AN ONTOLOGY-BASED FRAMEWORK FOR AUTHOR-LEARNING CONTENT INTERACTION

Saša Nešić\(^1\), Dragan Gašević\(^2\), Mehdi Jazayeri\(^1\)

\(^1\)University of Lugano, Switzerland
\(^2\)Simon Fraser University Surrey, Canada

sasa.nesic@lu.unisi.ch, dgasevic@sfu.ca, mehdi.jazayeri@unisi.ch

ABSTRACT

This paper presents an ontology-based framework for capturing information about interaction between authors and learning contents. The approach is primarily focused on keeping track of changes made to a learning content as well as information about authors and tools, which take part in the interaction. We introduce the Author-Learning Content Interaction (ALCI) ontology, which gives a formal representation of changes and describes possible aspects of the interaction. Moreover, we discuss how information collected with the framework can assist authors during the authoring process and how it can support evolution and versioning of a learning content.

KEY WORDS: learning objects, content authoring, Semantic Web, interaction, content evolution

1. Introduction

The reusability and modification of learning materials and the ability to dynamically assemble them into new learning content that satisfy the requirements of a particular learner are important prerequisites for successful e-learning [1]. E-learning needs to be fast, just-in-time, low-cost and provide relevant experience [2]. In order to be easily reused in multiple contexts, easily modified to meet user-specific needs, and successfully delivered over the Internet, learning materials are built and organized as elements of a new type of computer-based instruction grounded in the object oriented paradigm [3]. These elements are known as learning objects (LOs). Generally, LOs are understood to be digital entities, which unlike traditional educational material, can be easily accessed and modified.

As authoring high quality LOs has proven to be a highly expensive task in terms of both time and money, reuse and modification of existing learning content (LOs) is a very common phenomenon among content authors. However, information about how learning content has been changed over time, the nature and structure of the changes, the authors responsible for changes, the authoring tools that have been used, and all other aspects of interaction between author and learning contents are rarely captured and stored. Some authoring tools have tracking services (e.g., MS Word), but they track semantically poor, general information about the authoring process and history of resource modification. Such logs usually focus on specific systems and provide only limited intra-system interoperability. Besides information about the interaction between authors and learning contents, data about user experience with learning content can also support choosing appropriate learning content and its modification. This data is known as attention metadata [4] and provides the means to track rich information about the learner, the system she/he accesses, the tools she/he employs and information she/he accesses.

In this paper, we focus on author-learning content interaction (ALCI) and how such interaction affects evolution and versioning of LOs [5], so that we can build a foundation for author recommender systems and more efficient version management systems. However, our goal is not only to capture interaction of one specific author that uses only one authoring tool. On the contrary, we want to stimulate collaboration and encourage sharing experiences among authors working on the same LOs. This means that we not only have to enable sharing learning objects, but also sharing (meta)data describing interactions between authors and learning content and relationships between various versions of LOs. Considering the nature of Semantic Web ontologies as means for explicit representation of shared knowledge [6], they seem as a promising solution to this problem. Following this characteristic of ontologies, we propose a Semantic Web ontology-based framework that intends to cover the majority of aspects of ALCI.

The heart of the framework proposed is the ALCI ontology. Having in mind that this ontology should be aware of how learning content is structured, the ALCI ontology is closely related to a learning object content structure ontology. In the next section, we briefly describe such a learning content structure ontology, which we have adapted to meet our needs. Section 3 gives an overview of interaction scenarios and describes types of changes that can occur to the learning content, structured by the chosen content structure ontology. A detailed description of the ALCI ontology is given in section 4. The ALCI framework, and the benefits it can give to the authoring process, and the process of learning content evolution and versioning, will be outlined in section 5.
2. Learning Object Content Ontology

The idea of using ontologies in e-learning is not new, and different authors proposed the use of ontologies for solving different tasks in e-learning. Here we refer to most commonly used classifications of e-learning ontologies that distinguishes between [6]: content (domain) ontologies enabling one to formally state what the learning material is about; context ontologies providing means to formally state in which form the learning content is presented; structure ontologies formalizing the structure of the learning material. Since we aim at defining an ontology for ALCI, in the rest of this section, we explain the last type of ontologies. We refer to it as a learning object content structure ontology.

The purpose of a learning object content structure ontology is to provide an explicit definition of a LO content structure, formally specifying both content components and the relationships between those components [7]. In order to be able to follow the modification process and capture changes made in a LO, we have to know the structure of the LO as well as the structure of its components. One attempt to address this problem is the Abstract Learning Object Content Model (ALOCoM) [7], and the ontology developed on top of it, which together provide a common representation for LOs and their components. Employing this model, in our approach to the conceptualisation of ALCI, we are able to follow not only the interaction between an author and the entire LO, but also between an author and the LO’s components. By capturing changes made in a LO’s components, we gather more broad and detailed information about modified learning content.

The ALOCoM Content Structure ontology (ALOCoM CS) [1] defines a number of concepts for different types of content units that form the structure of a LO. It distinguishes between content fragments (CFs), content objects (COs), and learning objects (LOs):

- **Content fragments** are atomic learning content elements. They can be regarded as raw digital resources and specialized into discrete (graphic, text, image, etc.) and continuous (audio, video, simulation, etc.).
- **Content objects** aggregate content fragments and add navigation. A content object (e.g., title, paragraph, and body) can also include other content objects.
- **Learning objects** aggregate content objects around a single learning objective. Examples of LOs are a slide presentation or a text document.

All three concepts are sub-concepts of the more general **Content Unit (CU)** concept.

The ALOCoM CS ontology meets the majority of our needs well. However, we have had to extend it, so that the remainder of our goals could be achieved. This extension is needed to allow keeping track of CU versions and to make the process of content evolution clear and transparent. Each change of CF, CO, and LO creates a new version. In order to meet necessary version-requirements, the CU concept is supplemented with the **hasVID** property, of which the value is the content unit version identifier (VID). In addition to the VID, each content unit is extended with a unique content unit identifier (CUID) as a value of its property **hasCUID**. A change to the CU creates a new version of that CU, and each version is given the VID, while the CUID remains the same as in all previous versions of the CU. Using the CUID and VID ensures that the history and traceability of CU is never lost. However, to make the evolution process a more explicit and to simplify access to the information about the history of content units, we introduce one more property, namely **hasHistoryList**, to the CU class. This property points to a list of VIDs, beginning from the VID of the root CU to the VID of the CU’s first ancestor. How these extensions to the ALOCoM CS ontology are used in our system, and what their benefits are, is discussed in the sections 4 and 5. The changed ALOCoM CS ontology is available at [8].

3. Author-Learning Content Interaction

In our consideration of ALCI, we regard it as interaction between an author and a LO. Using the ALOCoM CS ontology, an author can directly access, modify, and reuse a LO’s components. Accordingly, in our approach we look at this interaction as an interaction between an author and a learning object component (i.e., a CU according to the ALOCoM CS ontology) rather than an author and a whole LO. In practice, CUs are stored in Learning Object Repositories (LORs), while LORs are collections of ontology-based metadata that refer to CUs (i.e. components of LOs) through URLs and provide indexing, categorization, and searching services. Analyzing possible scenarios of interaction between an author and CUs, we have identified the following ones:

1. Author searches the LOR and selects a suitable CU.
2. Author loads the chosen CU into an appropriate authoring tool.
3. Author modifies the chosen CU by using an authoring tool.
4. Author creates a new CU.
5. Author incorporates the chosen CU in a new CU.
6. Author performs semantic annotation (classifies it according to a taxonomy or an ontology) of the newly created or modified CU.
7. Author saves the newly created or modified CU.
8. Author searches the Web looking for appropriate CU, and uploads it directly to the LOR without modifications.
9. Author deletes the CU from the LOR.

Although we consider equally all of the aspects listed above, one of them is essential in a sense that it causes the most implications to both content units and authors, namely, the process of content modification by the author using an authoring tool. We now take a closer look at the set of actions that an author performs during the modification process, in the chronological order:

1. Author chooses a CU.
2. Author chooses an appropriate authoring tool.
3. Author loads the CU into the chosen tool.
4. Author locates the part of the CU that needs to be changed.
5. Author checks if the located part can be modified in the current authoring tool, or if it demands using another authoring tool.
6. Author modifies the located part.
7. Author performs the semantic annotation (by using a domain ontology) of the modified CU.
8. Author uploads the CU back into the LOR.

The main goal of our work is to capture relevant information about the action via which the author interacts with the CU. That information can be roughly split into information about the author, the authoring tool, and the CU. The action performed also has a set of general characteristics such as start-time, end-time, action-type, and duration, all of which can be recorded. The information about the CU includes its identifier, version identifier, and a formal description of all the changes that have occurred in the CU. A set of changes in the CU within an action leads to a new version of the CU. The description of changes should allow the derivation of a transition between CU versions, in both directions, without losing information.

We have developed an ontology-based framework for conceptualizing the interaction between the author and the learning content. Before we explain the framework, we enumerate all of the possible kinds of changes of CUs whose structure is described by the ALOCoM CS ontology.

3.1 ALOCoM CS Allowable CU Changes

Relying on the three basic types of a CU, defined in the ALOCoM CS ontology, we distinguish between:

- **Content Fragment change**
- **Content Object change**
- **Learning Object change**

A *Content Fragment change* is a change made to a CF. This change leads to a new version of the CF. Since the CF is an atomic CU, that is, we cannot disaggregate it into smaller units, a CF change can only be determined by comparing the old and new versions of the CF. For example, if we change an audio record, which is a CF, we are not able to formally describe occurred change. In order to keep track of this kind of change, the presence of both the old and new version is required. By deleting the old version, we lose all information about the changes that translate this version into a new version.

A *Content Object change* is a change made to a CO. We have identified the following types of CO modifications:

- addition of new CFs,
- subtraction of CFs,
- reordering of CFs in a CO, and
- CF changes (changes of a CF that is a part of the CO).

All these types of modification lead to a new version of the CO. It is also worth mentioning that the subtraction of CFs from a CO does not mean that their deletion from the LOR. The assumption is that they can exist independently of the COs in which they are involved.

A *Learning Object change* is a change made to a LO. Possible types of a LO modification are:

- addition of a new CO,
- subtraction of a CO,
- reordering of COs within a LO, and
- CO changes (modification of a CF that is a part of the LO).

These modifications create a new version of a LO. Changing a LO by modifying its COs or changing a CO by modifying its CFs may cause some inconsistencies if these COs and CFs are also included in other LOs and COs respectively. This problem is solved by the fact that both the CO changes and the CF changes, which appear as parts of the LO change and the CO change, respectively, create new versions of the CO and CF, while the old versions remain available for others.

4. ALCI Ontology

In this section, we describe the ALCI ontology that formalizes types of interactions between authors and learning content (whose structure is described by using the ALOCoM CS ontology) identified in the previous section. The ALCI ontology is an OWL ontology and it is available at [8]. Figure 1 gives a graphical representation of the ALCI ontology in a form of a Resource Description Framework (RDF) graph. The class *Action* formally describes the concept of an action that an author performs on a CU. Each instance of the *Action* class has references to instances of the *Author* and *ContentUnit* classes, and is related to a general info-data via the *hasGeneralInfo* property. The general information is formally represented with the *GeneralInfo* class and its data type properties: *actionStart* and *actionEnd*, which keep the values of the date and time when the action started and ended, an optional property *description* that keeps a short description of the action, and the *actionType* property. Considering the wide range of available tools, we distinguished between six possible values for *actionType*: *CUcreation*, *CUModification*, *CUUpload*, *CUDownload*, and *CUBrowse*. In terms of the ALCI ontology, the main type of users is *Author* who performs one of these action types. In this conceptualization of ALCI, we are primarily focused on the *CUModification*. Using captured information about this action we can improve the authoring process and learning content evolution, which is our main goal.

The *CUModification* action starts at the moment when a particular CU is loaded into a chosen authoring tool. The tool, which is used to load the entire CU, will be referenced in the rest of this section as the *main-authoring tool*. As a CU is usually composed of different types of CUs, in order to modify them, we may need to employ other format-specific authoring tools. We call these tools *subordinate-authoring tools*. For example, assume an author intends to modify a slide presentation (LO). He loads it into an authoring tool (e.g. MS PowerPoint) and
on a slide (CU) locates a picture (CF) that she/he wants to modify. Then she/he opens the picture in a graphics editor (e.g., PhotoShop) and modifies it. In this example, MS PowerPoint is the main-authoring tool, while PhotoShop is the subordinate-authoring tool. The action ends when the main-authoring tool has saved all applied changes in a CU and uploaded it back to the LOR.

During a CUModification action, one or more changes may occur in an entire CU and its sub-units. These changes are formally represented with classes: LOChange, COChange and CFChange, which are all subclasses of the class CUChange. As we have already mentioned, each format-specific CU demands a specific authoring tool (application) for its modification. Accordingly, each change is characterized by the application in which it occurred and the time when that application started and ended. These data are formally represented with the Session class and its properties: hasSessionStart, hasSessionEnd and hasApplication. Each instance of the session class also has a unique identifier, the hasSessionId. Introducing the Session concept allows us to capture information about more authoring tools, which are used to modify more CUs at the same time. Each instance of the CUChange class has its unique change identifier hasChangeId, and the information about the content unit identifier (CUID) and the content unit version identifier (VID).

If an author interacts with a CF, then only one CF change is possible during an action, and it is formally described with the CFChange class. As a CF is an atomic CU, its change can only be captured by a pair of references to the new and old versions of the CF. These references are kept as a value of hasOldCF and hasNewCF properties of the CFChange class.

If an author interacts with a CO, then only one CO change is possible during an action. Beside this change, a number of changes can occur in the CFs, which are part of the current CO. The CO change is formally described via the COChange class and its properties. Each instance of the COChange has references to all COs that are added or subtracted as well as references to the old and new CF order lists, which define the order of CFs in the old and new version of the CO. Since each change to the CO leads to a new version of the CO, the COChange also keeps track of the new CU version identifier. Together with a CU identifier (the value of an inherited property from the CUChange), the instance of the COChange has all
necessary information to uniquely determine the new version of the CO.

If an author interacts with a LO, then only one LO change and a number of CO and CF changes are possible. The LO change is formally described via the \textit{LOChange} class and its properties. The \textit{LOChange} class has the same set of properties as the \textit{COChange} class, with the difference of having CFs as building units instead of COs.

5. The ALCI Framework

In this section we described the ALCI framework, which generates, collects, and merges ALCI ontology-aware metadata and based on them makes recommendations to authors during the authoring process. Figure 2 illustrates this framework. Authors usually interact with a wide range of authoring tools while constructing learning content. Examples of such tools are MS PowerPoint, OpenOffice, Macromedia-Dreamweaver, and MS Word that are used to create new learning content by re-aggregating existing ones, by authoring new content, and by modifying existing one. Usually, more then one authoring tools are employed in an authoring process of a CU. As we have mentioned in section 4, we distinguish between two different types of authoring tools depending on their role in authoring process. Those are the \textit{main-authoring tool} and the \textit{subordinate-authoring tool}. The \textit{main-authoring tool} is a tool that loads the entire CU from the LOR and uploads modified or newly created CUs back into the LOR. The \textit{subordinate tool} is a tool that can be used during the authoring of the CU to modify its specific sub-units. There can be only one \textit{main authoring tool} and a number of \textit{subordinate tools}. All of these tools generate ontology-aware metadata described by ALCI ontology (ALCI metadata). That metadata needs to be merged to provide a complete set of information on the interaction between the author and the CU. The metadata collector accepts generated ALCI metadata from subordinate and main authoring tools and supplies the recommender system with merged metadata.

The ALCI metadata provides a rich source of information that can be used in almost all parts of the e-learning process. However, it has the most important implications to the authoring process. Using this metadata along with authors’ profiles, which describes their interests, preferences, likes, and dislikes, recommender system can make recommendations to authors during the authoring process. In this way the authoring process can significantly be improved, both in quality and time. Authors get information that helps them to select the highest-quality learning content, which meets their needs the best, and to determine the most suitable and the fastest way of learning content modification. Besides assisting authoring, this metadata can support versioning of learning content and make the evolution process more transparent. It can also suggest a garbage collection system to delete CUs, which are anymore not useful for authors [9]. If there is, for example, an animation, that has not been used for a long time or there is a new version that replaced it everywhere, the garbage collection system can be suggested to delete this animation.

\textbf{5.1 Implications on authoring process}

The main goal of the ALCI-framework is to improve the authoring process by supplying authors with a detailed history of the learning content. The history is represented by the ALCI ontology-aware metadata. Using this metadata authors can easily get answers to many common questions that they may have during the authoring process. These answers enable authors to choose the most suitable CU for their purpose, to choose an appropriate tool for its authoring and to determine the best method for its modification. By querying the ALCI metadata related with the CU, the following information can be extracted:

- a kind of action in which the CU is used the most,
- average length of modification time,
- the number of modifications and their session time,
- authoring tools used for modification,
- authors who used the CU,
- parts of the CU that have been changed the most,
- the way in which CU’s parts has been changed (e.g., adding, subtracting, and reordering),
- a number of versions of the CU,
- a set of changes that transform one CU version to another, and
- a version of the CU that is the most often used.

\textbf{5.2 Implications on versioning and evolution}

A change to a learning CU (LO, CO, CF) creates a new version of that CU. Versions that are meant to replace previous versions are called revisions while versions that meant to coexist with previous versions are called variants. The set of changes between two versions is referred to as a \textit{delta} and can be captured either as a set of differences between two versions (called \textit{symmetric delta}) or a set of change operations that can be applied to one version to create the new version (called \textit{directed delta}) [5].

Ontology-aware metadata collected by the ALCI framework can support more aspects of the CU evolution and versioning. One of the main version control principles, \textit{sameness} [10], requires a method by which two CUs can be examined to check if they are of the same version. As we explained in section 2, each CU’s version is uniquely determined by its CUID and VID. However, two versions (variants) of the CU can completely be the
same because of different evolution paths they have followed. Having collected sets of changes \((\textit{delta})\), which transformed a CU from its root variant to a targeted variant, it is easy to determine if these variants are the same or not. Two CU variants are the same if and only if they have an identical set of changes between them and a root variant. The ALCI metadata also ensures that the history and traceability of the CU is never lost. This is especially important in the case of revisions because previous versions do not exist any more. Very often, there is a need for previous CU versions, either because of a badly done revision or incompatibility with some applications, which use that CU. By using ALCI metadata, previous versions of the CU can be easily rebuilt and employed. However, it is only possible if some parts of previous version are not deleted from the LOR.

6. Analysis

There are a few approaches that can be compared to our approach. One of them is the Ontology-Based Authoring Support Framework [11], which is based on the ATO ontology. This ontology defines a set of \textit{generic adjectives}, representing the modifications of the domain objects. In this conceptualisation, a LO is seen as a sub-variant of the domain object. However, unlike the ALCI ontology these adjectives are concerned about changes in roles which LOs have in the authoring tasks, rather than changes to the LO structure. There are no correlations with learning content structure ontologies, as well as support for versioning. Comparing with TANGRAM [1], which is also based on the ALOCoM CS ontology, we may say that it is focused on automatic annotation of LOs and their adaptation to the specific needs of individual students, while it does not consider modifications of LOs that occur during the authoring process. Authoring process in TANGRAM is limited on its own authoring functionalities and does not allow other authoring tools to be involved.

We have extended LO version model [5] with a new concept, \textit{hasHistoryList}, which contains information about all LO’s ancestors, and thus speeds up the access to each of them. As a summary remark, we can say that our approach supplements the IEEE LOM standard with versioning schema and description of LO components and changes that may occur to them.

7. Conclusion

In this paper, we have proposed an ontology-based framework, which generates ontology-aware metadata about interaction between authors and learning content. The framework is based upon the ALCI-ontology, which formally specifies types and aspects of interaction and gives a formal representation of possible changes to a learning content. The main rational for having generated ALCI-metadata is to improve the authoring process by assisting authors. Querying this metadata, authors can acquire information that will simplify and improve reusability and modification of existing learning content. This metadata can also serve as a foundation for more efficient version control management system keeping detailed information about history of learning content modification.

In the future, we plan to develop plug-ins that will extend some of the authoring tools (e.g. MS Word, MS PowerPoint, and OpenOffice) with support for creating the ALCI metadata. We will later use this ALCI metadata to develop an intelligent system that will make recommendations to authors during the authoring process. In this way, authors will be able to share not only LOs, but also their experience in the work with LOs as well as information about the history of LOs’ behaviour.

Acknowledgment

The work of the University of Lugano is supported by the EU IST FP6 project NEPOMUK – The Social Semantic Desktop (FP6-027705), while the work of Simon Fraser University is supported by Canada's LORNET Research Network.

References