[LHRS07] *Lasse Lehmann, Tomas Hildebrandt, Christoph Rensing, Ralf Steinmetz*; ***Capturing, Management and Utilization of Lifecycle Information for Learning Resources***; E. Duval, R. Klamma, and M. Wolpers: Creating New Learning Experiences on a Global Scale, Second European Conference on Technology Enhanced Learning, (EC-TEL 2007), September 2007, S. 187-201

Capturing, Management and Utilization of Lifecycle Information for Learning Resources

Lasse Lehmann[1], Tomas Hildebrandt[1], Christoph Rensing[1], Ralf Steinmetz[1]

[1]KOM - Multimedia Communications Lab, Technische Universität Darmstadt, Merckstrasse 25, 64283 Darmstadt

Abstract. During their lifecycle, Learning Resources undergo a multitude of processes while being created, used, provided or re-used. However, in order to be reusable, a Learning Resource often has to be adapted to a new context of use. This in turn implies multiple Re-Authoring processes being performed on the Learning Resource. During all these processes different types of information emerge. When captured, this information can be helpful for a later on retrieval, use or re-use of the Learning Resources. In this work, the lifecycle of Learning Resources along with the information being generated herein is analyzed and a distributed architecture is proposed, that allows the capturing, processing, management and utilization of the named information in a generic way.

1 Introduction and Motivation

During their lifecycle, Learning Resources undergo a multitude of processes while being created, used, provided or re-used. All these processes generate information about the Learning Resource that is not taken into account in most systems. Besides, it is a widely accepted fact, that Learning Resources should be re-used in order to be efficient. However, a re-use of Learning Resources "as is", i.e. in an unchanged and not adapted shape is seldom possible. Learning Resources are mostly created in a specific context and with a high granularity. From a didactic point of view this surely makes sense [1]. However, the re-use of these resources is quite difficult. Usually it is inevitable to edit the Learning Resources, change or remove parts of them, add parts of other Learning Resources, update them or adapt the Learning Resources to new facts in order to re-use them [4] [13]. All these actions are subsumed by the concept of Re-Authoring and defined, described and classified in [13]. In the course of Re-Authoring processes a multitude of information about the resources taking part in these processes emerges and can be captured. Especially the relations that result from Re-Authoring processes are not considered by most existing approaches. Often, an adaptation or Re-Authoring of a Learning Resource is seen as the starting point of a new lifecycle of a new Learning Resource without taking the relations that connect both instances into account. Nevertheless, there is a multitude of situations where this additional information can be helpful. However, the captured information has to be organized and stored in order to be processible. Most existing systems do not support the capturing and storage of lifecycle information in a sufficient way and if so, the information gets stuck at system borders. This paper proposes a system for the capturing, management and utilization of lifecycle information beyond system borders.

In section 2 of this paper our definition of a Learning Resource's lifecycle is presented and the information that is generated when a Learning Resource proceeds through this lifecycle is analysed. Additionally, possibilities for the utilization of lifecycle information are depicted. Section 3 addresses the storage of lifecycle information and proposes an extension for the well known LOM standard [8]. In section 4, a comprehensive architecture for the capturing, management and utilization of lifecycle information is described while section 5 covers the implementation of the proposed system in the course of the Content Sharing project [3]. Section 6 handles related work in this area and section 7 concludes this paper and gives an outlook on ongoing and future work on this topic.

2 Lifecycle Information

In this section the lifecycle of Learning Resources is analysed. Starting there from we identify two general types of lifecycle information that occur in the different stages of the lifecycle: *Relation* information and *context* information. After taking a closer look on both types, we close this section with an analysis of methods and approaches to utilize both kinds of information.

2.1 The Lifecycle of Learning Resources

In Figure 1 the lifecycle of a Learning Resource following our definition is shown. Learning Resources are created with authoring tools (*Authoring Phase*), before they are provided to customers, teachers or learners, e.g. in a learning object repository (*Provision Phase*). Finally they are used and utilized, which typically takes places in a learning management system (*Learning Phase*). However, in the majority of cases, the Learning Resources available in a repository do not fit the special needs of most customers or users who search for Learning Resources in repositories. In order to be reusable the Learning Resources have to be adapted to a new context of use. To cover this, the *Re-Authoring Phase* is introduced to the lifecycle model. By using Re-Authoring tools, existing Learning Resources can be unitized, adapted, updated and re-aggregated. Parts can be added to or removed from a Learning Resource or parts of different Learning Resources can be joined to form a new one. In Figure 2 an example for this whole process is depicted. At first, the existing Learning Resource E is disassembled. Thus, four new Learning Resources are generated (A, B, C and D). In this scenario, two of the newly generated resources are adapted to a new context (Adaptation), before the parts are put into a new order (Permutation) and, together with a new Learning Resource (H), put together to form the Learning Resource E' (Aggregation). This process is called Re-Purposing, because the original Learning Resource is changed to suit a new purpose. Re-Purposing is a special kind of Re-Authoring [7]. We distinguish two kinds of information being generated during the lifecycle of a Learning Resource: *Relation* and *Context* information. In the following we will discuss both concepts and analyse in which phases of the lifecycle which kinds of information occur.

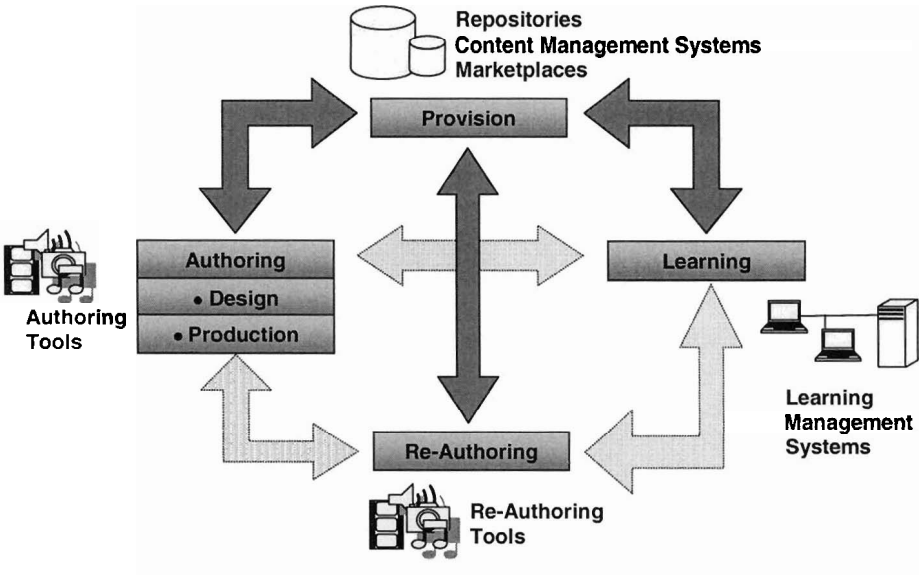

Figure 1: The Lifecycle of Learning Resources

2.2 Relation Information

Relation information emerges in consequence to specific authoring or Re-Authoring actions performed by a user. A relation always connects two instances of a Learning Resource to each other. Regarding the lifecycle shown in Figure 1, relation information is mostly generated in the Authoring and the Re-Authoring phase, since these are the two phases where the content of the Learning Resources is actually changed. In some cases relations are build in other phases, too, 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, being generated during the Authoring and Re-Authoring phase of the lifecycle model. These relation types are described in the following, before they are correlated to certain (Re-) Authoring actions.

Aggregation Relations or 'part of' relations result from the composition of several Learning Resources in order to get a new Learning Resource. Each of the composed Resources has than a 'part of' relation to the latter.

Sequence Relations exist between Learning Resources with a certain sequential order. Two consecutive Learning Resources are connected by a predecessor or successor relation respectively.

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

A Reduction / Extension Relationships occurs, when parts are removed from a Learning Resource. In that case the two versions of the Learning Resource are connected by an isReductionOf or rather isExtensionOf relation.

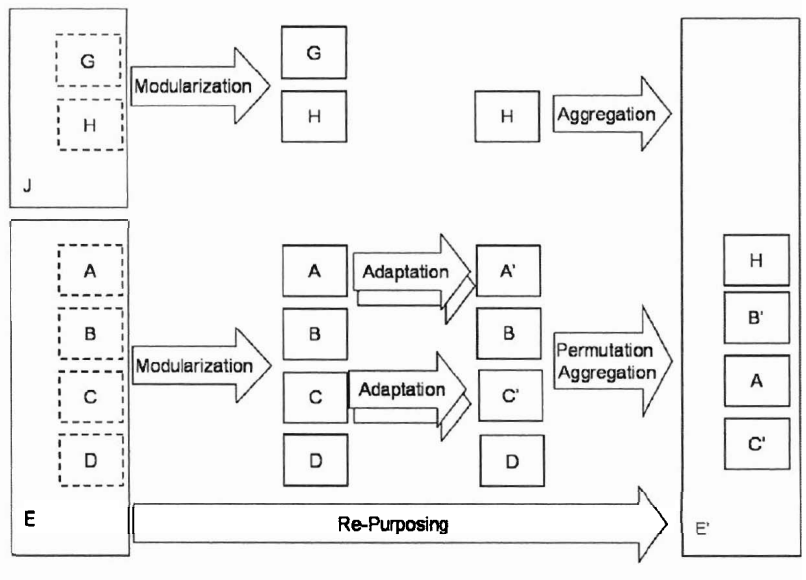

Figure 2: Re-Purposing Process

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 each other. These relations occur, if a Learning Resource is only slightly changed, for example in order to correct errors or to update facts.

Variant Relations persist between different variants of a Learning Resource. Variants are branches in the version history of a Learning Resource. They mostly result from adaptation processes, like translations, layout adaptations or changes in the design of a Learning Resource, which transfer the Learning Resource to a different context of use.

In [11] and [16] Re-Authoring processes in general, as well as adaptation processes in particular, are defined. This encompasses 15 different adaptation processes used most often in practice. Each of these Re-Authoring and adaptation processes implies a certain type of relation information being generated. Table 1 gives an overview of these processes along with the information that occurs.

Table 1: Re-Authoring 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, provide Printability, provide Accessibility, Layout etc. see [16]	Variant, Reduction
Aggregation	Aggregation	Reduction / Extension / Permutation / Aggregation / Sequence

Updates and *Corrections* are Re-Authoring processes often performed, which implicate the creation of version relations. Examples for user actions inflicting these relations are the adaptation of the content to new circumstances like the introduction of the Euro currency or the correction of errors.

The *Modularization* of a Learning Resource implies that it is decomposed into a certain number of modules. Modules are, by definition, Learning Resources by themselves. The decomposition accounts for the generation of relation information like aggregation relations between the decomposed Learning Resource and each of the modules and sequential relations between consecutive modules.

The *Adaptation* of a Learning Resource implies – as mentioned above – a Variant relationship between the source and the target Learning Resource of the adaptation process. A variant relation implies a heavier change of the Learning Resource than a version relationship does. A variant has its own version history that goes parallel to the other one. Version and Variant relationships are a little bit fuzzy by nature and very generic. Therefore the kind of adaptation that led to the generation of a variant or version should be taken into account, too. Thus the variant relation needs to by typed. The actual type of adaptation can easily be captured during the Re-Authoring process. It just has to be stored in a proper way.

Finally, the *Aggregation* process implies different types of relation information, like aggregation, permutation, reduction, sequence or permutation relations. In fact most of these relations occur during the whole Re-Authoring process, but since the Aggregation is often the last step of the whole process, the named relations can not be captured until the final aggregation has been performed.

2.3 Context Information

While relation information always connects two or more Learning Resources, context information is restricted to one Learning Resource and thus represents the context of one specific Learning Resource. It is generated implicitly, mainly during the usage or retrieval of a Learning Resource. Thus, it is mostly generated during the *Provision* and *Learning Phase*. An example for context information is the number of views a Learning Resource got in a repository or market place. Accordingly, the number of downloads or the number of times a Learning Resource was sold could represent valuable information. In a community based scenario ratings, comments or feedback messages are context information, too. In the Learning Phase several kinds of context information can be collected. The learning duration a learner took to learn a Learning Resource, the assessment statistics or the number of students who viewed or even failed the assessment of a certain Learning Resource are only few of the many different types of information that can be collected. Naturally even in the authoring or Re-Authoring phase there is context information to capture, e.g. the time a Learning Resource has been edited, which editor was used or by whom it was edited. Even feedback from learners or other authors related to a Learning Resource is context information in our opinion. The concept "Attention Metadata", sometimes called "Contextualized Attention Metadata", is comparable to what we call context information (see section 6).

2.4 Utilization of Lifecycle Information

There are several possibilities for the utilization auf lifecycle information. The identification of new ways to support authors, learners, providers or just plain users of Learning Resources is an ongoing process. We implemented capturing mechanisms for lifecycle information in our combined repository / authoring tool ResourceCenter [7]. In [9] we present several utilization approaches, including the ranking of Learning Resources and the provision of links to related Resources. There are several approaches trying to help in finding resources that are somehow structurally or semantically related to a target resource. This can be made a lot easier, if the relations between Learning Resources are actually captured when they emerge. Figure 3 shows an example for the utilization of relation information we implemented in the ResourceCenter. Here, related Learning Resources and instances are linked on the overview page of a Learning Resource.

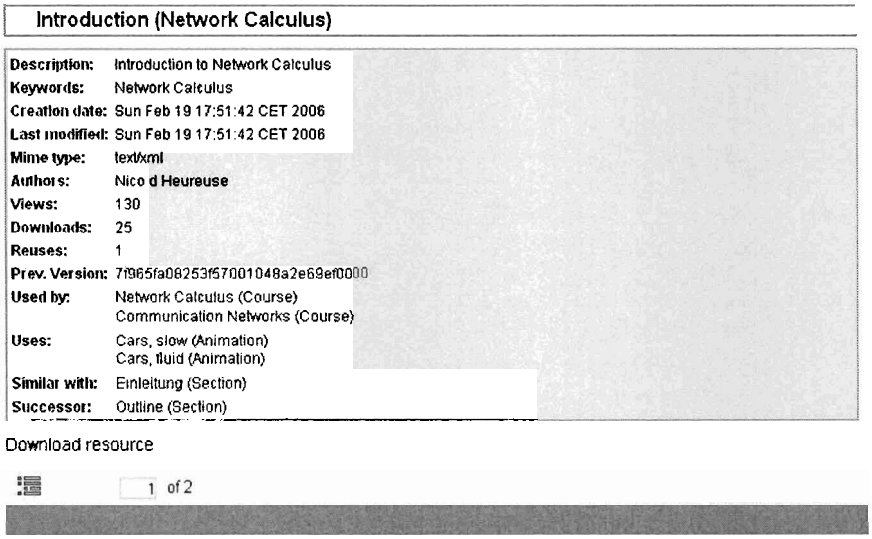

Figure 3: Utilization of Relation Information in the ResourceCenter

A few examples and ongoing work regarding the utilization of lifecycle information in general is given in the following:

- Ranking search results for Learning Resources [9], [12]
- Recommendations for Learning Resources [12]
- Browsing along relations for better search results [9]
- Searching 'without' search terms [9]
- Finding structurally and semantically related Learning Resources (ongoing work)
- Extending information about learner behaviour for efficient learner modelling [10], [11]

- Support authors in authoring by aggregation by recommending Learning Resources with the help of the aggregation context (ongoing work)
- Collecting and providing feedback from learners and other authors to the authors of Learning Resources [14]
- Notification of authors about interesting updates or new Learning Resources through relation and context information (ongoing work)
- Update and consolidate metadata with context information (ongoing work)

3 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.

Since the Learning Object Metadata (LOM) Standard [8] is a widely used standard for metadata, we decided to use it as basis for the storage of relation based lifecycle information. LOM metadata consist of 9 categories with about 60 fields. The category that matches our interests best is category 7: Relation. In this category the storage of relation information is intended. It may consist of an arbitrary number of relation fields containing the ID of the resource the relationship exists to and the type of relation. However, the vocabulary, taken from the Dublin Core standard [15], that is intended to express the relationships in the LOM relation category is not sufficient to fulfil our needs specified in section 2. Therefore we developed our own vocabulary, which enables us to express our types of relations. Table 2 shows how the different kinds of relations are named.

Table 2: Relation types and their vocabulary

Relation Type	**Vocabulary of the LOM Extension**
Aggregation	haspart / ispartof
Sequence	ispredecessorof / issuccessorof
Permutation	ispermutationof
Reduction/Extension	isreductionof / isextensionof
Requirement	requires / isrequiredby
Version	hasversion / isversionof
Variante	hasvariant / isvariantof

As shown in section 2.2, variant and version relations need to be typed. Therefore we need to figure out a unique mapping to a certain aspect of change without changing the underlying LOM standard too much. The LOM standard itself is very rich and covers many aspects in respect of content. Therefore we can relate to these aspects and thus stay independent from the content of the Learning Resource itself. For this purpose, the relation category was extended by one field named *Changes*. This field exists for every relation and consists of a pointer and a value. The pointer points to the LOM metadata 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 implied which changes. In turn it is also possible to include the changes in the original Learning Resource in order to determine, which changes have been done to it, too. While it is especially helpful to type variant relations, other kinds of relations can be typed, too. The following figure shows an excerpt of a relation element depicting a reduction relation resulting from a change of the semantic density and learning duration of a Learning Resource.

```
<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>educational/typicalLearningTime</dataelement>
       <oldvalue>PT30M</oldvalue>
   </changes>
   <changes>
       <date/>
       <categorie>educational</category>
       <dataelement>educational/semanticalDensity</dataelement>
       <oldvalue>medium</oldvalue>
   </changes>
</relation>
```

Figure 4: LOM Extension Excerpt

While relation information is independent from the application or context it was generated in and therefore it makes sense to store it within the metadata close to the Learning Resource itself, *context information* is highly dependent on the system it emerges in. That means that for example the number of views or queries for a Learning Resource in one marketplace might have a different meaning than in another marketplace or repository, due to the number of users, the number of Learning Resources provided or the target group. Therefore context information is stored in an independent format in a central instance. The schema for the storage of context information includes the identifier of the Learning Resource, the type of information - like sold, bought, downloaded, viewed, ... - and finally an identifier for the system the information was captured in. Thus it is possible to weight the captured context information accordingly.

4 Architecture

Form the different types of systems covering the different phases of a Learning Resource's lifecycle, like authoring tools, repositories or learning management systems, there are few, where lifecycle information is captured at all; and if it is captured, the captured information remains in these systems and gets stuck at system borders. Examples for information already being captured by some systems are usage and assessment information in learning management systems or the number of downloads or purchases in repositories or marketplaces. However, if the Learning Resources are transported via system borders, this information gets lost. This is due to the lack of a standardized format for the storage and management of this kind of information. The goal of the proposed architecture is therefore to enable the storage, capturing and utilization of lifecycle information beyond system borders. To achieve this, we propose an architecture with a central component called *Lifecycle Information System* and distributed *Capturing* and *Accessing Components*. The different components are described in detail in the following sections.

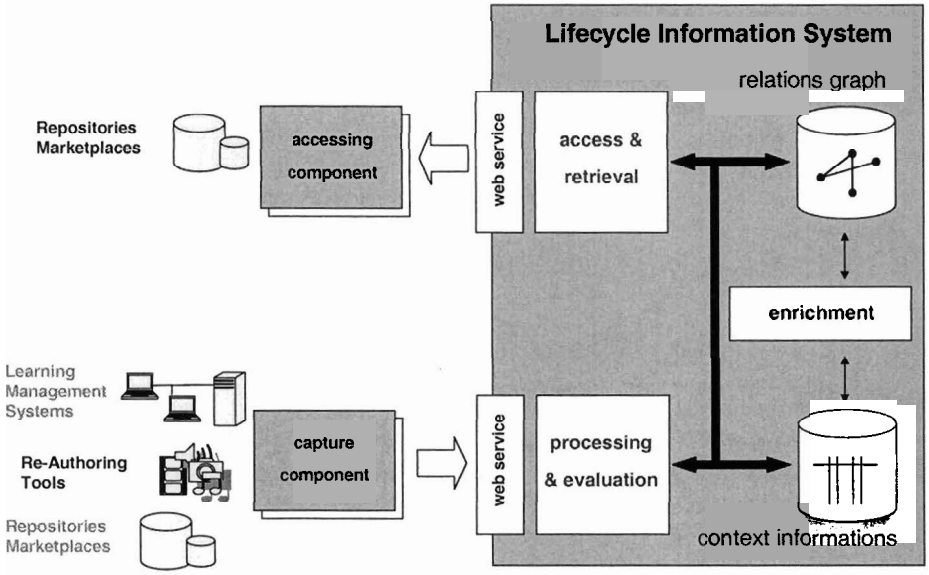

Figure 5: Lifecycle Information System

4.1 Lifecycle Information System

The components of the LIS are shown in Figure 5. The LIS is designed as an online central application and is interconnected to the capture and accessing components via web services, i.e. the components use web services running on the LIS to send the information collected for a certain set of Learning Resource to the LIS or to retrieve it from there respectively. We decided to have the LIS centralized, because this allows

us to easily connect to it and have the information being updated. The LIS is the one central system, which integrates all the data sources and manages the incoming data. In the LIS the gathered information is processed, evaluated and stored in a database. The separately stored context and relation information is enriched by making use of a special rule set (e.g. if A isPartOf B and B isPartOf C then A isPartOf C etc.). The enriched information about a Learning Resource can be retrieved by the accessing components as an XML document.

4.2 Capture Components

The distributed Capture Components are integrated into the tools that are used for creation, usage and modification of Learning Resources. A capture component monitors the creation, change and usage processes and extracts the necessary information generated during these processes on an event handling basis. One feature of the capture component is that it does not require a persistent connection to the LIS. It may work in an offline mode, in which it caches the captured lifecycle information. When a connection is established again, the cached information is transmitted to the LIS. As a fallback solution – in case an online connection can never be established – the lifecycle information may be attached as metadata to the Learning Resource; the Resource itself then serves as the transfer medium. Figure 6 shows the component diagram of a capture component. The core component is connected to the application where the information is captured by a generic interface.

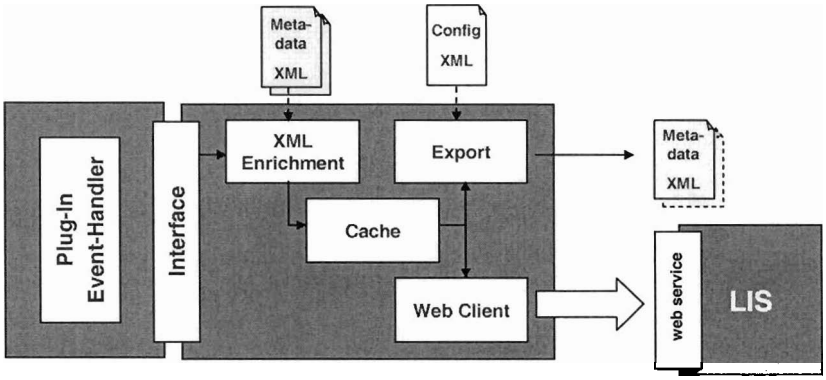

Figure 6: Capture Component

The captured information of a certain Learning Resource is merged with already existing lifecycle information about it received from the LIS to enrich its existing metadata. The cache is used for caching the gathered information on the local computer, where the application is in use, while the web client connects to a web service of the LIS to send the information there, if a connection is available. If this is not the case, the information is cached until a connection is available and can be utilized by the applications on the local computer. Besides that, there is also the possibility to export the information to a local serialized metadata representation (usually XML), which can be

configured during runtime by an external configuration file. The capturing itself is realised by plug-ins which connect to the generic interface of the core component and collect the usage or modification data on an event-handling basis. The LIS collects all information that is related to any version of a Learning Resource. From these individual facts, a lifecycle record, for that Learning Resource is built. This lifecycle record contains a representation of all instances (versions and variants) of the Learning Resource, relations between the instances and context information for each instance.

4.3 Accessing Components

Accessing components are plugged into applications where the gathered 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 accessing components retrieve the information about the Learning Resources via a web service interface from the LIS and present it in a helpful way. That might for example be by means of providing links to closely related Learning Resources or ranking a search result on the basis of the collected context information (see section 2.4). Accessing components, as well as capture components, are designed as plug-ins for applications being used. However, for most applications it makes sense to have not exclusively an accessing or capture component plugged in, but both of them. While searching in a repository, the captured information about the Learning Resources contained in the repository is especially helpful, so that an accessing component for the utilization is mandatory. Additionally there is information being generated while using the repository as well: The selection and access of Learning Resources will increase the selection and access counters and provide additionally information on the relative significance of certain Learning Resource instances. On top of that it is possible to use lifecycle information even, when no internet connection is available. A local accessing component can make use of the local cache of the capture component as a data source.

5 Implementation

We have implemented the above architecture as a proof of concept in the course of the Content Sharing project [3]. Figure 7 shows which components are implemented at the moment and how. Capture components are integrated in the developed Module Editor as well as in the Content Sharing Repository. An Accessing Component in the repository helps users in finding, searching and retrieving the Learning Resources they want. And finally the Lifecycle Information System is implemented, too. For this implementation the local caching features of the Capture Components are used to transfer the captured data via the Content Sharing Repository to the Lifecycle Information System. The Module Editor is a combination of modularization, aggregation and adaptation component. Currently the adaptation processes shown in Figure 8 are supported. Hence the Capture Component theoretically captures all the relation information described in section 2.2. Although this is only possible if the according processes are actually performed. At the moment this Capture Component captures relation information only.

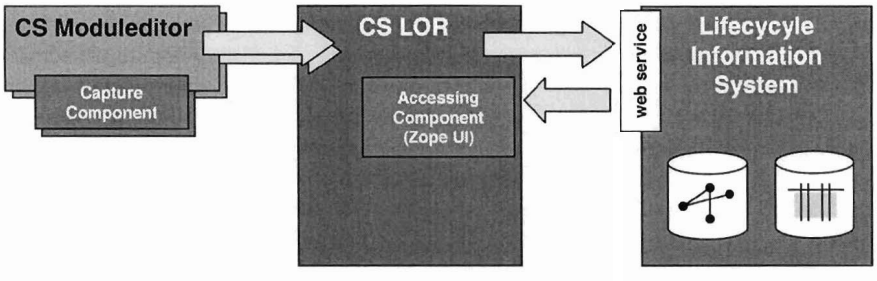

Figure 7: Implemented Components

The Capture Component integrated into the Content Sharing Repository captures context information like the views or purchases of the Learning Resources in the repository, while the Accessing Component processes all the captured relations and provides links to related Learning Resources. For the sake of the utilization of lifecycle information, this implementation is far from complete, but serves as a proof of concept.

An example **usage scenario** with this system looks as follows: A Learning Resource is downloaded from the Content Sharing Repository, while the Capture Component of the repository counts the view and the download towards the context information of this resource. It is opened in the Module Editor and modularized into smaller Learning Resources. The Capture Component in the Module Editor captures the aggregation and sequence relations between the existent resources and stores it with the metadata of the Learning Resources in the local cache (or sends it to the LIS).

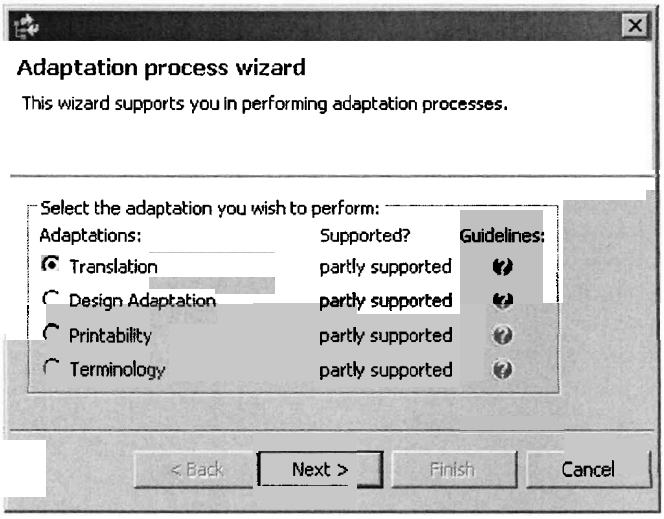

Figure 8: Adaptations of the CS Module Editor

Some of the sub modules are adapted, which inflicts relation information being captured as well, and even newly created Learning Resources are added. Finally the Learning Resources are aggregated and the corresponding information captured. The author can decide which of the created Learning Resources he wants to upload to the repository. He may upload any created or adapted Learning Resource or he may choose to upload the final Learning Resource only. The relation information is transported to the repository either via the metadata of the Learning Resources or via the central LIS. In the current implementation the former is the case. In the repository, the Accessing Component gets the information from the metadata or the LIS respectively and processes it to generate links or recommendations. Before this is done, the information is filtered to show links to resources only, which truly exist in the repository.

6 Related Work

Regarding the capturing of information during the lifecycle of a Learning Resource there is a similar approach called *Ecological Approach* [2], [10]. This work also constitutes that information should be gathered during the actual use of a Learning Resource and not during explicit labelling phases. However, the authors focus mainly on one phase of the lifecycle to gather information about the learner to support their approach to learner modelling. Relations between Learning Resources or other phases of the lifecycle than the actual learning phase are not taken into account.

The utilization of *context* information in a wider sense is conducted for several years now in known applications like eBay or Amazon as well as in many works in the information retrieval research area [6], though there are hardly any works that concentrate on the capturing of information during its creation.

Attention Metadata is a kind of context information about the attention a user pays to different Learning Resources via different applications. It can be gathered and utilized to receive information about the user's experience [11], or actually be used for the retrieval of Learning Resources [12]. For the capturing of Attention Metadata a similar approach involving plug-ins and a central instance is used. While at first only the actual usage of an object was considered by the approach the authors propose in [10] that the creation phase as well as re-used components should be taken into account, too. Therefore the CAMS system is very closely related to our work. However, they do not consider relations between different Learning Resources and instances of Learning Resources.

A system where *relation* information is used is the *HyLOS* system [5]. HyLOS is a Learning Management System in the first place and the user or semi automatic generated relations are used to provide additional links to learners in order to enable a constructivist learning style. The method to store the relation information is mainly based on the named Dublin Core extension for the LOM relation category. The relations used in HyLOS are on a semantically higher level than the relations taken into account by our approach, although it would be nice to have the approaches merged. One of the core features the HyLOS system provides is the relation enrichment. An existing set of relation is processed involving a certain rule set to generate new relations. This is a quite the same what the enrichment component in the here proposed LIS does.

7 Conclusion and Future Work

In this paper we have shown that lifecycle information of Learning Resources can be helpful in many ways. The information generated in the different phases of a Learning Resource's lifecycle was analysed and structured. We proposed a system that supports the capturing, management and utilization of lifecycle information and enables us to collect and use the information in all phases of the lifecycle. The implementation shows that this system is possible and practical. To proof its efficiency, evaluations with user groups still have to be conducted. The generic, plug-in based architecture enables us and other developers to extend the number of supported applications easily. In future one major step, besides the evaluation, will be the development of further plug-ins for capture and accessing components for different types of applications. Additionally, with the growing availability of information about Learning Resources new ways of utilization will emerge that have to be implemented to support learners, authors, providers or just plain users of Learning Resources in different ways. We are currently trying to widen the focus of this approach and not consider Learning Resources only but knowledge documents in general, as well.

8 Literatur

1. Baumgartner, P. (2004). The ROI Paradox. Keynote Presentation at the Gesellschaft für Medien in den Wissenschaften Conference held in Graz, Austria. http://www.c3-initiative.info/peter/2004/09/12
2. Brooks, C. & McCalla, G. (2006),Towards Flexible Learning Object Metadata, *in* 'In Proceedings of Int. J. Cont. Engineering Education and Lifelong Learning, Vol16, Nos 1/2'.
3. Content Sharing Project, http://contentsharing.com, 2007
4. Duval, E.; Hodgins, W. (2003): A LOM Research Agenda. In: (Hencsey, G.; White, B.; Chen, Y.; Kovacs, L.; Lawrence, S., Hrsg.) Proceedings of the twelfth international conference on World Wide Web, 2003; Seite 1-9.
5. Engelhardt, M.; Hildebrand, A.; Lange, D. & Schmidt, T.C. (2006), Semantic Overlays in Educational Content Networks, *in* 'Proceedings of TERENA Networking Conference 2006'.
6. Good, N., Schafer, J.B., Konstan, J.A., Borchers, A., Sarwar, B., Herlocker, J. & Riedl, J. (1999), Combining Collaborative Filtering with Personal Agents for Better Recommendations, *in Proceedings of the 'AAAI/IAAI*Ecological Approach
7. Hoermann, S.; Hildebrandt, T.; Rensing, C. & Steinmetz, R. (2005), ResourceCenter - A Digital Learning Object Repository with an Integrated Authoring Tool, *in* 'Proceedings of the Edmedia 2005'.
8. IEEE Learning Technology Standards Committee (2002): IEEE Standard for Learning Object Metadata 1484.12.1
9. Lehmann, L.; Rensing, C. & Steinmetz, R. (2007), Lifecycle Information Management and Utilization in an Authoring by Aggregation Environment, *Accepted at* Edmedia 2007.
10. McCalla, G. (2004), 'The Ecological Approach to the Design of E-Learning Environments: Purpose-based Capture and Use of Information About Learners', *Journal of Interactive Media in Education* 7.
11. Najjar, J.; Wolpers, M. & Duval, E. (July 2006), Towards Effective Usage-Based Learning Applications: Track and Learn from Users Experience(s), *in* 'Proceedings of the IEEE ICALT 2006'.

12. Ochoa, X. & Duval, E. (2006), Use of Contextualized Attention Metadata for Ranking and Recommending Learning Objects, *in* 'Proceedings of the CAMA 2006'.
13. Rensing, C., Bergsträßer, S. Hildebrandt, T., Meyer, M., Zimmermann, B., Faatz A., Lehmann L. and Steinmetz, R.; Re-Use, Re-Authoring, and Re-Purposing of Learning Resources - Definitions and Examples. Technical Report, Technische Universität Darmstadt, 2005.
14. Rensing, C.; Tittel, S.; Lehmann, L. & Steinmetz, R. (September 2006), Ein System zur Realisierung expliziten Lerner-Autor Feedbacks im E-Learning, in 'Proceedings of the Workshop "Effiziente Erstellung von E-Learning Content" at the DeLFI 2006'.
15. Request for Comment (RFC) 2413, Relation Element Working Draft (1997) http://dublincore.org/documents/1997/12/19/relation-element/
16. Zimmermann, B.; Rensing, C. & Steinmetz, R. (2006), Format-übergreifende Anpassungen von elektronischen Lerninhalten, *in* 'Proceedings of the DeLFI 2006 - die 4. e-Learning Fachtagung Informatik 2006'.