Creating Software Models with Semantic Annotation

Walter F. Tichy
Karlsruhe Institute of Technology
Am Fasanengarten 5
Karlsruhe, Germany
walter.tichy@kit.edu

Sven J. Körner
Karlsruhe Institute of Technology
Am Fasanengarten 5
Karlsruhe, Germany
sven.koerner@kit.edu

Mathias Landhäußer
Karlsruhe Institute of Technology
Am Fasanengarten 5
Karlsruhe, Germany
lama@ipd.uka.de

ABSTRACT

Requirements engineering is a big part of software engineering and consumes a lot of time. We propose a novel approach of automatically creating software domain models from textual requirements specifications using semantic annotation. Natural language processing (NLP) has progressed much in the last years and the usage of NLP tools for automatic annotation shows promising results. We use thematic roles [4] to explicitly denote the semantic relations in a sentence.

Semantic annotations also maintain the connection between textual artifacts and their corresponding model elements. Therefore changes in the domain model can be fed back to the textual specification. Additionally, changes in the textual specification can be analyzed and their impact towards the software model can be assessed.

Categories and Subject Descriptors

D.2.1 [**Software Engineering**]: Requirements/Specifications; D.2.10 [**Software Engineering**]: Design—*Methodologies, Representation*

General Terms

Design, Documentation, Experimentation, Theory

Keywords

Requirements Engineering, Automatic Annotation

1. INTRODUCTION

Today, semantic annotation is used especially in search related fields and knowledge management. A new but feasible approach is the usage of semantic annotation for software development. As part of software development, requirements engineering (RE) is time consuming and costly. Other work in this area include form based methods of specifying software such as semantic ontologies of software components, or the application of controlled languages such as ACE. We claim that requirements are and will stay mostly in "common" natural language. Therefore tool support is desirable [3]. Our research [7, 8, 5] shows that semantic annotation in combination with the latest achievements in natural language processing (NLP) can be used to support this process.

ESAIR'10, October 30, 2010, Toronto, Ontario, Canada.
ACM 978-1-4503-0372-9/10/10.

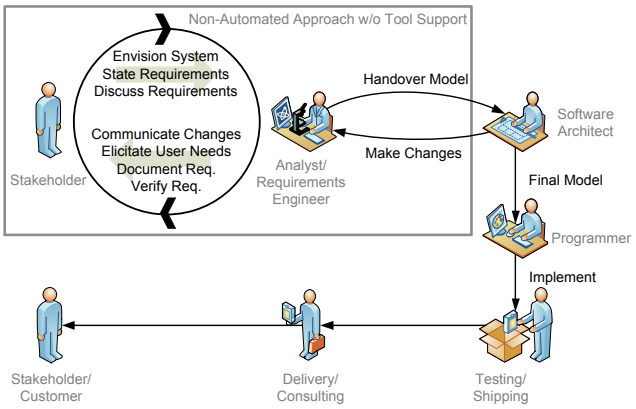

Figure 1: RE Lacks Tool Support

As shown in Fig. 1, RE lacks distinct tool support in the so called manual and psychological tasks. These tasks include the elicitation, verification, documentation, formalization, and transformation of stakeholder views into actual domain software models. The size of the gaps between the stakeholders' meaning, the analysts' understanding and the software architects' implementation vary a lot. The quality of requirements mostly depends on the skill set of the concerned groups. Using an automated approach to build software domain models improves the process in speed and precision and also gives the process a deterministic approach.

We consider semantics the main issue in RE where analysts have to grasp the envisioned system of the stakeholders as well as being able to communicate their findings to software architects and vice versa. Software architects make decisions and changes on software models which then affect the original requirements of the stakeholders. Synchronization between the corresponding groups takes place constantly, but not comprehensively. Also, the effort of synchronization sometimes excels the effort of the actual implementation. This is one of the main reasons why software requirements, implementation, and documentation underly an erosion process. Considering the fact that requirements specifications often found the base for legally binding contracts, we postulate the necessity to maintain the connection between Software Lifecycle Objects (SLOs) [2].

Taking a look into the future of software and market expansion, one has to prepare more efficient ways of software engineering. Even companies which do not understand soft-

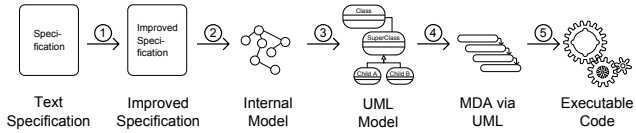

Figure 2: Automatic Model Creation from Text

ware development as their main competence, need tools to steer and support their development.

2. SEMANTIC ANNOTATION

We use an extension of Fillmore's thematic roles [4] to annotate textual requirements specifications. This set of 67 roles allows the explicit decoration of all semantic entities within a given text [5]. An example annotation is the following listing where the thematic role `AG` (agens) depicts the acting entity, `PAT` (patiens) the entity being acted upon, and `ACT` (actus) the action, that is being performed.

[**A** **shoemaker** | AG **repairs** | ACT **shoes** | PAT] .

As can be seen in Fig. 2, the specification is at first improved semi-automatically (1), including cleaning of text, disambiguation, and further tasks [6]. The automatic semantic annotations are then being used in the model extraction process. Using the annotations, we transform the requirements specification into an internal data model (2) [8]. This model is then automatically transformed into a UML model representation (3) which can be transformed into XMI output format (4) [5] for later use in the model driven architecture (MDA) as shown in (5).

We have run case studies to verify the results of our approach which include the WHOIS protocol specification, the FIDE laws of chess [1], and specifications used in text books and various papers. The automatic annotations have been compared to gold standards of the corresponding specifications. The results yielded an coverage of 83% with an incorrect annotation rate of only 5%.

3. NEW APPLICATION FIELDS

Using the latest NLP techniques to individually annotate semantics and applying transformation rules is the main step to automatically creating models from text. Combining these strategies with the "semantic power" of knowledge bases (e.g. WordNet or Cyc) tremendously improves computer generated results and leaves little manual work for the analyst.

The huge benefit of having semantic annotations in natural language textual requirements is that SLOs stay connected over time and all process steps. According [2], a SLO is any kind of artifact (see Fig. 3) involved in the software development process. SLOs are vertically dependent, which means that for instance a group of requirements is dependent from another group of requirements. Horizontal dependencies describe the connection of different types of SLOs. These dependencies are 1:n relations.

Establishing vertical dependencies between SLOs manually is tedious. For example, this can be done on the requirements end, establishing vertical links using tools like IBM's Rational Rose or HP's QualityCenter. Still, it is manual work and the analyst needs to gain complete overview

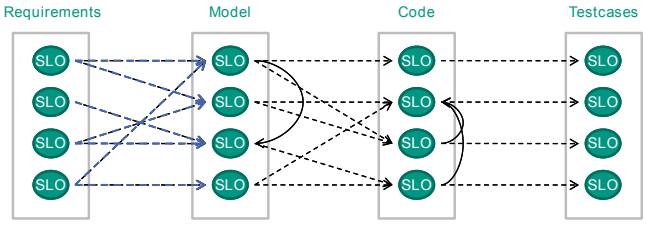

Figure 3: Keeping SLOs in Sync

of the process to generate the right dependencies. Maintaining horizontal dependencies is even more complex due to the lack of tool support as well as the gaps that exist between the various steps of the development process. Creating horizontal connections automatically when generating the models, supports the development process.

4. CONCLUSIONS

Creating software models from natural language texts is feasible using semantic annotations. Creating these annotations automatically is possible. This approach could speed up software development while decreasing the amount of possible errors that occur during the process. Also, establishing automatic semantic annotation is the first step of keeping SLOs in sync. We already create models from text but plan to refine our concept to a stage where changes in all SLOs can be reflected into their dependent counterparts. We are currently researching the automatic feedback of model element changes into text as well as the impact analysis of textual requirements changes and additions to already existing software models.

In the future we plan the further usage of our semantic roles concepts to create (non-trivial) test cases for software directly from textual or API specifications.

5. REFERENCES

[1] FIDE Handbook – E.I.01A. Laws of Chess, Feb. 2008.

[2] S. A. Bohner and R. S. Arnold. An introduction to software change impact analysis. In S. A. Bohner and R. S. Arnold, editors, *Software Change Impact Analysis*, pages 1–26. IEEE Computer Soc. Press, 1996.

[3] B. H. C. Cheng and J. M. Atlee. Research directions in requirements engineering. In *Proc. Future of Software Engineering FOSE '07*, pages 285–303, 23–25 May 2007.

[4] C. J. Fillmore. Toward a modern theory of case. In D. A. Reibel and S. A. Schane, editors, *Modern Studies in English*, pages 361–375. Prentice Hall, 1969.

[5] T. Gelhausen and W. F. Tichy. Thematic Role Based Generation of UML Models from Real World Requirements. In *Proc. International Conference on Semantic Computing ICSC 2007*, pages 282–289, 2007.

[6] S. J. Körner and T. Brumm. Natural language specification improvement with ontologies. *International Journal of Semantic Computing (IJSC)*, 03(04):445–470, 2010.

[7] S. J. Körner and T. Gelhausen. Improving Automatic Model Creation using Ontologies. In Knowledge Systems Institute, editor, *Proceedings of the Twentieth International Conference on Software Engineering & Knowledge Engineering*, pages 691–696, July 2008.

[8] S. J. Körner and M. Landhäußer. Semantic enriching of natural language texts with automatic thematic role annotation. NLDB 2010, June 2010.