DOCUMENT ENGINEERING : BUILDING SCALABLE DOCUMENT BY REUSING COMPONENTS

Win Pa Pa Htun
University of Computer Studies(Mandalay)
papawin009@googlemail.com

ABSTRACT

Building a new document needs the preparation of required information and finding the related written parts that have already identified from existing documents. In this paper, we outline the importance of structuring documents in order to facilitate the reuse of their content. We show how explicit structure representation facilitates the understanding of the original documents, identification of reusable existing parts and propose techniques to facilitate the configuration of the desired document. Our approach is based on Semantic Web Technologies, in particular, Variability Modelling reasoning with ontologies, and the idea of Active Documents extended with the adopted methods from Software Product Lines Engineering respectively.

Keywords: Active Documents, Software Product Lines Engineering, Semantic Web Techniques, Semantic Labels.

1.INTRODUCTION

Today, Building a new document needs the preparation of required information and finding the related written parts that have already identified from existing documents in order to save time and cost. Document Engineering provides us with procedures and algorithms how to specify, design, and implement sets of data into knowledge repositories which are called documents. Those documents contain different kinds of knowledge (such as marketing or advertisement, technical description or user tutorials), and they are represented in various formats (presentation, XML, Word documents) and very often not well-structured or formalized technical systems.

The use of XML allows for flexible elaboration of advanced algorithms with detailed document analysis, modification, partial reuse and adaptation. Further, the formats can be extended to serve special needs and are thus very flexible. The approach of Active Documents, raised in [3], assumes that a document contains both data and software (e.g., scripts, macros). As a result, documents can be manipulated interactively, for instance, they react immediately on user changes (hot updates), and can be derived automatically from a set of reusable components. It needs to indentify the proper methods for knowledge modelling, acquisition and automated techniques for data reuse that has hindered the development in the areas of document engineering and knowledge management. Knowledge management systems are designed to allow users to access and utilize the rich sources of data, information and knowledge stored in different forms, but also to support knowledge creation, knowledge transfer and continuous learning for the knowledge workers [8].

Our paper is organized as follows: first, we present the related works with this paper. Second, we introduce theory for the Document engineering and Reusing, and next, we represent the architecture style (abstract metamodel) for the document, which is called here Active Document. The architecture provides a fixed structure and separation of concerns, supporting reuse

procedures to map with variability modeling to represent the available active documents. The actual intention is to fill the gap between user requirements and an increasing number of available resources. We also introduce the light-weight formal modeling technique, called Semantic Labels, for each Active Document based on Semantic Web ontologies. A Semantic Label is a keyword on a meta-description of the documents, supported with certain ontology behind. Third, we present the system overview briefly. At the end, we summarize the proposed techniques, and the introduced architecture style, the modeling of available documents, and application of Semantic Labels.

2. RELATED WORKS

Sangpachatanaruk [3] described the design and implementation of architecture to support discovery, advertizing and fusion of multimedia information on the Internet. The basic tenet of the proposed architecture is a metadata abstraction, referred to as **adlet**. Based on this abstraction, a document dynamically built metadata, advertised itself to other active documents on the Internet, and eventually joined groups of documents of the same interest. The proposed architecture supported the creation and composition of adlets, and the negotiation and communications protocols required to manage adlet groups of common interests.

In [4] Uwe Assmann also presented an idea and proposed techniques to facilitate the generation of documents in collaborative team or distributed environment based upon the notion of Semantic Labels, which are realized as simple markup elements of the data and text, but represented in a logic-based manner. Thus, it provided platform-independent techniques and formal methods to describe a structure of the ultimate document, to compose reusable elements identified by semantic labels and fill out the semantically structured templates for the required document.

Karl Trygve Kalleberg [6] introduced a new approach for describing variable requirements for software product lines exploring the relationship between feature models and ontologies. First, examined how previous extensions to basic feature modeling move it closer to richer formalisms for specifying ontologies such as MOF and OWL. Then, explored the idea of feature models as views on ontologies. Based on that idea, proposed two approaches for the combined use of feature models and ontologies : view derivation and view integration. Finally, gave some ideas about tool support for these approaches.

Pohl and Metzger [7] provided a systematic approach for creating a diversity of similar products at low cost, in short time, and with high quality by explicitly modeling and managing variability, software product line engineering. This focused on the two principle differences of software product line engineering when compared to single systems development: the differentiation of two key development processes (domain engineering and application engineering) and the explicit representation and management of variability. And then, they characterized the two processes and their main activities and introduced the orthogonal variability modeling approach (OVM). Finally, illustrated the OVM approach in the product line requirements engineering and product line testing activities.

3. BACKGROUND THEORY

3.1. Document Engineering

Document Engineering helps us specify, design, and implement these documents and the processes that create and consume them. It synthesizes complementary ideas from information and systems analysis, electronic publishing, business process analysis, and business informatics to ensure that the documents and processes make sense to the people and applications that need them. A document-centric philosophy unifies these

different analysis and modeling perspectives. Using patterns for document exchanges and document components ensures we can build applications and services that are robust but adaptable when technology or business conditions change.

3.2. Reuse

Reuse is not a new concept, and the use of reuse to improve document quality and productivity has been investigated for some years now. [9]Reuse is defined as the "Use of work component without modification in the development of other document." Another kind of reuse is leveraged reuse, which is modifying existing work components to meet specific system requirements. The current build-from-scratch techniques of systems must give way to techniques that build systems by reusing building parts. These building parts can be the reuse of some design that solves some kind of problem. Other systems can use the same design to solve the same problem. By reusing

- Leads to higher quality of the work products
- Increases the productivity of the developers
- Does not necessarily shorten time-to-market (TTM), the chain of activities that determine the total project duration.

But there are two problems for this:

- Lack of pieces to build on or the problems connected to find these pieces.
- Chosen parts do not fit well together. This problem is called the Architectural Mismatch.

3.3. Architecture of the Document

Each document is comprised from a set of building blocks called Active Document Components (ADC). An ADC is a predefined piece of text or/and software, which is properly structured using XML and identified with several keywords (such as, "Introduction", "Lecture", "CS"), later called as Semantic Labels. Each ADC can be either atomic or composed from other building blocks. The actual composition task for ADCs is very dependent on the architecture styles and principles on which it relies. The architecture style for Active Documents separates the building block into four levels: context part (the actual text), syntactic XML structure (based on either OpenXML or ODF formats), formal model (see Semantic Label chapter below), and representation constraints (responsible for the representation of the text due to its XML structure and formal model). This architecture is expressed as an upper level XML tags extension of existing formats. As it is shown in Figure 1, the common platform for each document is a template. A template is the initial building block allowing actual embedding into it. The function of the template is to support reuse of the documents and their components, to simplify complex configuration tasks. Thus, each template is a parameterized Active Document Component (i.e. a component for a desired document).

3.4.Variability Modeling with Software Product Lines Engineering

Software product lines engineering is widely used for the efficient development of variable software products, based on a common platform. Two important kinds of activities are performed in software product line engineering:

1. The developers identify and describe where the applications of the product line vary in terms of the features.
2. The developers create reusable artifacts of a product line these artifacts are sufficiently adaptable (variable).

Variability modeling, which is used to document the variability of the product line, is a powerful tool for managing the complexity that is involved with the above kinds of activities. This is also applicable for documents: each feature addresses the possible potential requirement, assuming atomic or composed functionality behind. Numerous variability modeling approaches exist today to support domain and application engineering activities. Here, we follow an approach called Orthogonal variability modeling approach (OVM-A), in which the assets model and the variability model are kept separate. The variability model relates to different parts of the assets model using artifact dependencies. The central concepts used in OVM are variation points (VP)) and variants (V). A VP documents a variable item and a V its possible instances. Both VPs and Vs can be either optional or mandatory. Optional variants of the same VP are grouped together by an alternative choice. An optional variant may be part of at most one alternative group. To determines how many Vs may be chosen in an alternative choice, the cardinality notation [min..max] is used. OVM also supports the documentation of Vs belonging to different VPs. Simple constraints between nodes (mutex or require) can be graphically represented and can be applied to relations between Vs, but also to VP-V and VP-VP relations.

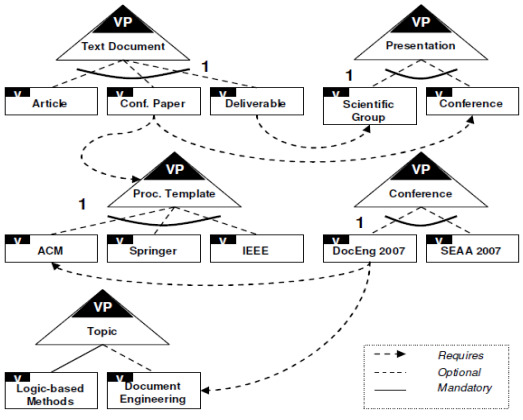

Figure 1. An Example of Variability Model

The configuration task is performed by collecting user requirements about the desired document and mapping them to features from the variability model. Mapping comprises expansion of requirements due to the information from the ontology and relating them with known variants and variation points. Then, by following the graph model, needed additional features are added or otherwise excluded from the particular configuration. After completing the design of the specific document model (architecture), the embedding is based on automated integration of existing documents. It is achieved because of using semantic interoperability between variants in the configuration of the document and Semantic Labels as models or proxies of existing ADCs.

3.5. Semantic Labeling

Obviously, current open standards for document specification will profit from the extension with formal models based on semantics. We propose to support current techniques of the open document idea instead of thinking about new "semantic related" models. Our approach is based on the notion of *Semantic Label*, which is a mark-up element, applied to the part of the document that is assumed to be reused in the future (ADC). An ADC could be a definition of a theorem, a value of a variable expected for later calculations, a figure, or a structural element, e.g. heading or special paragraph. A Semantic Label itself is a keyword, sentence or set of words (without any meaning), and it is identified during document preparation, or at the time of ultimate document planning. After creation, each label is registered on the server side in the repository, hence all users can see existing labels and their properties. The repository is organized as an ontology-based system, which means that a label becomes automatically a class or an instance of the class of the related ontology. The ontology is responsible for the definition of the label and formal semantics. Once the user provides

a definition for his environment or domain, the Semantic Label is becoming a part of it.

As an example, consider this paper, which can be described with keywords: “Software Product Lines”, “Semantic Web Techniques”; by authors responsible for creation of the document; date, and a related event: the conference on Document Engineering. Also, the information about the used template can be significant, but in our case ,this is already predefined on variability level as a requirement relation between variant “DocEng 2007” and template of proceedings – “ACM”. Later, when identifying relevant documents to be somehow reused and integrated into the prepared template, theontology plays a key role. It provides an inferencing base for the expansion of calling constructs: for instance, the call for paper prepared on topic of “Document Engineering” and written with the use of ACM template will result in the paper prepared for “DocEng conference” .

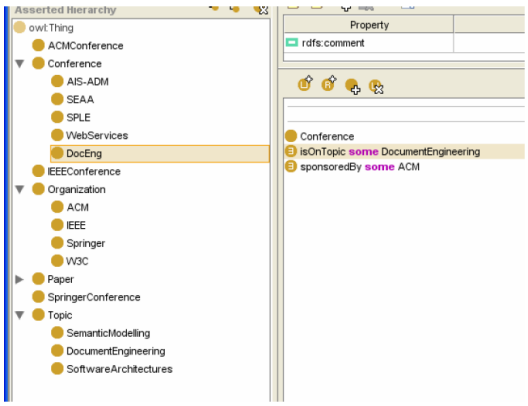

Figure 2. Ontology Example

3.6. Reuse Detection

Components that are reused in different documents would result in duplicate components in the repository. Reuse detection is used to avoid these duplicates an approach is the vector-space model, which treats a document as a bag-of-words. Similarity is commonly determined by the cosine similarity measure. The approach represents each document as a vector in an n-dimensional space. Document similarity to another document is then defined as the distance between the two vectors:

$$sim(R,Q) = \frac{\sum_{i=1}^{N} \alpha_i^2 * F_i(R) * F_i(Q)}{\sqrt{\sum_{i=1}^{N} \alpha_i^2 * F_i^2(R) * \sum_{i=1}^{N} \alpha_i^2 F_i^2(Q)}} \quad (1)$$

where α_i is the weight associated with the occurrence of the i^{th} word and $F(D)$ (size n) is the *frequency vector*. F_i (D) is the number of occurrences of word w_i in text fragment D. To illustrate the similarity computation, consider a registered text fragment R=”a b c” and new text fragmentsS_1 =”a b c” and S_2 =”c d e”. Using the cosine similarity measure for the example and assuming uniform weights for words (α = 1),

$$sim(R,S_1) = 1*1+1*1+1*1 \Big/ \sqrt{3*3} = 1 \quad (2)$$

$$sim(R,S_2) = 0*1+0*1+1*1 \Big/ \sqrt{3*3} = 0{,}3 \quad (3)$$

Identical text fragments have a similarity value 1, while text fragments that do not have much overlap have a low value (e.g. 0,3 in the example). The approach has been extended to incorporate term frequencies, document, and term frequency/inverse document frequency weighting. Such extensions improve the effectiveness of similarity comparisons[15].

4.SYSTEM OVERVIEW

Our system starts with some an inspiration from already read documents, acquiring some parts of a context to be revised, adopted and adapted to the particular ideas, and further filled into the template. Usually, the document being created belongs to the same domain (e.g., particular topic of research, or account of a manager) and needs mechanisms for keeping information about its

content, available topics, purposes for creation of new documents, etc.

From the knowledge about a certain domain, we obtain a template and structure, manually write a plan for the context, and keep in mind what we can use and reference. Thus, we are making several contributions for facilitating the initial stage of document creation and planning. And then acquiring parts are labeled by semantic storing in repository when creation of new documents, adjust with variability modeling. But, storing parts are not to be duplicate, is calculated by the vector-space model.

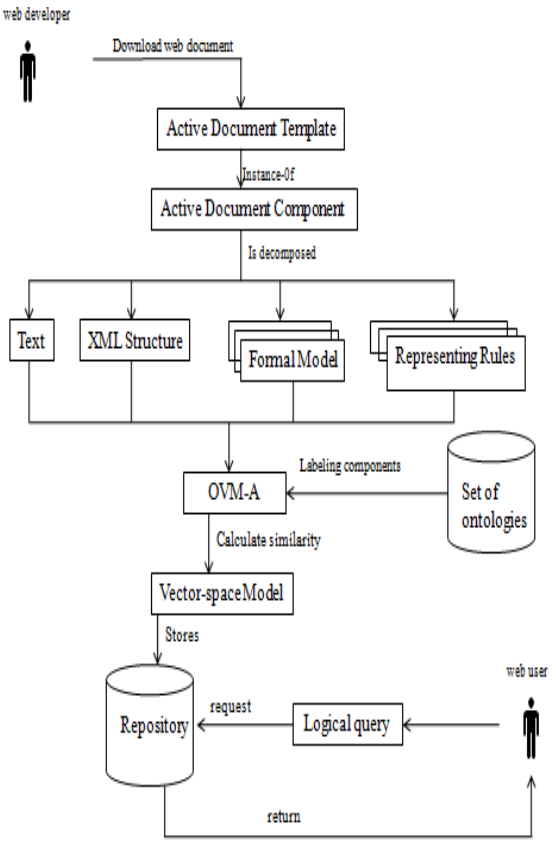

Figure 3. Overview of System

As we can see, semantic label together with the whole supporting framework can be a well-defined model of the component. But an application of semantic labels on components involves a large variety of information from different application domains and of various categories, like terms and definitions, behavior rules, probability relations, and temporal properties during the knowledge generation can be processed with various inference machines, and in particular with Prolog. Thus, it seems to be the obvious to choose the most expressive logical formalism that is capable to formulate and formalize the entire needed information. But, doing so very likely results in severe decidability problems and exponentially growing up computation time.

5. CONCLUSION

Addressing today's challenges in document engineering to already known methods of software engineering can solve a lot of problems, and improve already existing methods and techniques. By using properly suited modeling, easy search and identification of documents is possible without retrieving their context, which turns to be very critical, time-consuming and costly to build.

The proposed approach needs a proper framework, where the central part is based on the use of different heterogeneous models to derive a needed solution. The core part of the architecture is built on Prolog processing user requirements, a formal variability model, Semantic Labels and ontologies as domain description, interoperating on various models. As a result, configuration, identification and composition tasks are performed as classical inferencing problems. The later use of the Java Prolog Library [16] allows the integration of this core functionality into any Java-environment and provides flexibility in building new document processing tools.

REFERENCES

[1] OpenDocument Format (ODF),

http://en.wikipedia.org/wiki/OpenDocument.
[2] Open XML, http://en.wikipedia.org/wiki/Open_XML.
[3] Chatree Sangpachatanaruk,Taieb F. Znati, and S. K. Chang, "An Architecture for a Personalized Web of Active Documents", Department of Computer Science, Department of Information Science and Telecommunication, University of Pittsburgh, Pittsburgh, PA 15260

[4] Mikhail Roshchin, and Uwe Assmann, "Semantic Labeling for Active Documents",TU Dresden Germany

[5] Glushko R.J., McGrath T. "*Document Engineering: Analyzing and Designing Documents for Business Informatics and WebServices*", MIT Press, November 2005.

[6] W. Cohen and L. Jensen, "A structured wrapper induction system for extracting information from semi-structured documents", *17th International Joint Conference on Artificial Intelligence, Workshop on Adaptive Text Extraction and Mining*, Seattle, USA, 2001.

[7] M. Henzinger and S. Lawrence, "Extracting knowledge from the World Wide Web", *in Proceedings of the National Academy of Science*, USA, 101: 5186-5191, 2004.

[8] Razmerita L., Angehrn A., Maedche A. "Ontology-based User Modeling for Knowledge Management Systems."

[9] W. C. Lim: "Effects of Reuse on Quality, Productivity and Economics", in: IEEE Software, 9, 1994

[10] D. Garlan, R. Allen and J. Ockerbloom: "Architectural Mismatch: Why Reuse Is So Hard", in: IEEE Software, 11, 1995

[11] G. Carenini, R. T. Ng, and E. Zwart, "Extracting knowledge from evaluative text", *in Proceedings of the 3rd International Conference on Knowledge Capture*, Banff, Canada, 2005, pp. 11-18.

[12] W. R. Cyre, "Knowledge Extractor: A Tool for Extracting Knowledge from Text", *in Proceedings of Fifth International Conference on Conceptual Structures (ICCS)*, Seattle, USA, 1997, pp. 607-610.

[13] M. Vargas-Vera, E. Motta, J. Domingue, S. Buckingham Shum, and M. Lanzoni, "Knowledge Extraction by using an Ontology-based Annotation Tool", *In Proceedings of the Knowledge Markup and Semantic Annotation Workshop*, Victoria, Canada, 2001, pp. 5-12.

[14] Krzysztof Czarnecki, Paul Grünbacher, Andrzej Wąsowski, Rick Rabiser, Klaus Schmid ,"Cool Features and Tough Decisions:A Comparison of Variability Modeling Approaches"

[15] Katrien Verbert, Xavier Ochoa, and Erik Duval, "The ALOCOM Framework: Towards Scalable Content Reuse"

[16] Java-Prolog Library, http://www.swiprolog.org/packages/jpl/.