applications*, pp. 348-362, Springer, 2006.
[9] P. Chirita, R. Gavriloaie, S. Ghita, W. Nejdl, and R. Paiu, "Activity Based Metadata for Semantic Desktop Search," *Proc. Second European Semantic Web Conf. (ESWC)*, 2005.
[10] B. Collis and A. Strijker, "Technology and Human Issues in Reusing Learning Objects," *J. Interactive Media in Education*, vol. 4, 2004.
[11] *Content Sharing Project*, http://contentsharing.com, 2007.
[12] E. Duval, E. Forte, K. Cardinaels, B. Verhoeven, R.V. Durm, K. Hendrikx, M.W. Forte, N. Ebel, M. Macowicz, K. Warkentyne, and F. Haenni, "The Ariadne Knowledge Pool System," *Comm. ACM*, vol. 44, no. 5, pp. 72-78, 2001.
[13] E. Duval and W. Hodgins, "A LOM Research Agenda," *Proc. 12th Int'l Conf. World Wide Web (WWW '03)*, G. Hencsey, B. White, Y. Chen, L. Kovacs, S. Lawrence, Hrsg., Seite 1-9, 2003.
[14] ECMA, *CSharp Language Specification*, http://www.ecma-international.org/publications/standards/Ecma-334.htm, 2008.
[15] M. Engelhardt, A. Hildebrand, D. Lange, and T.C. Schmidt, "Semantic Overlays in Educational Content Networks," *Proc. TERENA Networking Conf. (TNC)*, 2006.
[16] N. Good, J.B. Schafer, J.A. Konstan, A. Borchers, B. Sarwar, J. Herlocker, and J. Riedl, "Combining Collaborative Filtering with Personal Agents for Better Recommendations," *Proc. 16th Nat'l Conf. Artificial Intelligence and 11th Conf. Innovative Applications of Artificial Intelligence (AAAI/IAAI)*, 1999.
[17] O.C.Z. Gotel and A.C.W. Finkelstein, "An Analysis of the Requirements Traceability Problem," *Proc. First Int'l Conf. Requirements Eng. (ICRE)*, 1994.
[18] T. Groza, S. Handschuh, K. Moeller, G. Grimnes, L. Sauermann, E. Minack, C. Mesnage, M. Jazayeri, G. Reif, and R. Gudjonsdottir, "The NEPOMUK Project—On the Way to the Social Semantic Desktop," *Proc. Int'l Conf. Semantic Systems (I-Semantics '07)*, T. Pellegrini and S. Schaffert, eds., pp. 201-211, 2007.
[19] T. Hodel, R. Hacmac, and K.R. Dittrich, "Using Text Editing Creation Time Meta Data for Document Management," *Proc. 17th Conf. Advanced Information Systems Eng. (CAiSE)*, 2005.
[20] S. Hoermann, T. Hildebrandt, C. Rensing, and R. Steinmetz, "ResourceCenter—A Digital Learning Object Repository with an Integrated Authoring Tool," *Proc. World Conf. Education Multimedia, Hypermedia and Telecomm. (EDMEDIA)*, 2005.

[21] *IEEE Learning Technology Standards Committee: IEEE Standard for Learning Object Metadata 1484.12.1*, IEEE, 2002.
[22] A. Ip, A. Radford, and E. Canale, "Overcoming the Presentation Mosaic Effect of Multi-Use Sharable Content Objects," *Proc. 20th Ann. Conf. Australasian Soc. Computers in Learning in Tertiary Education (ASCILITE)*, 2003.
[23] D. Karger, K. Bakshi, D. Huynh, D. Quan, and V. Sinha, "Haystack: A General Purpose Information Management Tool for End Users of Semistructured Data," *Proc. Conf. Innovative Data Systems Research (CIDR)*, 2005.
[24] L. Lehmann, T. Hildebrandt, C. Rensing, and R. Steinmetz, "Capturing, Management and Utilization of Lifecycle Information for Learning Resources," *Proc. Second European Conf. Technology Enhanced Learning (ECTEL)*, 2007.
[25] L. Lehmann, C. Rensing, and R. Steinmetz, "Capture of Lifecycle Information to Support Personal Information Management," *Proc. European Conf. Technology Enhanced Learning (ECTEL)*, 2008.
[26] L. Lehmann, C. Rensing, and R. Steinmetz, "Lifecycle Information Management and Utilization in an Authoring by Aggregation Environment," *Proc. World Conf. Education Multimedia, Hypermedia and Telecomm. (EDMEDIA)*, 2007.
[27] G. McCalla, "The Ecological Approach to the Design of E-Learning Environments: Purpose-Based Capture and Use of Information About Learners," *J. Interactive Media in Education*, vol. 7, 2004.
[28] M. Meire, X. Ochoa, and E. Duval, "SAmgI: Automatic Metadata Generation 2.0," *Proc. World Conf. Education Multimedia, Hypermedia and Telecomm. (EDMEDIA)*, 2007.
[29] MERLOT, *MERLOT: Multimedia Educational Resource for Learning and Online Teaching*, 2008.
[30] M. Meyer, T. Hildebrandt, C. Rensing, and R. Steinmetz, "Requirements and an Architecture for a Multimedia Content Re-Purposing Framework," *Proc. First European Conf. Technology Enhanced Learning (ECTEL '06)*, W. Nejdl and K. Tochtermann, eds., pp. 500-505, 2006.
[31] M. Meyer, C. Rensing, and R. Steinmetz, "Improving Authoring-by-Aggregation and Using Aggregation Context for Query Expansion," Creating New Learning Experiences on a Global Scale, *Proc. Second European Conf. Technology Enhanced Learning*, E. Duval, R. Klamma, and M. Wolpers, eds., pp. 505-510, 2007.
[32] M. Meyer, B. Zimmermann, C. Rensing, and R. Steinmetz, "An Interactive Tool for Supporting Modularization of SCORM-Based Learning Resources," *Proc.World Conf. Education Multimedia, Hypermedia and Telecomm. (EDMEDIA '07)*, pp. 3164-3171, 2007.
[33] Microsoft, *Office Open XML*, http://www.microsoft.com/germany/interop/openxml/, 2008.
[34] N. Mueller, "An Ontology-Driven Management of Change," *Proc. Wissens-und Erfahrungsmanagement, Lernen, Wissensentdeckung, Adaptivitaet Conf. (LWA)*, 2006.
[35] J. Najjar, M. Wolpers, and E. Duval, "Attention Metadata: Collection and Management," *Proc. WWW Workshop Logging Traces of Web Activity: The Mechanics of Data Collection*, 2006.
[36] J. Najjar, M. Wolpers, and E. Duval, "Towards Effective Usage-Based Learning Applications: Track and Learn from Users Experience(s)," *Proc. Sixth IEEE Int'l Conf. Advanced Learning Technologies (ICALT '06)*, July 2006.
[37] W. Nejdl and R. Paiu, *I Know I Stored It Somewhere—Contextual Information and Ranking on Our Desktop*, 2005.
[38] OASIS, *Open Document Format for Office Applications v1.1*, http://www.oasis-open.org/specs/index.php#opendocumentv1.1, 2008.
[39] X. Ochoa and E. Duval, "Use of Contextualized Attention Metadata for Ranking and Recommending Learning Objects," *Proc. Int'l ACM Workshop Contextualized Attention Metadata: Collecting, Managing and Exploiting Rich of Usage Information (CAMA)*, 2006.
[40] C. Rensing, S. Bergsträßer, T. Hildebrandt, M. Meyer, B. Zimmermann, A. Faatz, L. Lehmann, and R. Steinmetz, "Re-Use, Re-Authoring, and Re-Purposing of Learning Resources—Definitions and Examples," technical report, Technische Universität Darmstadt, 2005.
[41] C. Rensing, S. Tittel, L. Lehmann, and R. Steinmetz, "Ein System zur Realisierung Expliziten Lerner-Autor Feedbacks im E-Learning," *Proc. DeLFI Workshop "Effiziente Erstellung von E-Learning Content,"* Sept. 2006.
[42] Request for Comment (RFC) 2413, *Relation Element Working Draft*, http://dublincore.org/documents/1997/12/19/relation-element/, 1997.
[43] L. Sauermann, "The Gnowsis Semantic Desktop for Information Integration," *Proc. WM First Workshop Intelligent Office Appliances: Knowledge-Appliances in the Office of the Future (IOA)*, 2005.
[44] K. Verbert, D. Gasevic, J. Jovanovic, and E. Duval, "Ontology-Based Learning Content Repurposing," *Poster Session of the Int'l World Wide Web Conf. Committee*, 2005.
[45] K. Verbert, J. Jovanovic, E. Duval, D. Gasevic, and M. Meire, "Ontology-Based Learning Content Repurposing: The ALOCoM Framework," *Int'l J. E-Learning*, vol. 5, no. 1, pp. 67-74, 2006.
[46] M. Wolpers, G. Martin, J. Najjar, and E. Duval, "Attention Metadata in Knowledge and Learning Management," *Proc. Sixth Int'l Conf. Knowledge Management (I-Know)*, 2006.
[47] M. Wolpers, J. Najjar, K. Verbert, and E. Duval, "Tracking Actual Usage: The Attention Metadata Approach," *Int'l J. Educational Technology and Soc.*, vol. 11, pp. 106-121, 2007.
[48] B. Zimmermann, C. Rensing, and R. Steinmetz, "Format-Übergreifende Anpassungen von Elektronischen Lerninhalten," *Proc. DeLFI Die 4. e-Learning Fachtagung Informatik*, 2006.

Lasse Lehmann received the Dipl-Ing degree in electrical engineering and information technology from Technische Universität Darmstadt in 2005. He is a PhD student in the Multimedia Communications Laboratory, Department of Electrical Engineering and Information Technology, Technische Universität Darmstadt. His research interest lies in the area of knowledge media. He is especially interested in the capture and utilization of information emerging throughout the lifecycle of documents. In 2007, he was awarded the Best Student Paper at the European Conference on Technology Enhanced Learning.

Tomas Hildebrandt received the Dipl-Inform degree from the Technische Universität Darmstadt in 2001. He is a PhD student in the Multimedia Communications Laboratory, Department of Electrical Engineering and Information Technology, Technische Universität Darmstadt. His research interests lie in the area of networked gaming, especially the technical challenges in the community-driven environment of multiplayer online games.

Christoph Rensing received the PhD degree in 2003, with a thesis on "A policy-based access control architecture for the multi-service Internet." He is currently a research group head in the Multimedia Communications Laboratory, Department of Electrical Engineering and Information Technology, Technische Universität Darmstadt, where he leads a team of researchers in the area of knowledge media. In addition, he is research consultant at the Hessian Telemedia Technology Competence Center. He has published more than 60 papers in national and international conference proceedings and journals. He is a member of the ACM and the German Informatics Society (GI).

Ralf Steinmetz worked for more than nine years in industrial research and development of distributed multimedia systems and applications. Since 1996, he has been the head of the Multimedia Communications Laboratory, Department of Electrical Engineering and Information Technology, Technische Universität Darmstadt, Darmstadt, Germany. From 1997 to 2001, he directed the Fraunhofer (former GMD) Integrated Publishing Systems Institute (IPSI), Darmstadt. In 1999, he founded the Hessian Telemedia Technology Competence Center (httc e.V.). His thematic focus in research and teaching is on multimedia communications with his vision of real "seamless multimedia communications." With more than 200 refereed publications, he has become an ICCC Governor in 1999 and was awarded the ranking of fellow of both the IEEE and the ACM in 1999 and 2002, respectively.