A Platform to Automatically Generate and Incorporate Documents into an Ontology-Based Content Repository

Matthias Heinrich
SAP AG, SAP Research
Dresden, Germany
matthias.heinrich@sap.com

Antje Boehm-Peters
SAP AG, SAP Research
Dresden, Germany
antje.boehm-peters@sap.com

Martin Knechtel
SAP AG, SAP Research
Dresden, Germany
martin.knechtel@sap.com

ABSTRACT

In order to access large information pools efficiently data has to be structured and categorized. Recently, applying ontologies to formalize information has become an established approach. In particular, ontology-based search and navigation are promising solutions which are capable to significantly improve state of the art systems (e.g. full-text search engines). However, the ontology roll-out and maintenance are costly tasks. Therefore, we propose a documentation generation platform that automatically derives content and incorporates generated content into an existing ontology. The demanding task of classifying content as concept instances, setting data type and object properties is accomplished by the documentation generation platform. Eventually, our approach results in a semantically enriched content base. Note that no manual effort is required to establish links between content objects and the ontology.

Categories and Subject Descriptors

D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement—*Documentation*; I.7.2 [**Document and Text Processing**]: Document Preparation

General Terms

Documentation

Keywords

Software Documentation, Text Generation, Semantic Annotation, Ontology Completion

Notice

1. INTRODUCTION

The increasing digital knowledge base created by the information society asks for potent content search and query methods. Traditionally, search engines solely rely on algorithms indexing a given content base. Besides analyzing text syntactically advanced techniques are capable to exploit metadata. This approach enables end-users to navigate, filter and search various information sources more efficiently. In order to use metadata-based access strategies, existing content repositories have to be annotated. Enriching information pools with metadata or so called semantic annotations is a major effort. In particular large information repositories need automatic or semi-automatic processing in order to constrain costs.

Hence, we propose a documentation generation platform that allows for content creation and automatic content annotation. The software documentation domain is utilized to validate the platform. All generated content is derived from graphical software models (e.g. BPMN, UML). The preprocessed content (bullet point lists, screenshots, etc.) empowers technical authors to efficiently write requested documentation (e.g. end-user manual, technical specification). As stated before, the content generation is accompanied by metadata creation. That includes the instantiation of individuals, object properties and data type properties which are all specified with the Web Ontology Language (OWL) [7]. The ontology captures the domain knowledge. Therefore, the typical workflow of technical authors can be supported by various features (e.g. display relevant context information, semantic drill-down, semantic search) leveraging generated semantic annotations. Eventually, the proposed documentation generation platform enables semantic enrichment at lean costs.

In this paper, we proceed with a brief discussion of related work in section 2. Section 3 describes the distinct platform components and section 4 outlines the required steps to apply the documentation generation platform. In section 5, we summarize the benefits and propose several enhancements with respect to the introduced platform.

2. RELATED WORK

Documentation generation can be initiated in two different software development steps, either during or after the implementation phase.

The method to extract documentation from compiled software is called reverse engineering. This extraction method focuses on technical artifacts, such as API definitions or system architecture information. Reverse engineering systems are described in [14] and [12].

There are several means to integrate documentation tasks within the implementation phase. One way is to annotate source code. Literate Programming [9] and Elucidative Programming [11] are examples of this approach. Literate Programming is a method to combine source code and documentation in a human-readable fashion. High level descriptions such as process definitions are not covered. Elucidative Programming is a variation of literate programming that establishes links between source code and documentation. A tool supporting Elucidative Programming is the Development Environment for Tutorials [3]. It allows writing and maintaining tutorials while integrating source code snippets. If the documented source code changes, the examples in the tutorial change as well.

Another technique to generate documentation is coupled with the software generation process. Software Documentation Support [8] fosters information generation at the specification and development phase.

Ontologies are used to interlink software artifacts and documentation. In [6] their goal is to improve system maintenance. Other approaches describe a method to populate ontologies from a set of documents. For example Witte et al. describe in [16] an ontological formalism to integrate source code and software documentation. The method incorporates two specific ontologies, a source code and a documentation ontology. Both ontologies are automatically populated using source code analysis and text mining respectively. The ontologies support tasks to establish traceability links between source code and documentation. In this approach, information is extracted from existing documentation whereas we focus on documentation generation with automatic annotation.

3. PLATFORM

The documentation generation platform implements a two-step process: generate content and classify the derived content. The platform's high-level architecture is illustrated in figure 1 and consists of 3 major building blocks. At first, software engineers have to specify software models using the developer's workbench. After completing the software model specification, the extraction component runs and filters relevant information for documentation purposes. That results in saving derived data in the content and ontology store. Finally, technical authors access the content and ontology store through a dedicated authoring environment and transform the preprocessed content into final documentation.

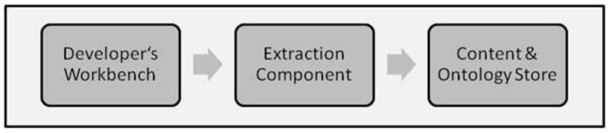


Figure 1: Overview of the Documentation Generation Platform

Figure 2 shows the detailed platform architecture. In order to automatically process software models they have to be stored in a structured manner. The model structure is manifested in a so called metamodel. A metamodel formalizes how a valid model (or instance of a metamodel) has to be constructed. The developer's workbench offers a dedicated set of editors which is based on metamodels and therefore enforces metamodel-compliance. The current prototype uses the Eclipse Modeling Framework (EMF) [13] in order to facilitate metamodel definition and model management (e.g. load, save, access models).

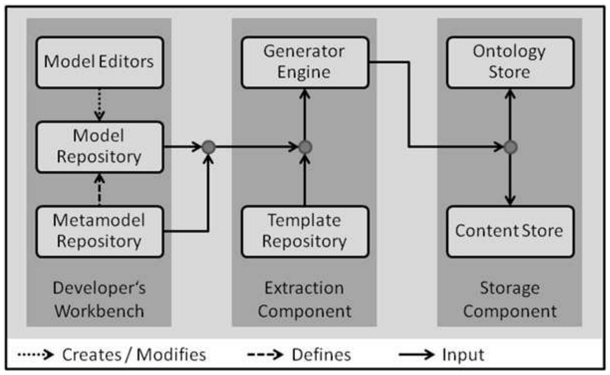


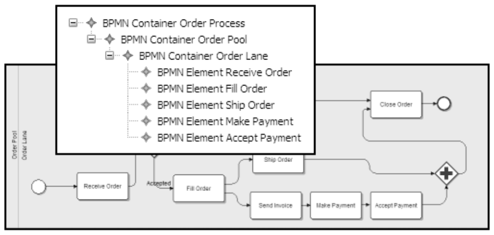
Figure 2: Detailed Architecture of the Documentation Generation Platform

In order to filter information represented in software models, the extraction component evaluates models. Rules defining whether information blocks are valuable for the documentation process are captured in text generation templates. Since templates are also metamodel-based, they can traverse the underlying metamodel structure. Thus, the extraction component – unlike simple transformation engines – is able to access associated model elements. Hence, the entire model-context can be interpreted in order to determine relevant content. Besides generation, templates also define links between information objects and ontological entities. Generated content for instance might be a member of a class, a data type or an object property. Possible values are defined in the linked ontology. The content creation is accomplished by a generator engine. The engine reads in models and metamodels in order to process the text generation templates. The implementation is realized using the openArchitectureWare (oAW) framework [5]. The oAW framework supports EMF models and provides the text generation language Xpand [2] as well as an Xpand editor. The generator engine is distributed as a set of Eclipse plug-ins.

The storage component saves and manages generated content delivered by the extraction component. It is divided in a content store and an ontology store. The saved information is exhibited through an authoring environment where technical authors can refine raw, semantically enriched information. Once the documentation is completed, a transformation to an established publishing format concludes the documentation process. The prototypical implementation leverages the Semantic MediaWiki (SMW) [10] that is designed to hold content interwoven with semantic annotations.

4. EXAMPLE

This minimal example demonstrates the application of the documentation generation platform. The objective of the example is to support technical authors. They are supposed to document business processes. Therefore, they have to transform existing process models into documentation. While writing business process documentation, a convenient feature is a process drill-down view that permits quick access to all information described by the process. Figure 3 displays a process drill-down view and its linked process model.



BPMN Container Order Process
BPMN Container Order Pool
BPMN Container Order Lane
BPMN Element Receive Order
BPMN Element Fill Order
BPMN Element Ship Order
BPMN Element Make Payment
BPMN Element Accept Payment
Order Pool
Order Lane
Receive Order
Accepted
Fill Order
Ship Order
Send Invoice
Make Payment
Accept Payment
Close Order

Figure 3: Drill-Down View and the underlying Business Process

The shown drill-down view can be generated from the linked business process. This example lists crucial steps to derive the drill-down view's data by applying the documentation generation platform.

Initially, the documentation generation platform requires some configuration effort. This includes the following steps:

1. Create an ontology taking into account relevant domain concepts and their relations.

2. Write text generation templates with respect to the supported metamodels.

In the example, the ontology has to capture parent-child relationships. These relationships are exploited in order to construct the drill-down view. Such relationships are for instance

- business process diagrams consist of multiple pools or
- process lanes hold a set of tasks.

The defined OWL ontology is depicted in figure 4.

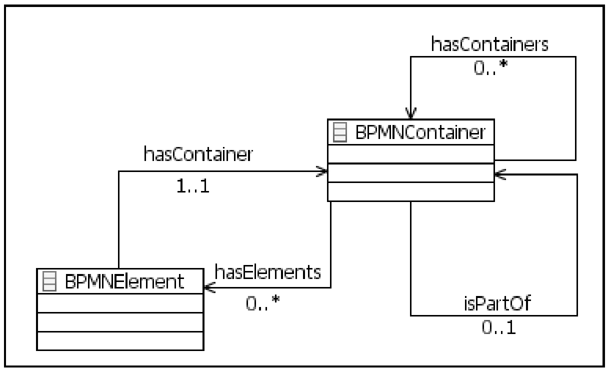


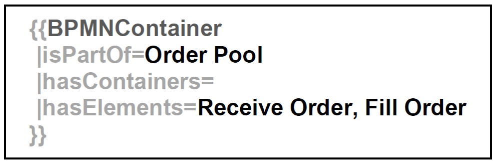
Figure 4: Visualization of the Domain Ontology

The ontology comprises OWL classes (BPMNElement, BPMNContainer) and OWL object properties (hasContainers, isPartOf, etc). Since the storage component is realized using the SMW, the OWL ontology has to be transformed to the SMW vocabulary according to table 1.

Table 1: OWL to SMW Mapping [15]

Web Ontology Language	Semantic MediaWiki
Class	Category
Object Property	Relation
Data Type Property	Attribute

After mapping classes and object properties to categories and relations, BPMNContainer and BPMNElement can be instantiated in the SMW. Figure 5 illustrates the syntax to create a SMW page that is an instance of category BPMNContainer. Besides creating instances, relations to other SMW pages are established using the "=" operator.



```
{{BPMNContainer
 |isPartOf=Order Pool
 |hasContainers=
 |hasElements=Receive Order, Fill Order
}}
```

Figure 5: Example Instance of Category BPMNContainer (using SMW-Template Syntax)

Until now, the storage component is capable of managing ontological entities that adhere to the ontology in figure 4. The next step is the specification of text generation templates. The templates generate BPMNContainer and BPMNElement instances while respecting the SMW syntax. Typically, templates define static and dynamic aspects. In figure 5 everything except the object property values (Order Pool, Receive Order, Fill Order) are static parts. In the Xpand code generation template in figure 6 one can easily recognize the static parts since they are defined in plain text. Dynamic parts are marked by guillemets (<<, >>). They evaluate data provided by the business process model. For instance, the << *pool.name* >> expression in figure 6 calculates the name of the pool the lane belongs to. The syntax to navigate the model (e.g. << *pool.name* >>) is defined by the BPMN metamodel[1] which is imported into the Xpand template. Besides dynamically determining the pool name, the hasElements object property enumerates all tasks included in the current lane. Consequently, the template in figure 6 specifies a rule expandActivity which prints all task names separated by comma.

After completing the Xpand template description, the generator engine executes templates and submits generated text to the SMW system. To programmatically initiate text generation the oAW framework accommodates a workflow mechanism [2]. Data submission is facilitated by a dedicated MediaWiki API [1]. As soon as all components are properly setup, business processes can be automatically parsed and content will be pushed into the SMW. Finally, all semantic relations required by the drill-down view are established.

The example demonstrates the ease of automatically deriving and storing data in an ontology-based content repository. Thereby, the generation process is fully supported by the documentation generation platform.

5. CONCLUSION AND FUTURE WORK

[1]The example utilizes the metamodel specified by the Eclipse BPMN Modeler [4].

```
«IMPORT bpmn»
…
«DEFINE expandLane FOR Lane»
        {{BPMNContainer
         |isPartOf=«pool.name»
         |hasContainers=
         |hasElements=
        «EXPAND expandActivity FOREACH activities»
        }}
«ENDDEFINE»

«DEFINE expandActivity FOR Activity»
        «IF activityType.toString().matches("Task")»
                «name»,
        «ENDIF»
«ENDDEFINE»
```

Figure 6: Xpand Template to Generate BPMNContainer Instances

In this paper, we demonstrated a documentation generation platform that is capable of extracting raw information and linked semantics from software models. Applying the documentation generation platform leads to a semantically enriched content base. Machine-readable semantics are the key to elevate search engines to the next level and produce more precise result sets. Enhancing content repositories with semantic annotations goes beyond advancing search engines. It offers new ways to navigate information pools using typed hyperlinks. Furthermore, the usage of the W3C standard OWL fosters the integration of heterogeneous information sources. An OWL-based ontology establishes a common vocabulary throughout various systems. The uniform vocabulary promotes modularization and evolvement which are essential assets to an open system landscape. To summarize, the presented documentation generation platform unlocks the benefits of an OWL-based content repository in a cost-efficient manner.

Currently, the documentation generation platform only masters text generation from software models. However, several context parameters also contain valuable information. For example a technical author might use contacts attached to generated artifacts to initiate author-developer communication. Therefore, the inclusion of context information within the generation process will be elaborated.

Another task is the design of a semantic-driven user interface. The SMW user interface is suited to collaboratively edit documents but lacks a lot of features provided by enterprise authoring environments. Thus, a user interface streamlining the authoring process and leveraging semantic annotations will be implemented.

Eventually, the documentation generation platform has to be evaluated. This covers a comparison of state of the art systems with the presented system. The evaluation will especially focus on process efficiency.

6. ACKNOWLEDGMENTS

The project was funded by means of the German Federal Ministry of Economy and Technology under the promotional reference ”01MQ07012”. The authors take the responsibility for the contents.

7. REFERENCES

[1] *MediaWiki API Documentation*, 2008. http://www.mediawiki.org/wiki/API.

[2] *openArchitectureWare User Guide*, 2008. http://www.openarchitectureware.org/pub/documentation/4.3.1/html/contents/.

[3] *DEFT - Development Environment For Tutorials*, 2009. http://sourceforge.net/projects/deftproject.

[4] *Eclipse BPMN Modeler*, 2009. http://www.eclipse.org/bpmn/.

[5] *openArchitectureWare Framework*, 2009. http://www.openarchitectureware.org/.

[6] A. P. Ambrosio, D. C. de Santos, F. N. de Lucena, and J. da Silva. Software engineering documentation: An ontology-based approach. In *Proceedings of the Webmedia and La-Web Joint Conference - 10th Brazilian Symposium on Multimedia and the Web 2nd Latin American Web Congress*, 2004.

[7] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. *OWL Web Ontology Language Reference*, 2004. http://www.w3.org/TR/owl-ref/.

[8] E. Horowitz and R. C.Williamson. Sodos: a software documentation environment-its use. *IEEE Transactions on Software Engineering*, 12(11):1076–1087, 1986.

[9] D. E. Knuth. Literate programming. *Computer Journal*, 27(2):97–111, 1984.

[10] M. Krötzsch, D. Vrandecic, and M. Völkel. Semantic mediawiki. In *Proceedings of the 5th International Semantic Web Conference*, 2006.

[11] K. Nørmark. *The Elucidative Programming Home Page*, 2009. http://www.cs.aau.dk/~normark/elucidative-programming.

[12] C. Riva and Y. Yang. Generation of architectural documentation using xml. In *Proceedings of the Ninth Working Conference on Reverse Engineering*, 2002.

[13] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks. *EMF: Eclipse Modeling Framework*. Addison-Wesley, 2 edition, 2009.

[14] A. van Deursen and T. Kuipers. Building documentation generators. In *Proceedings of the IEEE international Conference on Software Maintenance*, 1999.

[15] D. Vrandecic and M. Krötzsch. Reusing ontological background knowledge in semantic wikis. In *Proceedings of the First Workshop on Semantic Wikis: From Wiki to Semantics*, 2006.

[16] R. Witte, Y. Zhang, and J. Rilling. Empowering software maintainers with semantic web technologies. In *Proceedings of the 4th European Conference on the Semantic Web: Research and Applications*, 2007.