Ontology-based Domain Modelling for Consistent Content Change Management

Muhammad Javed, Yalemisew M. Abgaz, Claus Pahl

Abstract—Ontology-based modelling of multi-formatted software application content is a challenging area in content management. When the number of software content unit is huge and in continuous process of change, content change management is important. The management of content in this context requires targeted access and manipulation methods. We present a novel approach to deal with model-driven content-centric information systems and access to their content. At the core of our approach is an ontology-based semantic annotation technique for diversely formatted content that can improve the accuracy of access and systems evolution. Domain ontologies represent domain-specific concepts and conform to metamodels. Different ontologies - from application domain ontologies to software ontologies - capture and model the different properties and perspectives on a software content unit. Interdependencies between domain ontologies, the artifacts and the content are captured through a trace model. The annotation traces are formalised and a graph-based system is selected for the representation of the annotation traces.

Keywords—Consistent Content Management, Impact Categorisation, Trace Model, Ontology Evolution.

I. INTRODUCTION

ONTOLOGY-BASED MODELLING of multi-formatted content is a challenging area in content management. Domain ontologies become essential for dynamic information systems and computer science technologies. Organisations are looking into them as machine processable data for many software application areas such as Bioinformatics [1], Educational Technology Systems [2], Web services [3], E-Learning [4], Indexing and Retrieval [5] etc. The Semantic Web and collaborative environments create a high demand for the sharing the semantics of the data. Access to such content is another issue in content management systems.

Software content can be of different types, in terms of their format from a simple text based document to an executable or abstract entity or in terms of other aspects like language. By saying content, we mean digital

information available in a collaborative environment. It could be a source code (such as java script), an executable element (such as applet), commands (such as print), semi-structured text (such as HTML, XML documents), software elements (such as GUI feature) etc. We primarily focus on the representation of content of a software system in semi structured text form such as web files in HTML/XML format.

Knowledge-based software application content is always changing with time. The dynamic nature of software content in every field requires domain ontologies to change over time. The reason for change in ontology can be the change in the domain, the specification, the conceptualization or any combination of them [6]. The change in software content may lead to change in the respective domain ontology and vice versa.

As domain ontologies are subject to real-time content changes, they need to keep themselves consistent. The context has been explored in a number of recent research projects [7], [8], [9], [10]. In our research area domain, *consistency* is a property of holding together all entities of ontology and its artifacts. Thus we can define a *consistent ontology* as one that does not contain a contradiction and the defined integrity constraints are not violated. Inconsistent ontologies may lead to false content access. Some of the changes during content evolution are about the introduction of new concepts, removal of outdated concepts, change in the structures and the meanings of concepts. This requires an effective ontology change management approach.

In this context, ontology-based software application domain models can play a critical role. They can help and guide software evolution, enforce consistency and reduce critical risks of loss of knowledge involved in content management. Our approach is ontology-based domain modelling by taking advantage of semantic annotation technique. Some central features of our approach are

- Digital information modelling technique for software application systems
- Addition of ontological layer at the top of content management layer
- Semantic guided access to the software application content through domain ontologies
- Consistency management between domain ontologies and system components

Muhammad Javed is PhD student at Centre for Next Generation Localisation, Dublin City University, Dublin 09, Ireland. (Phone: 353-1 700 6912; fax: 353-1 700 6702; e-mail: mjaved@ computing.dcu.ie).

Yalemisew M. Abgaz is PhD student at Centre for Next Generation Localisation, Dublin City University, Dublin 09, Ireland. (Phone: 353-1 700 6912; fax: 353-1 700 6702; e-mail: yabgaz@ computing.dcu.ie).

Dr. Claus Pahl is Senior Lecturer at School of Computing, Dublin City University, Dublin 09, Ireland. (Phone: 353-1 700 5620; fax: 353-1 700 5442; e-mail: cpahl@ computing.dcu.ie).

The paper is structured as follows: We discuss our case study on ontology-based modelling of software applications in section 2. In section 3 we discuss the consistency management between domain ontologies and software application content. Related work is discussed in section 4. A short evaluation is given in section 5 and we end with some conclusions.

II. ONTOLOGY-BASED CONTENT MANAGEMENT

Change management of differently formed and structured/semi-structured content is a key focus here. Content centric systems require special solutions for the management and manipulation of the content. The semantic access to the content is another issue of importance. Content can be of different types, however we are mainly interested in structured/semi-structured textual representation of the system components in content management files.

A. Trace Model for Guided Access to the Content

The core of the framework is an ontology-based semantic annotation trace model, as a key approach for adding semantics to the content and their guided access. Figure 1 represents the key components of proposed content modelling framework. As a case study we took a content-centric perspective on the software application system. The application is a content management and archiving system. It contains a number of task components such as Archiving, Searching, Sorting, Messaging etc. We specifically focus on the help system of the software application which contains help files and help management. The help files are linked to the domain ontologies through the trace model. One can find the key entities, such as GUI elements, commands, procedures, role etc. of different software components in these help files. Such entities are identified from the help files and are linked to the respective domain ontologies. We distinguish between software components, annotation trace model and domain ontologies as key artifacts in our case study.
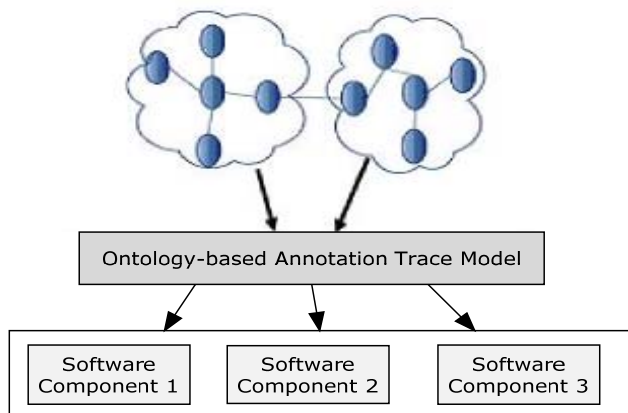

Fig. 1 Annotation Model linking Software Components to the Ontologies

The traces, generated as the result of ontology-based

semantic annotations, will be utilized in two aspects, i.e. querying the software application content for searching relevant information and change management for the dynamic content.

1) Trace-based Information Retrieval: The traces will be used to query the content semantically by taking advantage of the domain ontologies. The query may be in a form such as *How to sort archived folders*. It can be deduced from the query that user is interested in a certain procedure, performed on a concept, available in a certain software component. A query agent will utilize the annotation traces to identify the relevant software component and domain ontologies. Suitable files will be retrieved by accessing the appropriate concepts in domain ontologies through annotation traces and forwarded to the query initiator.

2) Trace-based Change Management: A set of change operations will be offered to perform the changes (such as add, delete) in the annotation traces. They will be used for the activities such as modification, rollback, versioning of traces etc. The multilayered change operations proposed in [9] will be utilized for the operationalisation of ontological changes.

B. Annotation Set Formalisation

An annotation set (AS) consists of an unordered sequence of annotation traces, AS =< A_1, A_2, A_3, $\cdots A_N$ >. An annotation trace structure, linking content to the domain ontologies, is given below in Figure 2. Each annotation trace contains two types of attributes, i.e. Meta attributes (M) and the Trace attributes (T). Meta attributes mainly consists of metadata information about the trace itself. It is in the form of M = (id, u, t) where id represents the Identification Key of the trace, u represents the User who generated the trace, t represents the time the trace was constructed.
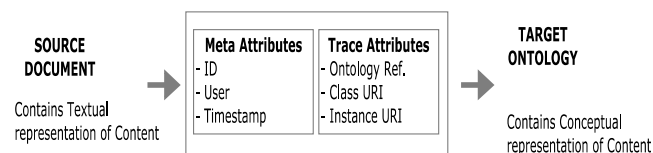

Fig. 2 Annotation Trace, Linking Software Application Content to Domain Ontologies

The trace attributes contain the central information how the source content and the targeted domain ontology are linked to each other. The Source Document (SD) is a tuple of References to the Document and the Type of Content. The Target Ontology is a triple consisting of a Reference to Ontology, a Class URI and an Instance URI.

$$SD = (D_{ref}, TC)$$
$$TO = (O_{ref}, C, I)$$

where SD, TO, D_{ref} , O_{ref} , C, I and TC represents the Source Document, Target Ontology, Document Reference, Ontology Reference, Class URI, Instance URI and Content

Type, respectively. Thus, overall trace attribute set can be given as T = (SD, A_t, TO) where A_t represents the Annotated text.

1) Graph-based representation of Annotation Traces: In order to formally represent the annotation traces, a graph-based representation has been selected [11]. Graphs are preferable as they are simple to implement, simple to explain and can be further refined. They make the facts clearer and more understandable for non-experts. They can be used to represent the facts in visual form. The benefit of graph-based representation is the availability of its well established algorithms, properties and its well known characteristics which can be used for querying and management of annotation traces effectively with a great performance. The graph-based formalisation will be used to store, access and manipulate the traces efficiently in an ontology-based content change management system.

III. CONSISTENCY MANAGEMENT BETWEEN CONTENT AND ONTOLOGIES

The annotation traces will be used to keep the bi-directional consistency between the content and the domain ontologies. That means, if the content or ontology changes at a certain time, the annotation traces will evolve and will provide feedback to other trace components for their necessary evolution process.

We need to keep track of changes in the components of the software system, reflected through the content management layer. These changes must be reflected in the representative domain ontologies. The inter-dependencies between domain ontologies and the content will be captured through Trace Consistency Model. Changes can be made either in the ontology or in the domain content. We will realize changes from both perspectives, i.e. bottom-up, top-down approach (Figure 3). For example, if there is a change in the content, it needs to be reflected by changes to the respective ontologies and the annotation traces. Similarly, if there is a change in the ontology, it needs to be reflected in the annotation traces and in the content, depending on the significance of change. As traces, formed through the semantic annotation scheme, capture central consistency constraints, we will use the traces for consistency management. For the operationalisation of ontology changes, the change operator framework detailed in [9] will be used.

To uniquely identify and keep the traces consistent, one needs to identify consistency constraints for them.

Definition - *Trace Consistency Constraints*: We define the consistent trace as

A single annotation trace is consistent with respect to its trace model if it preserves the constraints defined for the model.

To deal with consistency issues, we have introduced the notion of trace invariants (I). Invariants are must hold consistency constraints for every trace in the set. Every

trace should keep the correctness of the invariant. Some of the invariants are given as follows.

I1: *Unique Identity Invariant*: Each trace must have its own individual identity to be uniquely identified.

I2: *Trace Closure Invariant*: All traces must be an element of the annotation set AS (A_{T1}, A_{T2} · · ·A_{TN} ∈ AS).

I3: *Trace-Ontology Closure Invariant*: Each trace can be connected to at most one concept/instance of the target ontology (1:N).

I4: *Trace-Content Closure Invariant*: Each trace can contain at most one link to the source document (1: N).
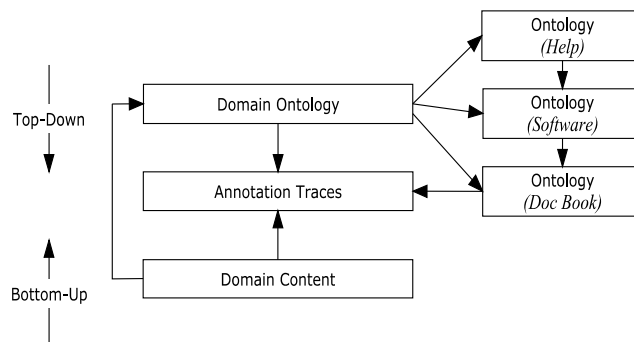

Fig. 3 Top-Down and Bottom-Up Approach for Consistency Management between Ontologies and Domain Content

Following to the invariants, a number of trace manipulation techniques can be set to deal with the consistency issues. For example, a trace manipulation technique can be offered for the 3rd invariant (I3), i.e., if ontology changes list an operation in which a concept, which is currently been used in the formulation of a trace, is deleted, the trace
- must also be deleted or
- the deletion of such concept must not be allowed from the ontology or
- user must be informed about referential integrity before deletion of the concept.

Such flexibility will give a free hand to the user to deal with the consistency issue based on individual requirements.

A. Impact-based Categorisation of Content Changes:

A change in software content or domain ontology may lead to the addition, modification or deletion of relevant annotations. Below in Table 1, we discuss a number of the changes that can occur in the content of a software system and their impact on the annotation traces and ontology.

The effect of changes can be categorized based on the structural or semantic level impact. For example, if a new feature is added to the software system, new concepts will be added in the respective ontology and so new traces will be generated. On structural level, it has a low level impact as no concept in the current ontology will be affected except an addition of a concept and traces in the ontology and annotation set, respectively. If we look at the same

change semantically, it has a high impact on the ontology and annotation set, because a new concept, which was not there before, will be added.

We have categorized the effect of changes based on their semantic level into Higher (H), Medium (M) and Lower (L) types. The semantic level impact is high when a change requires addition or deletion of certain element. Semantic level impact is medium if the content change requires ontology to be edited or if the change is at the instance level. The semantic level impact of change is low if does

not affect the ontology semantically and involves structural ontological changes such as renaming etc. In such cases only annotations have to be added or edited. Impact-based categorisation of ontological and content changes provides information to the ontology engineer to support decisions in content change management.

In Table 1, we discuss a number of the changes that can occur in the content of a software system and their impact on the annotation traces and ontology.

TABLE 1
IMPACT OF CONTENT CHANGES ON ANNOTATIONS AND ONTOLOGIES

Content Changes	Impact on Ontologies	Impact on Annotations	Effect
Addition of a new feature in software system.	New concepts will added in multiple ontologies.	New Annotations will be added	H
Addition of new topic in help file.	New instances will be added in ontology.	New Annotations will be added.	M
Removal of a feature from software system.	Concepts will be deleted from relevant ontologies.	Annotations will be deleted	H
New reference of help file is added.	New attributes to the concept will be added in ontology	Annotation will be edited	M
Content format is changed	No-Effect (if format instance is already available in ontology).	Annotation will be edited	L
A software feature is upgraded to a software component.	Concepts will be generalized.	Annotations will be edited	M
Adding a specialized feature under a component.	New specialized concepts will be added.	New Annotations will be added.	H

TABLE 2
IMPACT OF ONTOLOGICAL CHANGES ON ANNOTATIONS AND CONTENT

Ontological Changes	Impact on Annotation	Impact on Content	Effect
New concept is added	New Annotation will be added	New content is eligible to add.	H
Annotated Concept is renamed.	Annotation will be edited.	No-Effect	L
Concept is deleted.	The annotation will be deleted.	Content is eligible for deletion.	H
Parent Concept of an annotated concept is deleted from the ontology	The effect depends on evolution strategy we follow.		
i. sub concepts will be deleted	Annotations will be deleted	Content is eligible for deletion.	H
ii. sub concepts will be preserved	Annotations will be edited	Content is eligible for edition	M
Annotated Instance is renamed.	Annotation will be edited.	No-Effect	L
Annotated Instance is deleted.	The annotation will be deleted.	Content is eligible for deletion.	M
Parent concept of annotated instance is changed.	The annotation will be edited.	Content will be moved from one location to the other.	M
Class hierarchy is changed in the ontology	Annotation will be edited	Content is eligible for edition.	M
Ontology URI is changed.	The Annotation will be edited	No-Effect	L

B. Impact-based Categorisation of Ontological Changes:

In Table 2, we discuss a number of changes that can occur in the ontology and their impact on the annotation traces and content. We used the word *eligible* with the impact on the content changes, as the content manager will be notified about the ontological changes. Now it is his decision whether to perform the changes in the content or to assess the costs of change.

IV. RELATED WORK

We give a brief summary of current practice in the area of ontology-driven modelling of content, specifically through semantic annotation. The researchers have worked in the area of semantic annotation, mostly annotating the web pages in HTML and XML form, to help in realisation of semantic web. A few others have also worked of annotating multimedia content [12]. In [5], the authors discussed how semantic annotation can be used for generating metadata for the semantic web. Their automatic annotation system, the Knowledge and Information Management (KIM), is based on an upper level ontology (i.e. proton ontology) as a knowledge base. It examines the text and searches for references to the entities. Furthermore, it tries to match these searched entities with the classes and instances available in the proton ontology. Once found, they get annotated with the unique URI of the entity. The authors suggest that such annotated data can be used for indexing and information retrieval activities.

In [13], the authors share the idea of linking the web content dynamically. They suggest that it will be of great benefit for the user if related web documents are linked together. Their system, named Conceptual Open Hypermedia System (COHSE), is a combination of ontological services and Open Hypermedia Link Service and enables the content to be annotated via domain ontologies [14]. For the demonstration, they used the Gene Ontology [1].

A number of components have been proposed in the well formed Text Engineering Platform (GATE) [15]. One of them is (OAT) which is a manual annotation system. Another component of it is the Onto Root Gazetteer, which creates the dynamic gazetteer based on the loaded ontology and performs the annotation automatically.

The currently developed models deal with semantic annotation, however, none of them deal with the consistency issues, i.e. if there is a change in the content, it must be reflected in the respective domain ontology and vice versa. Consistency has to be established during content change management. The research in the area of mutual dependencies between the content and the domain ontologies are still at an explorative stage.

V. EVALUATION

We followed the categorisation approach given by the quality standard model ISO/IEC 9126 and identified *functional suitability* as evaluation criteria. The evaluation criteria are compliant with definitions provided in the quality model.

Functional suitability refers to the adequacy of the solution in terms of its coverage of user needs and correctness of implementation. In case of systems aiming at semantic support, functional suitability focuses on how accurately the trace model can be used for guided access to the content by querying the ontologies semantically, i.e. how correct the reflection of semantic links is.

The functional suitability of the trace model has been empirically evaluated. It has been observed that the solution is valid and suitable to handle the core issues of software evolution. Ontology-based software application model plays a semantic role in guided access to the content. The consistency between the different software application artifacts and domain ontologies is preserved using consistency check model.

VI. CONCLUSION

When content management needs to work in multidimensional, multi-format, web-based applications, *semantic annotation* is a technique to support access to the content. To do so, an ontological layer has been proposed to be placed at the top of content management layer of application system.

The empirical study indicates that the solution is valid and suitable to handle the issues of content management systems. Currently we are focusing on the formalization of annotation trace model and the impact categorisation of the software application changes on the consistency of the domain ontology. The implementation of the approach as a trace model which includes tools and techniques and graph-based formalisation of annotation traces is our future work.

REFERENCES

[1] Gene Ontology Tool for the unification of biology: http://www.geneontology.org/.

[2] Boyce, S., Pahl, C.: The development of subject domain ontologies for educational technology systems. Journal of Educational Technology and Society (ETS) IEEE **10**(3) (2007) 275–288

[3] Bandara, K.Y., Wang, M., Pahl, C.: Context modeling and constraints binding in web service business processes. In Proceedings of the First international Workshop on Context-Aware Software Technology and Applications (2009)

[4] Holohan, E., McMullen, D., Melia, M., Pahl, C.: Adaptive Courseware Generation based on Semantic Web Technologies. In: Proceeding of the International Workshop on Applications of Semantic Web Technologies for E-Learning (SW-EL2005) at the Twelveth International Conference on Artificial Intelligence in Education (AIED2005) , IOS Press (2005)

[5] Kiryakov, A., Popov, B., Terziev, I., Manov, D., Ognyanoff, D.: Semantic annotation, indexing, and retrieval. Volume 2. (2004) 49–79

[6] Noy, N.F., Klein, M.: Ontology evolution: Not the same as schema evolution. Knowledge and Information Systems. **6**(4) (2004) 328–440

[7] Qin, L., Atluri, V.: Evaluating the validity of data instances against ontology evolution over the semantic web. Information and Software Technology. **51**(1) (2009) 83–97

[8] Gruhn, V., Pahl, C., Wever, M.: Data model evolution as basis of business process management. In: OOER '95: Proceedings of the 14th International Conference on Object-Oriented and Entity-Relationship Modelling, London, UK, Springer-Verlag (1995) 270–281

[9] Javed, M., Abgaz, Y., Pahl, C.: A pattern-based framework of change operators for ontology evolution. In: 4th International Workshop on Ontology Content. Volume 5872 of LNCS., Springer (2009) 544–553

[10] Stojanovic, L.: Methods and tools for ontology evolution. PhD thesis, University of Karlsruhe (2004)

[11] Ehrig, H., Prange, U., Taentzer, G.: Fundamental theory for typed attributed graph transformation. In Proc. of 2nd Int. Conference on Graph Transformation (ICGT) (2004) 161–177

[12] Petridis, K., Bloehdorn, S., Saathoff, C., Simou, N., Dasiopoulou, S., Tzouvaras, V., Handschuh, S., Avrithis, Y., Kompatsiaris, I., Staab, S.: Knowledge representation and semantic annotation of multimedia content. In: proceedings on V.I.S. Processing, Special issue on Knowledge-Based Digital Media Processing. Volume 153. (2006) 255–262

[13] Bechhofer, S., Yesilada, Y., Horan, B., Goble, C.: Knowledge-driven hyperlinks: Linking in the wild. Volume 4018 of Lecture Notes in Computer Science. (2006) 1–10

[14] Carr, L., Bechhofer, S., Goble, C., Hall, W.: Conceptual linking: Ontology-based open hypermedia. In WWW10, Tenth World Wide Web Conference. (2001)

[15] General Architecture for Text Engineering: http://gate.ac.uk.