Requirements Management with Semantic Technology: An Empirical Study on Automated Requirements Categorization and Conflict Analysis

Thomas Moser, Dietmar Winkler, Matthias Heindl, and Stefan Biffl

Christian Doppler Laboratory
Software Engineering Integration for Flexible Automation Systems
Institute of Software Technology and Interactive Systems
Vienna University of Technology, Vienna, Austria
{firstname.lastname}@tuwien.ac.at

Abstract. Requirements managers aim at keeping the set of requirements consistent and up to date throughout the project by conducting the following tasks: requirements categorization, requirements conflict analysis, and requirements tracing. However, the manual conduct of these tasks takes significant effort and is error-prone. In this paper we propose to use semantic technology as foundation for automating the requirements management tasks and introduce the ontology-based reporting approach OntRep. We evaluate the effectiveness and effort the OntRep approach based on a real-world industrial empirical study with professional Austrian IT project managers. Major results were that OntRep provides reasonable capabilities for the automated categorization of requirements, was when compared to a manual approach considerably more effective to identify conflicts, and produced less false positives with similar effort.

Keywords: Requirements categorization, requirements conflict analysis, consistency checking, requirements tracing, case study, empirical evaluation.

1 Introduction

A major goal of requirements engineering is to achieve a common understanding on the set of requirements between all project stakeholders. Modern IT projects are complex due to the high number and complexity of requirements, and geographically distributed project stakeholders with different backgrounds and terminologies. Therefore, adequate requirements management (ReqM) tools are a major contribution to address these challenges. Current ReqM tools typically work with a common requirements database, which can be accessed by all stakeholders to retrieve information on requirements content, state, and interdependencies.

ReqM tools help project managers and requirements engineers to keep the overview on large amounts of requirements by supporting: (a) *Requirements categorization* by clustering requirements into user-defined subsets to help users find relevant requirements more quickly, e.g., by sorting and filtering attribute values; (b) *Requirements conflict analysis* (or consistency checking) by analyzing requirements from different

H. Mouratidis and C. Rolland (Eds.): CAiSE 2011, LNCS 6741, pp. 3–17, 2011.

stakeholders for symptoms of inconsistency, e.g., contradicting requirements; and (c) *Requirements tracing* by identifying dependencies between requirements and artifacts to support analyses for change impact and requirements coverage. Unfortunately, ReqM suffers from the following challenges and limitations:

- Incompleteness [7] of requirements categorization and conflict identification, in particular, when performed manually.
- High human effort for requirements categorization, conflict analysis and tracing, especially with a large number of requirements [7].
- Insufficient completeness [6] for conflict analysis and tracing with automated approaches.
- Tracing on syntactic rather than on concept level: requirements are often traced on the syntactic level by explicitly linking requirements to each other. However, requirements engineers actually want to trace concepts, i.e., link requirements based on their meaning, which can be achieved only partially by information retrieval approaches like "keyword matching" [12] [13].

The use of semantic technologies seems promising to address these challenges: Ontologies provide the means for describing the concepts of a domain and the relationships between these concepts in a way that allows automated reasoning [18]. Automated reasoning can support tasks for requirements categorization, requirements conflict analysis, and requirements tracing.

In this paper, we propose OntRep, an automated ontology-based reporting approach for requirements categorization, conflict analysis and tracing based on ontologies and semantic reasoning mechanisms. The main criteria for the evaluation are: correctness and completeness of identified requirements conflicts, effort to develop a project or domain ontology. OntRep aims at lowering the effort for requirements management, while keeping high requirements consistency.

The OntRep approach automatically categorizes requirements into a given set of categories using ontology classes modeled in Protégé and mapping the terms used in the requirements to these classes. Further, OntRep analyzes the content of the requirements and identifies conflicts between requirements. Therefore, conflict analysis is not only based on traditional keyword-matching-approaches, but can also work when different terminologies are used for requirements formulation.

We empirically evaluate OntRep with a real-life project at Siemens Austria, where six project managers in two teams (a) categorized the requirements of the case study project into a set of categories and (b) inspected the given project requirements to identify conflicts between requirements. A requirements engineering expert provided control data for all tasks. Then, we performed the same tasks with OntRep to compare the effort necessary and the quality of results.

The remainder of the paper is organized as follows: Section 2 summarizes related work on requirements categorization, conflict analysis, tracing, and natural language processing technologies; Section 3 introduces the OntRep approach and motivates research issues. Section 4 outlines the case study and Section 5 presents results. Section discusses the results, concludes and suggests further work.

2 Related Work

This section presents related work on natural language processing technologies as foundation for automating the ReqM tasks requirements categorization, conflict analysis, and requirements tracing approaches.

2.1 Requirement Conflicts Detection and Requirements Tracing

Requirements conflict with each other if they make contradicting statements about common software attributes [7]. Requirements authors may use different terminologies for specifying requirements, although the terms used can be derived from the same common concepts.

- In principle there are the following main strategies to identify and eliminate requirements conflicts: Negotiation methods, where stakeholders manually (or with tool support) categorize, discuss, and analyze requirements for conflicts, such as the win-win requirements negotiation approach [1] or its tool-supported variant easy-win-win [2],
- Automation approaches for conflict analysis ([4][6][12]) that use tools to analyze requirements consistency in order to reduce human effort.

"Given that there may be up to n^2 conflicts among n requirements, the number of potential conflicts, could be enormous, burdening the engineer with the time-intensive and error-prone task of identifying the true conflicts" [7]. Several approaches address the issue of automated requirements conflict identification:

The *Trace Analyzer* by Egyed and Grünbacher [7] analyzes the footprint of test cases to generate trace dependencies. If two requirements affect the same part of a system, then their test runs execute overlapping lines of code. Trace dependencies and potential conflicts can be identified among requirements, if their test scenarios execute the same lines of code. However, the Trace Analyzer needs executable code to identify requirements conflicts, which is often not available in early project phases, when conflict analysis is a major goal.

Heitmeyer et al. [11] describe a formal analysis technique, called *consistency checking*, for the automated detection of errors, such as type errors, non-determinism, missing cases, and circular definitions, in requirements specifications. The approach only considers syntactical consistency and does not address semantic conflicts.

Automated requirements tracing approaches are also relevant for requirements conflict analysis: requirements tracing deals with identifying interdependencies between requirements [10] and conflicts between two requirements can be seen as a particular type of interdependency, i.e., tracing is a precondition for conflict analysis. There are reports on several trace automation approaches, such as Egyed's scenario-driven approach to traceability [6], Jackson's key-phrase-based traceability scheme [12]. Further, there are the heterogeneous traceability approach Cleland-Huang et al. [4], and approaches by Pinheiro et al. [20], Leuser [14], and McMillan et al. [16]. These approaches use different techniques to identify requirements interdependencies. Some of them require executable code, so they cannot be used for the identification of interdependencies in early project phases when there is no sufficient code base.

Within these trace automation approaches, information retrieval approaches, such as the RETH approach [13], seem of particular interest as they use keyword-matching techniques to identify requirements interdependencies. However, these techniques do not allow identifying conflicts or other interdependencies between requirements, if they use different terms for similar concepts. In practice these approaches are less effective, because they cannot identify the full set of interdependencies between requirements.

The extended Bakkus-Naur-Form (EBNF) [21] (see Fig. 2) is a general formal language description approach, which is used in the field of requirements analysis to improve the understandability of requirements for humans and machines. EBNF requirements templates contain mandatory and optional elements, e.g. conditions, obligations, actors, process verbs, which are the basis for clear requirements statements.

2.2 Natural Language Processing

Natural language processing (NLP) techniques are useful to parse and extract structure and content of requirements given in natural language for transformation into the structure of an ontology. NLP generally refers to a range of theoretically motivated and computational techniques for analyzing and representing naturally occurring texts [3]. The core purpose of NLP techniques is to achieve human-like language processing for a range of tasks or applications [15].

The core NLP models used in this research are part-of-speech (POS) tagging and sentence parsers [3]. POS tagging involves marking up the words in a text as corresponding to a particular part of speech, based on both its definition, as well as its context. In addition, sentence parsers transform text into a data structure (also called parse tree), which provides insight into the grammatical structure and implied hierarchy of the input text [3]. *Standford parser/tagger*[1] and *OpenNLP*[2] are the core set of NLP tools used in this research.

Another tool that can be used is *WordNet*, a large lexical database of English [17]. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. WordNet is a useful building block for requirements analysis (see below).

These NLP technologies can be used for our purpose, namely to improve the effectiveness of requirements management activities like categorization, conflict analysis, and tracing.

3 Ontology-Based Reporting

Due to the limitations of requirements analysis approaches that address only links between requirements based on syntactic equality, we explore in this work an approach based on semantic equality, i.e., OntRep links similar concepts, if they share the same meaning even if their syntactic representations are different. As ontologies are versatile for representing knowledge on requirements and for deriving new links

[1] http://nlp.stanford.edu/software/lex-parser.shtml
[2] http://opennlp.sourceforge.net

between requirements, we introduce an ontology-based approach for reporting analysis results on a set of requirements, so-called ontology-based reporting.

The goal of the ontology-based reporting approach *OntRep* is making ReqM tasks like requirements categorization, conflict analysis and requirements tracing more efficient based on the automation of selected steps in these tasks. The following subsections provide an overview on the approach and motivate research issues.

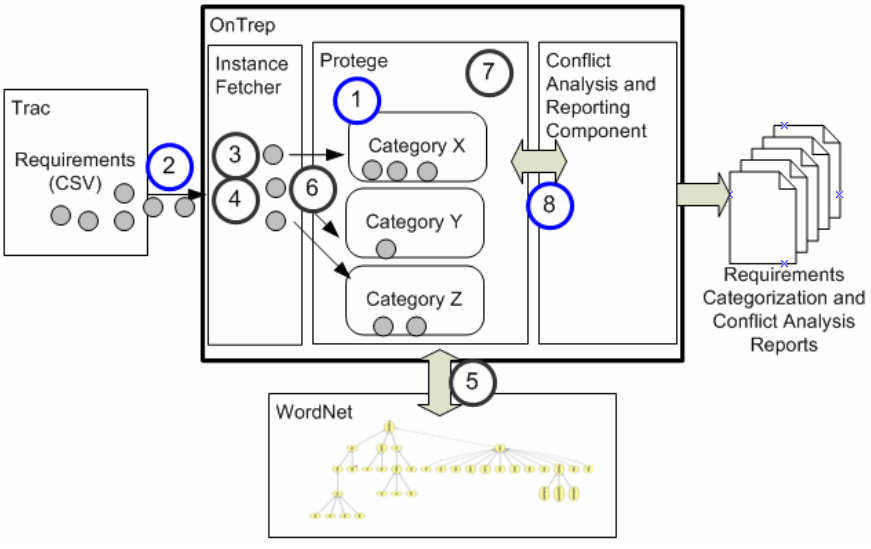


Fig. 1. Components and numbered steps of OntRep

We developed a prototype tool for the *OntRep* approach a plug-in to *Trac*[3], an open source collaboration platform consisting of a Wiki, ticket management system, and subversion integration, which can be extended by Python plug-ins.

Fig. 1 illustrates the *OntRep* tool (together with *Protégé*) consisting of two main components: 1) *Instance fetcher* takes input data, e.g., requirement tickets from *Trac*, analyzes their contents and assigns them requirements categories (classes) defined in the ontology; 2) *Reporting component* reasons on the input data and generates a requirements conflict report based on the analyzed requirements.

3.1 Semantic Requirements Categorization

In a first phase natural language texts have to be linked to semantic categories as preparation for further analysis and reporting. The following steps automate requirements categorization with OntRep (see numbered circles in Fig. 1):

1) *Define the requirement categories* in Protégé, e.g., categories X, Y, Z. Each category is defined as an ontology class in Protégé. It is important to define project-relevant "semantic" categories and not formal ones in order to enable the automated

[3] http://trac.edgewall.org/

categorization, e.g., “Security”. Typically, these categories can be defined based on a project glossary that contains important project-specific terms.

2) *Provide input data* to be categorized: Requirements are represented as tickets in *Trac*. For our research prototype we export these requirements (the small grey circles in Fig. 1) via CSV from *Trac* and import them into the instance fetcher.

3) *Remove irrelevant stop-words*, like “and”, “any”, “but”, which cannot be used for categorization. This step is performed automatically using a standard stop-word list[4].

4) *Bring all remaining words into their root form*: this process is called “stemming” based on a well-known algorithm, like the “Porter Stemmer” algorithm [19]. An example is to stem “jumping” to “jump”.

5) *Get all synonyms and hyponyms* of the analyzed words in the requirements by using the natural language processing library “WordNet” [17]. For example, “house” is a synonym for “building”, “dog” is a hyponym of “animal”. Further check all relevant substrings of a word like “net” as a substring of “network”.

6) *Heuristic-based assignment of each requirement to the defined categories* depending on the number of hits for 1) synonyms, 2) hyponyms and 3) substring matches. The heuristic checks if the hits for synonym, hyponym and substring matches meet the given threshold values. So the number of met thresholds is between 0 and 3. If this number is equal of higher than the number of thresholds that must be met, the word will be related to that category, otherwise not. If several categories reach these thresholds, the requirement will be categorized into all of these categories (multi-dimensional categorization is allowed)

7) *Save the element as an individual of the ontology class*, if it is not already in the class. This can only be checked if one or more of the elements attributes have been declared as primary keys (uniquely identifying the element). If the element has already been saved in another class as well (which could be the case), declare that the new element is the same as the already existing one with the *“owl:sameAs”* property.

8) *Semantic requirements conflict analysis:* If the requirements are formally described using a specified grammar (e.g., EBNF), the information contained in the textual requirement descriptions can be semantically analyzed in order to identify possible inconsistencies and/or conflicts, see subsection 3.2.

3.2 Semantic Conflict Analysis

In the second phase, analysis and reporting approaches build on the mapping of requirements to semantic categories. For formally specified requirement semantics, in our case following an EBNF template (see Fig. 2), semantic analysis can identify inconsistencies and conflicts using a set of assertions that should hold true for all available facts. These assertions are based on the available requirements, while the available facts are based on the environment and properties of the target system.

Fig. 3 depicts examples for conflicts between requirements (the CRR conflict type explained in section 4), e.g. the third conflict contains an inconsistency between requirements nr. 16 and nr.13: The “thing to be processed” part of requirement nr. 16 contains a value of 30 for “number of index updates”, whereas requirement nr. 13 contains the value 20 for “number of index updates, which finally is a requirements conflict.

[4] http://www.textfixer.com/resources/common-english-words.txt

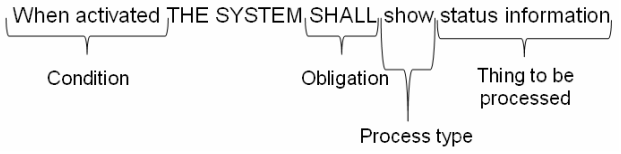
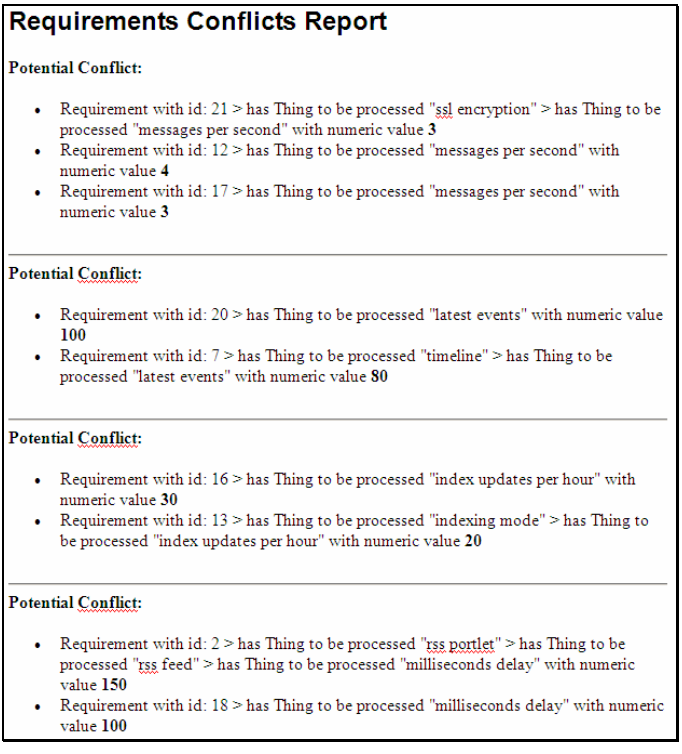


Fig. 2. EBNF requirements structure (sentence level) [21]



Requirements Conflicts Report

Potential Conflict:

- Requirement with id: 21 > has Thing to be processed "ssl encryption" > has Thing to be processed "messages per second" with numeric value **3**
- Requirement with id: 12 > has Thing to be processed "messages per second" with numeric value **4**
- Requirement with id: 17 > has Thing to be processed "messages per second" with numeric value **3**

Potential Conflict:

- Requirement with id: 20 > has Thing to be processed "latest events" with numeric value **100**
- Requirement with id: 7 > has Thing to be processed "timeline" > has Thing to be processed "latest events" with numeric value **80**

Potential Conflict:

- Requirement with id: 16 > has Thing to be processed "index updates per hour" with numeric value **30**
- Requirement with id: 13 > has Thing to be processed "indexing mode" > has Thing to be processed "index updates per hour" with numeric value **20**

Potential Conflict:

- Requirement with id: 2 > has Thing to be processed "rss portlet" > has Thing to be processed "rss feed" > has Thing to be processed "milliseconds delay" with numeric value **150**
- Requirement with id: 18 > has Thing to be processed "milliseconds delay" with numeric value **100**

Fig. 3. Example report on requirements conflicts

3.3 Research Issues

The underlying idea of this research is to use advanced semantic technologies, like ontologies and reasoning mechanisms, to increase the effectiveness and efficiency of ReqM activities. In a large software project, tasks like requirements categorization, conflict analysis, and tracing would need human effort and duration that often prohibits their use in practice. Therefore, software projects often end up with (a) unstructured requirements and (b) conflicts that get discovered late and expensively.

In this context, the main research question of this paper is: To what extent can a semantic-technology-based approach, like OntRep, increase the effectiveness and efficiency of requirements categorization and conflict analysis compared to a traditional

manual approach? In order to address the research question we derived the following variables (according to [8]) to consider for evaluation:

- The *number of requirements* determines the effort necessary for categorization and conflict analysis.
- *Number of requirement categories* used to categorize the requirements. Further, the *total number of true requirements conflicts* existing in a list of requirements, which can be identified by various approaches for conflict detection. This is a baseline measurement for the effectiveness of an approach, i.e., a perfect approach would find 100% of the true requirements conflicts.
- *Approach for categorization/conflict analysis*: e.g., for automation approaches the formal structure of requirements is an important factor. As described above, we use the EBNF template for specifying requirements. Using plain text or other formats probably affects the correctness and completeness of identified conflicts.

Dependent variables that we want to study by the evaluation are:

- *Number of conflicts identified*. This number consists of two measures: recall (number of correctly identified conflicts), for measuring the effectiveness of an approach, and false positives (number of wrongly identified conflicts).
- *True conflicts that have not been identified* (false negatives): subtracting the *number of correctly identified conflicts* from the *total number of true requirements conflicts* tells us about the recall of conflict identification.
- *Plausibility of requirements classification*: regarding categorization two kinds of error can occur: (1) requirements have been assigned to a wrong category, and (2) requirements have not been assigned to a category they actually belong to. In order to measure these parameters, we take the manual categorization results of a requirements engineering expert as reference. In addition, we count the *number of requirements in each category.*

Besides these parameters we also record the effort for requirements categorization and conflict analysis. This includes *preparation effort* (e.g., creating the ontology that is used for categorization and conflict analysis), *categorization effort*, and *conflict analysis effort*. The case study is described in detail in the following section.

4 Case Study Description

The following subsections describe the characteristics of the pilot study design.

Study Subject. The case study project “Technoweb 2.0” is an IT development project with the goal to design and implement a web application that serves as a platform for communication and networking between technology experts within Siemens. This platform builds on the Java technology *Liferay*, where *portlets* act as components of the web application. The project is performed in an agile way using the software development process “SCRUM” and the configuration & project management platform *Trac*. In Trac all requirements, tasks and bugs are stored as tickets.

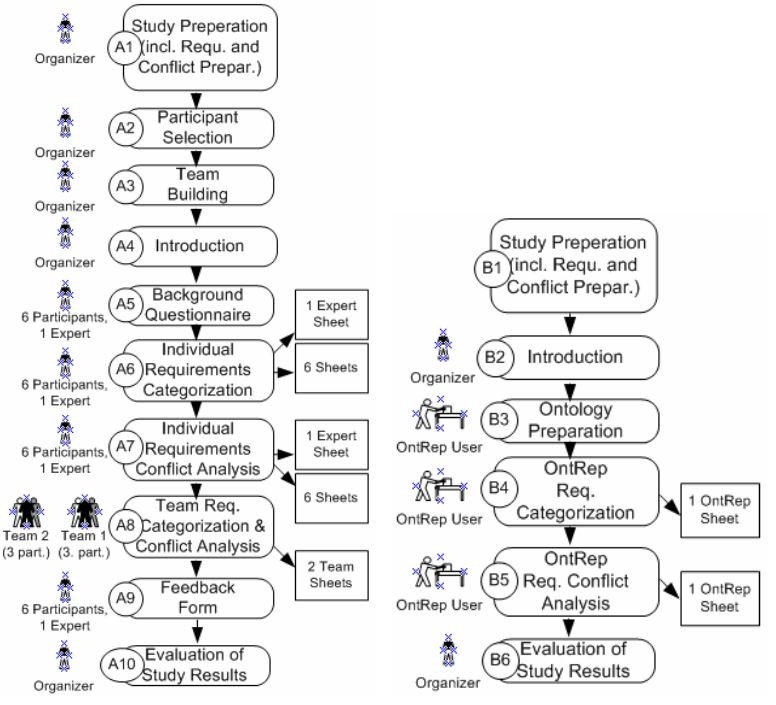


Fig. 4. Evaluation Setting: Manual (left), OntRep (right)

Study Design, Material, Participants, and Process. We applied the standard practices of empirical software engineering research according to Freimut et al. [8] and Wohlin et al. [22]. Fig. 4 illustrates the steps of the empirical study.

A1, B1) *Study preparation:* This step dealt with the creation/preparation of all *artifacts* necessary for the evaluation: 23 requirements in EBNF syntax, 8 categories as input for the requirements categorization step, deployment of 22 seeded conflicts (based on typical requirements conflicts found in practice at Siemens Austria). Further, we used questionnaires to capture the individual background experience of the participants and a feedback questionnaire to capture, whether the participants found the approach useful and usable.

A2, A3) *Participant selection and team building*: There were 6 participants who performed the manual categorization and conflict analysis tasks. They had similar experience on project management (3 to 5 years) and on requirements management but advanced general software engineering know-how. In addition, we had one expert with deeper know-how and experience, especially in ReqM. Finally, there was one OntRep tool user. He was well familiar with OntRep and had similar experience as the other participants. As described below, the evaluation consisted of individual work and team work. For the latter, the teams were assigned randomly.

A4, B2) *Introduction (Guidelines and Data Collection):* Before execution of tasks the study organizer introduced the participants to the project and the manual requirements categorization and conflict detection tasks. Further, the participants were

guided step-by-step through the requirements classification and conflict detection process. The participants were sitting in one room without talking to each other. In the team phase, the two teams (3 participants each) worked in separated rooms. The expert, as well as the OntRep tool user, also worked separately.

A5) *Background questionnaire:* before they started with the actual ReqM tasks, the participants filled in the questionnaire.

A6) *Individual requirements categorization:* Then the participants read through the 23 given requirements. The participants individually categorized the requirements into one or more of the given 8 categories. Each participant conducts the requirements categorization individually. In addition, one Requirements Engineering expert also does the categorization. The time needed by each participant is captured.

A7) *Individual requirements conflict analysis:* In addition to the 23 requirements that had been categorized before, further elements were displayed (as rows below the other requirements), namely: 11 constraints (technical and business), and 4 formal documentation rules (documentation guidelines).

The participants again read through the task description and then had to identify conflicts and enter them into the sheet. A conflict can have one of the following types: conflict between requirements (CRR), conflict of a requirement with a constraint (CRC), conflict of a requirement with a formal guideline, i.e., ill-formed requirement (CRG). In total, the case study data contained: 5 conflicts of type CRR, 7 of type CRC, and 10 of type CRG. After the evaluation of the manual approach, we again have the 6 individual results. Again, one Requirements Engineering expert also conducted the conflict analysis. The effort needed by each participant was captured.

A8) *Team requirements categorization & conflict analysis:* Afterwards, the participants harmonized their individual results within 2 randomly assigned groups. Effort was captured for this task. The results are 2 team sheets.

A9) *Feedback forms:* filled in at the end by the participants.

A10, B6) *Evaluation of study results*: The manually created results of the expert and the teams were then compared with the result generated by OntRep.

The process for the automated approach is:

B3) *Ontology preparation:* A tool expert created one ontology class in OntRep (Protégé) for each category and then imported the given requirements from Trac as CSV into OntRep.

B4) *OntRep requirements categorization:* The tool then executed the categorization and generated a final result. We captured the effort to create the ontology classes and to generate the final report.

B5) *OntRep requirements conflict analysis:* Then, we again provided the requirements as CSV-input to OntRep. Further, the tool expert had to model the constraints as facts and the formal guidelines as rules in the ontology. We captured the effort for this. Then, the tool executed the conflict identification and generated a final report.

Data Capturing Analysis and Statistical Evaluation. Finally, we analyzed and evaluated the following results: (a) 6 spreadsheets for requirements categorization and 6 spreadsheets for conflict analysis from each of the 6 individual participants, (b) 1 categorization spreadsheet and 1 conflict analysis spreadsheet from a requirement engineering expert, and finally (c) 1 categorization spreadsheet and 1 conflict analysis

spreadsheet created with the OntRep approach. The results were evaluated with descriptive statistics in Excel and R and are described in the following section.

5 Results

The following subsections describe the results of the pilot study regarding requirements categorization and conflict analysis.

5.1 Requirements Categorization

In order to evaluate the requirements categorization task, we took the categorization result of the requirements engineering expert as reference solution for comparing the results of the manual and automated approaches.

Table 1. Results of manual and automated req. categorization

	Individuals avg./std.dev.	**Groups** avg./std.dev.	**OntRep**
	MANUAL		AUTOMATED
1. Overfulfilled	9.5/3.9	**12.5/3,5**	6.0
2. Correct	5.7/2.7	6.0/2.8	**8.0**
Sum 1. & 2.	15.2/2.3	**18,5/0.7**	14.0
3. Partly correct	2.0/1.1	0.0/0.0	2.0
4. False	5.8/1.7	**4.5/0.7**	7.0

Table 1 summarizes the results of the manual and automated requirements categorization approaches: the rows in the table contain the quality levels of the categorization: "overfulfilled" means that a requirement was categorized into all correct categories but also into one or more additional ones, "correct" means that a requirement was categorized in the right categories. "Partially correct" means that a requirement was categorized in some but not all of the correct categories, "false" means that a requirement was categorized into wrong categories but not the right ones. The group results are better than the individual results: the number of false categorizations is reduced, and the number of correct and overfulfilled categorization is increased. Overfulfillment is not a problem, because all requirements are categorized into the right categories, and into some more categories, but this is just additional information which is allowed.

Categorization with OntRep was more accurate, i.e., 8 requirements (more than with the manual approach) have been categorized into the right categories without categorizing them in additional categories. On the other hand, comparing the sum of correctly categorized requirements (overfulfilled + correct) shows the lowest value for automation. Further, the number of false categorizations is also the highest. This is due to the fact, 4 requirements were not categorized at all. The reason therefore is that the terms used in these requirements could not be mapped to the categories, neither through substrings, synonyms or hyponyms.

The average effort for manual categorization was around 15 min per person. The group work took ca. 12 min in addition, resulting in an additional group effort of 36 person minutes. With OntRep the following preparations were necessary to enable the automated categorization: conversion of requirements into EBNF form (30 min.), preparation of ontology classes and user-defined synonyms (14 min.). After this, the run time for categorization was ca. 2 minutes. If the requirements exist in EBNF form, which is the case for some larger projects at Siemens Austria, the effort is similar to the manual average effort of manual categorization, but much more scalable.

5.2 Requirements Conflict Analysis

We analyzed conflicts of the three types described above, because conflicts of this type can be modeled in OntRep by means of facts and rules. The OntRep results for these conflicts are complete: all 22 conflicts of the defined conflict classes in the given data were identified, because OntRep works reliably, when the following prerequisite are met: requirements exist in EBNF as input via CSV, modeling of glossary terms in ontology (10 min.), modeling facts, constraints and rules in the ontology (46 min.). Therefore, the total OntRep preparation time is 100 min. for the given case. The overall report generation took 4 minutes.

Table 2. Results of conflict detection capability analysis

	# correctly identified conflicts (avg./ std.dev.)	Avg. % of true conflicts found (avg./ std.dev.)	# conflicts found (avg./ std.dev.)	False positives in % (avg./ std.dev.) of # conflicts found
Individuals	7.0/3.9	31.8/18	17.0/6.8	58.8/22.0
Groups	10.5/0.7	47.7/3.0	21.5/0.7	51.2/4.9
Expert	15.0	68.2	17.0	11.8
OntRep	**22.0**	**100.0**	**22.0**	**0.0**

In comparison, the manual conflict analysis approach resulted in a lower completeness (see Table 2): the individual participants identified only 31.8% of existing conflicts on average. The harmonization of results within the groups brought an improvement to 47.7%, which means that approximately 3 additional conflicts have been identified by merging of the individual results into one group result. Also the number of false positives was slightly reduced by 1. The correctness of the manual approach was also lower than with the OntRep approach: 58.8% of identified conflicts were false positives. This percentage could only slightly be reduced by the group harmonization. i.e., ca. 1-2 false positives were been eliminated during team work.

In addition to comparing the individual and group results with OntRep results, but also had one expert performing the conflict analysis. Compared to the other participants and the team results, he provided the best results, i.e., the highest number of correctly identified traces, and the lowest percentage of false positives. Regarding effort, the

expert was also the best with the manual approach: He needed 45 min. for conflict analysis, whereas the other participants needed 97 min. on average. In addition the group phase took 37 min., resulting in an additional group effort 111 person minutes.

5.3 Threats to Validity

We addressed threats internal validity [10] of the study by two measures: a) intensive reviews of the study concept and materials, and b) a test run of the study conducted by a test person in order to make sure that the guidelines, explanations, and task descriptions are understandable for the participants and to estimate the required effort/time frame. Regarding external validity [30], we performed this initial case study in a professional context at a software development company. The participants had medium requirements management know-how and advanced software engineering know-how. In addition, we had a requirements engineering expert as experimental "control group". Nevertheless, the small number of participants might limit the generalization of results. Therefore, we suggest replicating the study in a larger context.

Further, the requirements in this case study were formulated using the EBNF syntax, which is a major condition for OntRep to analyze the requirements. We did not yet analyze the quality of results with a set of requirements, which is not or only partially formulated in EBNF. Further studies are needed to evaluate this.

6 Discussion and Conclusion

Software and systems engineering projects are complex due to the increasing number and complexity of requirements, and the project participants with different domain backgrounds and terminologies. To keep the overview on requirements, project managers conduct requirements categorization, conflict analysis, and tracing. However, the manual conduct of these tasks takes significant effort and is error-prone.

In this paper we proposed semantic technology as foundation for automating the requirements management tasks and introduced the automated ontology-based reporting approach OntRep based on a project ontology and a reasoning mechanism. We used requirements formulated in EBNF as input to the proposed OntRep approach, which supports automated requirements *categorization* and requirements *consistency checking*. We evaluated the effectiveness and effort the OntRep approach based on a real-world industrial case study with 6 project managers in 2 teams. The study focused on requirements categorization and requirements conflict analysis. During the evaluation the study participants a) categorized the requirements of the case study project into a set of categories and b) inspected the given project requirements to identify conflicts between requirements. In addition a requirements expert and an OntRep user performed the same tasks to enable comparing the quality of results and the effort for all activities.

The case study results suggest that OntRep can be an attractive alternative for requirements categorization in typical software development projects, because it provides slightly lower effectiveness with similar effort compared to manual approaches, but much more scalable. OntRep's performance can be increased by adding additional, synonyms or hyponyms to the ontology (which has to be done manually at the

moment), so that all used terms in requirements can be mapped to categories. Regarding conflict analysis, OntRep found all conflicts in the requirements during the empirical study, while manual conflict analysis identified only 50 to 60% of the conflicts and produced more false positives with similar effort. OntRep analyzes three types of conflicts at the moment: conflicts between requirements, conflicts between requirements and some constraints, or conflicts of requirements with some formal guidelines. The OntRep automation approach seems beneficial for project managers who want to manage their requirements with less effort, but in the same turn keep the requirements consistency high. Using the OntRep approach, organizations in software development projects could benefit from reduced manual effort for categorization and conflict analysis, and reduced communication and clarification effort through semi-automated semantic conflict analysis support.

Further work will focus on the replication of this pilot study in a larger context, i.e., with more participants to improve the external validity of results. In addition, we want to increase the number of requirements to be categorized and analyzed for conflicts in order to analyze the correctness, completeness, and especially the effort for larger sets of requirements. We assume that especially the efficiency of OntRep will improve with the number of requirements when compared to a manual approach. Another aspect is to adapt OntRep for application to a set of requirements.

Acknowledgments. We want to thank Alexander Wagner for the prototype implementation of the OntRep concepts and his support during the pilot study. This work has been supported by the Christian Doppler Forschungsgesellschaft and the BMWFJ, Austria.

References

1. Boehm, B., In, H.: Identifying Quality-Requirement Conflicts. IEEE Software (1996)
2. Briggs, R.O., Grünbacher, P.: EasyWinWin: Managing Complexity in Requirements Negotiation with GSS. In: Proceedings of the 35th Hawaii International Conference on System Sciences (2002)
3. Choi, F.Y.Y.: Advances in domain independent linear text segmentation. In: Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference. Morgan Kaufmann Publishers Inc., Seattle (2000)
4. Cleland-Huang, J., Zemont, G., Kukasik, W.: A Heterogeneous Solution for Improving the Return on Investment of Requirements Traceability. In: 12th IEEE Int. Conf. on Requirements Engineering (2004)
5. Cruz, I.R., Huiyong, X., Feihong, H.: An ontology-based framework for XML semantic integration. In: International Database Engineering and Applications Symposium (IDEAS 2004), pp. 217–226. IEEE, Los Alamitos (2004)
6. Egyed, A.: A Scenario-Driven Approach to Traceability. In: Proceedings of the 23rd International Conference on Software Engineering (ICSE), Toronto, Canada, pp. 123–132 (2001)
7. Egyed, A., Grünbacher, P.: Identifying Requirements Conflicts and Cooperation: How Quality Attributes and Automated Traceability Can Help. IEEE Software (2004)
8. Freimut, B., Punter, T., Biffl, S., Ciolkowski, M.: State-of-the-Art in Empirical Studies, Report: ViSEK/007/E, Fraunhofer Inst. of Experimental Software Engineering (2002)

9. Gangemi, A., Guarino, N., Masolo, C., Oltramari, A.: Sweetening WordNet with DOLCE. AI Magazine 24(4), 13–24 (2003)
10. Gotel, O., Finkelstein, A.C.W.: An analysis of the requirements traceability problem. In: 1st International Conference on Requirements Engineering, pp. 94–101 (1994)
11. Heitmeyer, C.L., Jeffords, R.D., Labaw, B.G.: Automated consistency checking of requirements specifications. In: 2nd International Symposium on Requirements Engineering (RE 1995), York, England (1995)
12. Jackson, J.: A Keyphrase Based Traceability Scheme. IEEE Colloquium on Tools and Techniques for Maintaining Traceability During Design, 2-1-2/4 (1991)
13. Kaindl, H.: The Missing Link in Requirements Engineering. ACM SigSoft Software Engineering Notes 18(2), 30–39 (1993)
14. Leuser, J.: Challenges for semi-automatic trace recovery in the automotive domain. In: Proceedings of the ICSE Workshop on Traceability in Emerging Forms of Software Engineering, TEFSE (2009)
15. Liddy, E.D.: Natural Language Processing, 2nd edn. Encyclopedia of Library and Information Science. Marcel Decker, Inc., NY (2001)
16. McMillan, C., Poshyvanyk, D., Revelle, M.: Combining textual and structural analysis of software artifacts for traceability link recovery. In: Proceedings of the ICSE Workshop on Traceability in Emerging Forms of Software Engineering, TEFSE (2009)
17. Miller, G.A.: WordNet: A Lexical Database for English. Communications of the ACM 38(11), 39–41 (1995)
18. Pedrinaci, C., Domingue, J., Alves de Medeiros, A.K.: A core ontology for business process analysis. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 49–64. Springer, Heidelberg (2008)
19. van Rijsbergen, C.J., Robertson, S.E., Porter, M.F.: New models in probabilistic information retrieval, British Library Research and Development Report, no. 5587 (1980)
20. Pinheiro, F.A.C., Goguen, J.A.: An Object-Oriented Tool for Tracing Requirements. IEEE Software 13(2), 52–64 (1996)
21. Rupp, C.: Requirements Engineering und –Management. Hanser (2002)
22. Wohlin, C., Höst, M., et al.: Controlled Experiments in Software Engineering. Journal for Information and Software Technology, 921–924 (2001)