Improving Natural Language Specifications with Ontologies

Sven J. Körner
Institute for Programming and Data Structures
University of Karlsruhe
76128 Karlsruhe, Germany
Email: koerner@ipd.uka.de

Torben Brumm
Institute for Programming and Data Structures
University of Karlsruhe
76128 Karlsruhe, Germany
Email: brumm@ipd.uka.de

***Abstract*—User requirements are usually written in textual form. Dealing with natural language textual specifications is complex. Analysts [1] have to cover all aspects of the requirements engineering process when working with the customer, such as work flows, psychological issues, and linguistic problems. We show how analysts can be supported during requirements elicitation and documentation. We present an approach to improve natural language requirements specifications using ontologies. We demonstrate by several examples from real requirements how an ontological reasoner called RESI [2] can uncover gaps, inaccuracies, ambiguities and ask the user to clarify them. In many cases, it supports the analyst by giving a small number of reasonable suggestions to choose from. The implementation of RESI is currently in progress.**

I. Introduction

Natural language helps to align the user's understanding of the requirements with the analyst's own viewpoint. What the requirements originator really meant and the "facts" that can be derived from the statements that were given are not always congruent [1]. Incomplete and faulty requirements emerge not only during the elicitation of requirements, but also during requirements analysis and modeling [2]. Requirement specifications often need to be revised and rewritten during one of the many iterations in the software development process. Focusing on the improvement of requirements in the first stages of the software development process saves time and improves the final results of a software [3]. So far, this is an inter-disciplinary and human centered process. Improving and accelerating the manual processes with machine support is desirable.

This paper represents the concept of a machine supported tool which helps the analyst to improve natural language specification during the authoring process. We present as proof of concept that the automatic creation of UML models from natural language specifications can be enhanced with "common sense" from an ontology. Since natural language will stay the predominant kind of requirements notation for some time [2], [4], Natural Language Processing (NLP) is a valid approach for the requirements engineering process. The deficits of NLP are the understanding and processing of semantics and their correlating side effects. Therefore, including ontologies for semantic knowledge integration into the NLP process is an obvious improvement. Every ontology contains a structure of world knowledge (or domain knowledge) which can solve some of the problems in the requirements engineering process as listed in Section V.

We are currently developing a generic solution called RESI to query various ontologies such as WordNet [5], ResearchCyc [6], and ConceptNet [7] in the correct context during NLP. This approach supports the analyst while examining a large number of requirements.

Preliminary results show that ontologies can improve the authoring and elicitation process of natural language requirements specifications [8], [9]. As requirements specifications are mostly huge document bundles which sometimes take days, weeks, or months to examine, a system that supports the analyst during the requirements engineering process could be a real problem solver. This paper describes a tool which can improve natural language requirements during the automatic modeling process.

The following Section II covers the related work. Some problems that occur and need to be solved during the manual requirements elicitation process are explained in Section III via examples. Section IV shows how RESI works and demonstrates how certain issues can be solved. An abstract discussion about well-known problems in requirements engineering is led in Section V. We point out how we plan to answer these problems with RESI. A case study follows in Section VI and the paper ends with the conclusion in Section VII.

II. Related Work

Requirements engineering is concerned with the elicitation of high level goals. These goals are to be achieved by the envisioned system [10]. Requirements engineering includes the refinement of such goals and their operationalization into specifications of services and constraints. Today, several approaches use domain specific ontologies to cope with the problems that occur in the requirements engineering process [11], [12], [13], [14], [15]. Some of these projects research the application of formal specifications. They use ontology based systems to correctly classify textual information that has

[1] A requirements analyst is a person who knows how to ask questions for different stakeholders, has experience in working with customers and the management of existing requirements, etc. The characteristics a perfect analyst should have are explained in [1].

[2] Requirements Engineering Specification Improver

been delivered from stakeholders. Other projects for example make sure that the correct (domain specific) wording is used when the specification documents are elicited from different stakeholders. Some projects have a narrow field of application and are specified for certain conditions and use cases [16], [7]. Until today, real world applications as listed in [17] have not yet adopted many of the research approaches.

In 2000, Nuseibeh and Easterbrook [4] drafted a road map which shows future research areas to cope with the problems in requirements engineering. They enumerate especially the development of new techniques and the bridging of the gap between contextual inquiry and formal representations.

In 2007, Cheng and Atlee [2] wrote a detailed summary about the requirements engineering's state of the art. They categorize the various topics and try to predict the future of requirements engineering research. Since requirements engineering consumes a lot of time during the software engineering process and has long-term effects to every following stage of the development process, focusing on its improvement is desirable.

Complex and time consuming manual tasks have not yet pervasive tool support [2]. Automating parts of this procedure could not only speed up the processing times of requirements but also decrease error rates.

Cheng and Atlee conclude that it is important to realize that requirements define the problem, not the software itself. They show that requirements engineering activities – in contrast to other software engineering activities – are more iterative, involve more players who have more varied backgrounds and expertise, require more extensive analyses of options, and call for more complicated verifications of more diverse components (e.g. software, hardware, human).

As a possible solution, formal representations of natural language formalize the semantics of statements. This is impossible without a loss of usability or information. Formal specification languages are often perceived as difficult to use by practitioners [18]. Formal approaches need especially trained and skilled analysts to formalize natural language requirements [19]. Also after the formalization, it is hard for stakeholders to take part in further discussions about the requirements. They are not trained to think, act, and talk in formal representations like predicate logic or similar techniques [20]. So far, there is almost no evaluation of how well requirements engineering research reflects in industrial applications.

On the other hand, Fantechi et al. [21] explain that natural language is the perfect vehicle to represent use cases. No other language is as expressive as natural language. The use of natural language is also encouraging to end users who can easily follow and validate the use cases.

Robinson and Pawlowski [22] present empirical studies that show the difficulties and communication breakdowns that requirements engineering processes are frequently experiencing. Requirements inconsistency is a critical driver of the complete requirements engineering process and the requirements dialog.

Therefore, many projects focus on the disambiguation of natural language specifications [23], [24]. As Kiyavitskaya et al. mention, synonyms in specifications are mostly detected through the human analyst's domain knowledge [25].

III. Examples of Ontology Improvements

Domain and world knowledge help humans to process requirements, but understanding requirements is still complex. Problems occur not only when there is a lack of domain specific knowledge, but also when special knowledge and expertise overlap. Then stakeholders misunderstand each other without even noticing.

A. The Meaning of a Sentence

Consider the following sentence for example:

```
The gain should be doubled.
```

This sentence includes three major problems within five words.

1) Exceptional Case: The modal operation does not cover the exceptional case. The sentence does not describe what needs to be done if the gain is not doubled.

2) Homonyms and Polysemy: The word `gain` does have several meanings. Depending on the context and the stakeholders involved in the requirements elicitation process, it could mean gain as in "financial revenue" (this is how an MBA would interpret it) or gain as in "tube voltage" (as an electric engineer would interpret the sentence). This is called lexical ambiguity. Depending on background and education, the analyst might realize the knowledge gap and the fact that `gain` does mean something entirely different in this case. But what happens if not? It is important to distinguish the use of such words and realize their meaning.

3) Quantity: The sentence brings in the quantity `double`. It is not correctly quantified since we cannot decide whether it means *exactly* 2.0, that is a 100% increase. Maybe `double` also applies when next quarters revenue numbers reach anything from 180% to 220% of the comparison value.

B. Disambiguation with Semantics

A short example:

```
(1) Tom saw the   plane   flying.
(2) Tom saw the mountains flying.
```

Sentence (1) states the fact that the plane is flying while being seen from a spectator on the ground. Sentence (2) resembles exactly the same structure as its predecessor. The sentence bears several ambiguities which need to be resolved when working with requirements. Ontology knowledge can enhance this process by suggesting the user which aspects are more relevant and by reformatting the sentence. In the first sentence, the object `plane` is flying and the subject `Tom` seeing. In the 2nd sentence, the verb `flying` (most likely) references the subject `Tom` and not the object `mountains`. In this case, distinguishing the difference using parsers does not work. Instead, ontologies know that mountains do not fly, but planes do.

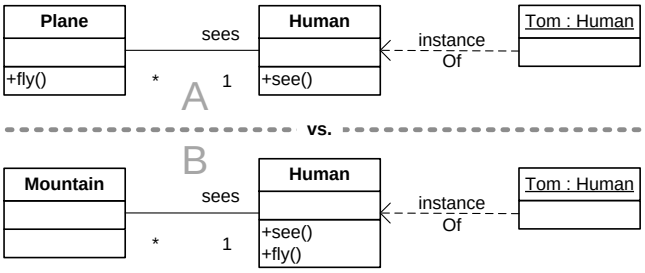

Fig. 1. Class Diagram

When modeling the above sentences in UML, the resulting class diagram of the first sentence looks like Figure 1A compared to the 2nd sentence in Figure 1B. Finding the meaning of words in their sentence structure is vital to software projects, especially when the analyst is not a domain expert. This is often the case, especially in times of offshoring.

C. Types of Ambiguity

As listed in Ceccato [26], there are many different types of ambiguities: Syntactic ambiguity occurs when grammatical structures lead to a different meaning of a sentence.

```
Porcelain egg container.
```

Does this describe an egg made out of porcelain, or is it the container that is made from porcelain?

Semantic ambiguity occurs when a sentence has more than one way of reading it. Take the the following sentences for example.

```
(1) Every man loves a woman.
(2) I saw a man on the hill with a telescope.
```

Does sentence (1) mean that all men love the very same woman or that each man loves "his" woman. And who carries the telescope in sentence (2)? Who is standing on the hill? There are many possible meanings (permutations) for each of the above sentences.

D. Coherence Checking

Other problems arise with references that stem from context knowledge. Checking coherence for example looks like this:

```
Tom is a man.
Larry is a cat.
He lifts him up and puts him in his basket.
```

The meaning of who `he` is becomes instantly clear to a human observer. This is due to the background information humans gather from their own world ontology. A query to an ontology such as ResearchCyc [6] would result in the information that human beings weigh anything between 10 to 250 times as much as regular domestic cats. Therefore it is not possible for the cat to lift the human, but vice versa. Of course the supporting system would have to ask the user to specify what kind of `cat` the text is referencing to. It makes a number of suggestions about the most likely case from what it already knows from the ontology. If the `cat` is a tiger, this might result in a different interpretation.

TABLE I
THEMATIC ROLES AND THEIR MEANING

Thematic Role	Explanation
AG	The acting person or thing executing the action.
ACT	The action, executed by person or thing.
PAT	Person or thing affected by the action or on which action is being performed.
HAB	Possession or belonging; person or thing being received or passed on by person or thing.
POSS	The (current) owner of an element. The "possessor".

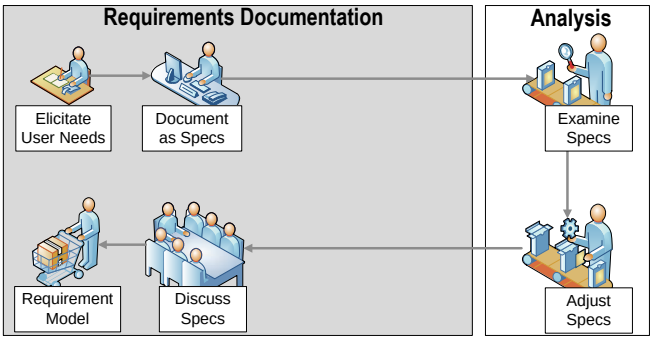

Fig. 2. Manual Requirements Engineering Process

IV. THE RESI PROCEDURE

As shown in related work, it is important to "formalize the process without formalizing it". This enables users to still understand the specification while machines can process the input. We use semantic annotation to enrich the specification with additional information [8], [9].

A. Formalizing Textual Specifications through Annotation

Semantic annotation is done by using thematic roles [9]. Thematic roles describe the role of each element [3] in a phrase. A short excerpt of the 70 thematic roles [27] we are working with can be found in Table I. This additional semantic information can also be used to query the ontology with its domain or world knowledge. The results support the analyst during the requirement creation process.

The manual process depicted in Figure 2 shows the stages of a requirements documentation process as it is today: First the user needs are elicitated. Then they are converted into requirements, documented as a specification, and later examined. The analysis most likely leads to adjustments in the requirements and their documentation. The process is iterative and after the requirements pass the quality gate, they go back to stakeholders where they need to be discussed and approved. These requirements form the basis of the requirement models which are passed on to other branches involved (e.g. development, quality assurance).

[3] In linguistics, this is called a constituent. A constituent is a word or a group of words that functions as a single unit within a hierarchical structure.

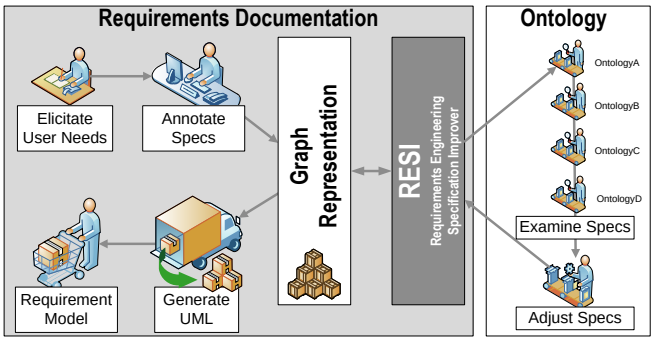

Fig. 3. Requirements Engineering supported by Ontologies

The new improved process that RESI uses is shown in Figure 3. It is semi-automated and uses the text as well as the semantic annotation of the specification. The semantic annotation leads to an altered specifications which is enriched with additional information. The specification is then compiled into a graph system which makes the specification machine readable. RESI has an interface to the graph system and is able to communicate bidirectionally.

B. Ontology as Magic Box

We are currently implementing RESI to run natural language queries on several ontologies simultaneously (see Figure 3). Depending on whether a domain specific knowledge base or a certain lexical ontology is necessary to retrieve the corresponding information, RESI will query the respective sources and take the corresponding action. For example, this could be a change in the sentence structure or a replacement of a certain term or subject. Depending on the type of the results, it is possible that the system cannot decide which action to take. User interaction via a graphical user interface is necessary. A collection of the type of queries that can be solved with RESI is listed in Section V.

C. Closing the Loop

The results RESI gathers from the different ontologies lead to changes in the graph. These improvements are fed back to the graph system (see Figure 3). The altered graph represents an improved specification which is more likely to support the development team in the software creation process since many standard obstacles of the requirements engineering process have been removed already. After the text has been formally represented in the graph, it is possible to create UML specific XMI files from this graph [28]. This ensures the direct connection between the natural language specifications and their model representations.

V. Challenges and Solutions

There are many challenges when supporting the human decision making with ontologies. According to Rupp [29], the most important problems with natural language specifications lie in:

A. Nominalization

Processes are sometimes hidden in a nominalization which needs to be explicitly specified for a correct requirement. E.g. the noun *notification* from the verb *notify*. This nominalization could be replaced by the explicit phrase *send message from A to B which includes information XYZ.*

B. Incomplete Process Words

Process words (such as verbs) need to be written in active voice. If passive voice is used, an actor (thematic role `AG`, see Table I) needs to be specified. Also all participants and circumstances involved in the process need to be stated. RESI checks the valency and ensures that all possible configurations are given indeed. It checks that n-ary predicates have all their n references.

C. Nouns without Reference Index

Not only process words, but also nouns need to be referenced and specified completely so that they do not represent super groups or sub groups of certain sets. The focus is especially on the articles `a` and `the` which are replaced with specific declarations like `every`, `five`, `nobody`, `none`, ... after involving the user via the graphical interface.

D. Incomplete Specified Conditions

When using conditional branching, the system needs to make sure that every condition does have its full set of possible branches to consider. It needs to check that every if-branch has an else-branch. Correctly specifying these conditions eliminates the problem of implicit knowledge which could lead to fatal assumptions and errors.

E. Modal Operators Expressing Necessity

Modal operators express conditions which ought to happen. These condition must not appear without the definition what to do in the exceptional case when the desired behavior cannot be met.

F. Implicit Assumptions (Presumptions)

Conducting the requirements elicitation and engineering process often confronts the analyst with the *curse of knowledge*. It leads to implicit assumptions on requirements and their aspects which are followed by misinterpretations. Finally, it could lead to omitting valuable information during the requirements description. RESI needs to make sure that relations between objects are adequately specified. For example: Every property (thematic role *HAB*) must have an owner (thematic role *POSS*) assigned.

All the above challenges can be addressed by RESI. For evaluation purposes, the system would also have to fulfill other functions. Since the system makes changes on the natural language specifications itself (or its model representation, to be precise), the changes need to be traceable. The specification needs to be legible after the improvements to the model have been made. This ensures

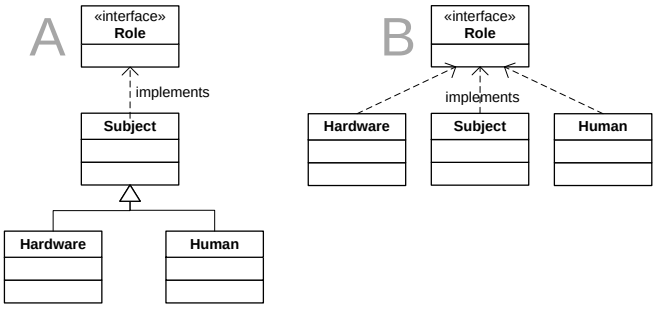

Fig. 4. Partial UML Representation of OMG's "Actor"

the usability for the stakeholders. The queries deliver results from various ontology sources to one specific problem. The results of these queries need to be combined and presented to the analyst in a fast and easy way. This turns into a problem if the results contradict each other. When using ontology systems with different degrees of generality, this behavior is not uncommon. A meaningful and sensible set of parameters for the treatment of queries and their results is required.

VI. Case Study

This section shows examples from real world specifications. For an evaluation of the concept proposed in this paper, we took several freely available software specifications and discussed them in a group of requirements analysts. We realized that there are various viewpoints to certain aspects. We argued about the meaning of the terms in the sentences and monitored the decision making mechanism used in our discussions. After the specifications had been reviewed, we checked our list of decisions and compared them to the information we retrieve from various ontologies. By applying the correct queries, many question can be answered with ontologies. Exemplary abstracts of the used specifications are listed below.

A. Object Management Group UML Specification

The Object Management Group's UML specification [30] describes the figure "Actor" in the class description of use cases (see paragraph 16.3.1) as the following:

> **(1) An Actor models a type of role played by an entity that interacts with the subject [...], but which is external to the subject (i.e., in the sense that an instance of an actor is not a part of the instance of its corresponding subject).**
>
> **(2) Actors may represent roles played by human users, external hardware, or other subjects.**

The first sentence has the verb `interact`. It is not clear whether it is `type of role` that interacts or the `entity`. RESI asks the user which role to denote to the corresponding objects. The second sentence mentions the `other subjects`. It is not clear whether `human users` and `external hardware` are also `subjects`. If they are, Figure 4A would be a correct UML representation. Otherwise Figure 4B could be a correct solution. The user has to be asked which deduction is feasible.

B. W3C HTML Specification

The W3C[4] states the following in the HTML specification [31]:

> **(1) An HTML form is a section of a document containing normal content, markup, special elements called controls (checkboxes, radio buttons, menus, etc.), and labels on those controls.**
> **(2) Users generally "complete" a form by modifying its controls (entering text, selecting menu items, etc.), before submitting the form to an agent for processing (e.g., to a Web server, to a mail server, etc.)**
> **(3) Users interact with forms through named controls.**

The verb `contain` in sentence (1) could reference `section` or `document`. When reading this in a specification, RESI has to ask which reference to take. The specification has a quote "`complete`". Its preposition `by` denotes that the term `modifying its controls` specifies the term `complete` in more detail. Quotes are non-specific and ought not to be used for the sake of precision in sentences. RESI points out this flaw to the user.

C. Ludo Specification

The textual specification of the game Ludo can be found in [32]. We list a short excerpt of the specification here:

> **(1) There are four players in a cyclic order: red, blue, yellow, and green.**
> **(2) If one of the following moves is possible, the player must choose one**

Humans realize that the colors in sentence (1) could be modeled as a single attribute (e.g. *color*) with an enumeration of values (*red, blue, yellow, green*). Ontologies know colors. Therefore RESI does not model four different attributes but recommends to group these attributes. The approach is obvious in this case, but becomes a lot more valuable once the analyst is not a domain expert. This disadvantage could be alleviated with a system that reads ahead and hints to conceptual relatedness as shown in [8].

The second sentence states `If one` [...] `is possible` and shows a familiar problem that most specifications possess: incomplete conditions. We worked out the details of incomplete conditions in Section IV-C and Section V-D. RESI should raise the question "What happens if none of the moves are possible?".

VII. Conclusion

The area of requirements engineering is important to any project. Until today, research focuses on improving the error-prone and complex manual processes that humans have to carry out. The elicitation of requirements and the review of the results still lie in the hands of the analyst.

Software systems have huge requirement collections that span hundreds if not thousands of pages. The probability that

[4]World Wide Web Consortium

facts are overlooked, misunderstood, or not completely specified is high. Many projects tend to go over time and budget because requirement specifications are faulty or incomplete. Most of the times these errors emerge at a later stage in the software development process and lead to immense costs for change requests. Formal approaches try to deliver a solution, but turn out to be quite complex to use.

User adaption is very important: all kinds of stakeholders with different levels of background knowledge need to interact during the requirements elicitation and engineering. Requirements are the interface between the expert and the customer. If we want to make sure that the customer can understand the requirements, we need to provide textual specifications which can be revised by the customer yet provide the requirements analyst and the development team with the necessary information for their work. Speeding up and improving this procedure by using ontology based recommender systems such as RESI seems a logical consequence. Our studies have shown that this solution is feasible and we are already implementing the software system that addresses the issues listed in this paper. RESI also supports the analyst by asking questions which limits the risk of overseeing aspects of the specification.

In the future, research in artificial intelligence will improve the algorithms for deduction as well as the speed and coverage of ontologies. The number of semantic applications increases and storage space and memory needed for the successful application of ontologies and their functions become more affordable.

ACKNOWLEDGMENT

AUTOMODEL is funded by ec4u expert consulting ag, Germany in cooperation with the University of Karlsruhe, Germany.

REFERENCES

[1] C. Rupp and R. Goetz, "Psychotherapy for System Requirements," *Proceedings of the Second IEEE International Conference on Cognitive Informatics (ICCI '03)*, 2003.

[2] B. H. C. Cheng and J. M. Atlee, "Research Directions in Requirements Engineering," in *Proc. Future of Software Engineering FOSE '07*, 23–25 May 2007, pp. 285–303.

[3] C. Rupp, "Requirements and Psychology," *IEEE, May/June 2002*, vol. IEEE SOFTWARE, 2002.

[4] B. Nuseibeh and S. Easterbrook, "Requirements engineering: a roadmap," in *ICSE '00: Proceedings of the Conference on The Future of Software Engineering*. New York, NY, USA: ACM Press, 2000, pp. 35–46.

[5] G. A. Miller, C. Fellbaum, R. Tengi, P. Wakefield, H. Langone, and B. R. Haskell, "WordNet." [Online]. Available: http://wordnet.princeton.edu/

[6] Cycorp Inc., *Cyc / ResearchCyc*, Cyc.com. [Online]. Available: http://cyc.com/

[7] C. Havasi, R. Speer, and J. Alonso, "ConceptNet 3: a Flexible, Multilingual Semantic Network for Common Sense Knowledge," in *Recent Advances in Natural Language Processing*, Borovets, Bulgaria, September 2007. [Online]. Available: http://web.media.mit.edu/~jalonso/cnet3.pdf

[8] S. J. Körner and T. Gelhausen, "Improving Automatic Model Creation using Ontologies," in *Proceedings of the Twentieth International Conference on Software Engineering & Knowledge Engineering*, Knowledge Systems Institute, Ed., Jul. 2008, pp. 691–696.

[9] 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*, 2007, pp. 282–289.

[10] A. van Lamsweerde, R. Darimont, and E. Letier, "Managing conflicts in goal-driven requirements engineering," vol. 24, no. 11, pp. 908–926, Nov. 1998.

[11] H. Kaiya and M. Saeki, "Ontology Based Requirements Analysis: Lightweight Semantic Processing Approach," in *Proc. Fifth International Conference on Quality Software (QSIC 2005)*, 19–20 Sept. 2005, pp. 223–230.

[12] ——, "Using Domain Ontology as Domain Knowledge for Requirements Elicitation," in *Proc. th IEEE International Conference Requirements Engineering*, 11–15 Sept. 2006, pp. 189–198.

[13] M. Saeki, "Ontology-Based Software Development Techniques," *ERCIM News*, vol. 58, pp. 14–15, 2004.

[14] Y. Zhang, R. Witte, J. Rilling, and V. Haarslev, "An ontology-based approach for traceability recovery," 2006.

[15] W. Meng, J. Rilling, Y. Zhang, R.Witte, and P. Charland, "An Ontological Software Comprehension Process Model," *3rd International Workshop on Metamodels, Schemas, Grammars, and Ontologies for Reverse Engineering (ATEM 2006). October 1st, Genoa, Italy*, 2006.

[16] H. Liu and P. Singh, "ConceptNet - a practical commonsense reasoning tool-kit," *BT Technology Journal*, vol. Vol 22, 2004. [Online]. Available: http://larifari.org/writing/BTTJ2004-ConceptNet.pdf

[17] Volere, "List of requirement engineering tools," 2009. [Online]. Available: http://www.volere.co.uk/tools.htm

[18] S. Konrad and B. H. Cheng, "Facilitating the Construction of Specification Pattern-based Properties," *IEEE International Conference on Requirements Engineering*, pp. 329–338, 2005.

[19] A. Pease and W. Murray, "An English to Logic Translator for Ontology-based Knowledge Representation Languages," *IEEE 0-7803-7902-0/03*, pp. 777–783, 2003.

[20] C. L. Heitmeyer, R. D. Jeffords, and B. G. Labaw, "Automated Consistency Checking of Requirements Specifications," *ACM Trans. Softw. Eng. Methodol.*, vol. 5, no. 3, pp. 231–261, 1996.

[21] A. Fantechi, S. Gnesi, G. Lami, and A. Maccari, "Application of Linguistic Techniques for Use Case Analysis," *IEEE International Conference on Requirements Engineering*, p. 157, 2002.

[22] W. N. Robinson and S. D. Pawlowski, "Managing requirements inconsistency with development goal monitors," vol. 25, no. 6, pp. 816–835, Nov.–Dec. 1999. [Online]. Available: http://www.cis.gsu.edu/~wrobinso/papers/TSE99.PDF

[23] L. Goldin and D. M. Berry, "AbstFinder, A Prototype Natural Language Text Abstraction Finder for Use in Requirements Elicitation," *Automated Software Engg.*, vol. 4, no. 4, pp. 375–412, 1997.

[24] C. Denger, D. M. Berry, and E. Kamsties, "Higher Quality Requirements Specifications through Natural Language Patterns." Los Alamitos, CA, USA: IEEE Computer Society, 2003, p. 80.

[25] N. Kiyavitskaya, N. Zeni, L. Mich, and D. M. Berry, "Requirements for tools for ambiguity identification and measurement in natural language requirements specifications," *Requir. Eng.*, vol. 13, no. 3, pp. 207–239, 2008.

[26] M. Ceccato, N. Kiyavitskaya, N. Zeni, L. Mich, and D. M. Berry, "Ambiguity Identification and Measurement in Natural Language Texts." [Online]. Available: http://eprints.biblio.unitn.it/archive/00000727/

[27] T. Gelhausen, B. Derre, M. Landhäußer, T. Brumm, and S. J. Körner, "SaleMX Project Website - the Thematic Roles of SALE," 2008. [Online]. Available: http://svn.ipd.uni-karlsruhe.de/trac/mx/wiki/MX/SALE/ThematicRoles

[28] T. Gelhausen, B. Derre, and R. Geiss, "Customizing GrGen.NET for Model Transformation," in *GRaMoT '08: Proceedings of the 3rd International Workshop on Graph and Model Transformation*. Leipzig, Germany: ACM, May 2008, pp. 17–24. [Online]. Available: http://www.ipd.uka.de/Tichy/uploads/publikationen/180/gramot2-gelhausen.pdf

[29] Chris Rupp & die SOPHISTen, *Requirements-Engineering und Management*, 4th ed. Carl Hanser Verlag, 2006.

[30] Object Management Group, "Unified Modeling Language: Superstructure Version 2.1.1 16.3 Class Description 16.3.1 Actor," PDF, February 2007. [Online]. Available: http://www.omg.org/docs/formal/07-02-03.pdf

[31] W3C, "HTML 4.01 Specification, 17 Forms," December 1999. [Online]. Available: http://www.w3.org/TR/html401/interact/forms.html

[32] M. Kroll and R. Geiss, "A Ludo Board Game for the AGTIVE 2007 Tool Contest," 2007. [Online]. Available: http://www.informatik.uni-marburg.de/~swt/agtive-contest/Ludo-Karlsruhe.pdf