A Combined Similarity Measure for Determining Similarity of Model-based and Descriptive Requirements

Katharina Wolter [1] and **Thorsten Krebs** [2] and **Lothar Hotz** [3]

Abstract. RSL is a requirements specification language that was developed in the ReDSeeDS project. The language allows requirements specifications using a meta model that defines both model-based and descriptive representations. Model-based representations provide activity and sequence diagrams similar to UML while descriptive representations offer scenarios in constrained language and sentence lists. In this paper we tackle the problem of determining similarity of requirements that use both types of representations. We argue that in this case a combination of similarity measures is needed. In order to confirm this claim we assess similarity measures from different research areas with respect to their suitability for comparing requirements specifications written in RSL and suggest a concept for a combined similarity measure.

1 Introduction

Reuse is still an open problem in current software development practice. Former approaches to solve this problem concentrated on the code level. However, the impact of reuse can be increased when integrated earlier in the development process. The work presented in this paper is part of the ReDSeeDS[4] project in which the participants develop a framework that supports reuse on the level of requirements.

Starting with an initial requirements specification a repository is searched for similar specifications. This repository contains former software development projects stored in the form of software cases. A *software case* comprises a problem (requirements) and a solution (architecture, design and implementation), similar to case-based approaches, for example described in [3]. Each requirements specification is mapped to appropriate elements of the solution.[5] The retrieved case is intended to be reused by modifying those places that need rework and keeping those places that can be reused without modification. Retrieval of similar requirements specifications from a repository is a key prerequisite for this approach. Therefore, in this paper we integrate a combination of well-known AI techniques, like Case-based Reasoning, Description Logics, Ontology, and commonly used retrieval techniques, like Information Retrieval, with software reuse.

Our approach demands that the requirements specifications can be computationally processed, e.g. for automatically identifying similar specifications. For this reason requirements specifications based only on text files, which is often applied in software development, are not sufficient. In our approach the requirements specification language RSL is developed for enabling processable requirements specifications of a software case [9]. RSL allows different kinds of representations, being more and less formal. However, also the similarity measure, which is needed for retrieving software cases, has to suit such a requirements specification language. Thus, instead of relying on one similarity measure, in this paper we argue that a combination of different similarity measures is needed for determining similarity of requirements since RSL allows to specify requirements on different levels of formality. Therefore, we examine similarity measures from different research areas (i.e. Information Retrieval, Case-Base Reasoning, Graph-based Matching, and Description Logics) for their suitability with respect to different requirement representations.

Please note that this paper is based on earlier work [20] that already proposes to combine different similarity measures for determining similarity of model-based and descriptive requirements. This paper additionally presents results of validating the Requirements Specification Language (RSL) (see Section 2.2) and further details the concept for a combined similarity measure (see Section 4).

The remainder of this paper is organised as follows. Section 2 briefly introduces the requirements specification language. Section 3 describes similarity measures from different research areas and assesses their suitability for comparing requirements specifications written in RSL. Section 4 explains our concept for a combined similarity measure. Section 5 concludes the paper with a discussion and summary.

2 The Requirements Specification Language (RSL)

The Requirements Specification Language (RSL) is a result of joint work from the ReDSeeDS project. A complete introduction of RSL is beyond the scope of this paper. Instead, in this section we present those aspects that are relevant for the selection of similarity measures. For more details we refer to [9].

2.1 Overview of RSL

Typically, requirements specifications are written using natural language. Some approaches use constrained language in order to make the specification more precise and unambiguous.[6] RSL covers both approaches and allows to write requirements specifications in form of less formal *natural language hypertext* or more formal *constrained language sentences*. The constrained form of sentences is based on subject-verb-object (SVO) sentences and has the advantage of being syntactically unambiguous and semantically rich and thus, provides

[1] HITeC e.V., University of Hamburg, Germany, email: kwolter@informatik.uni-hamburg.de
[2] email: krebs@informatik.uni-hamburg.de
[3] email: hotz@informatik.uni-hamburg.de
[4] www.redseeds.eu, for a list of all participants we refer to the acknowledgements
[5] A complete description of a software case's internal structure is out of the scope for this paper. For a more detailed description of this approach we refer the interested reader to [15].
[6] Two examples of constrained language text are *subject-verb-object (SVO)* sentences (for example recently researched in [17, 16]) and the *Attempto Controlled English (ACE)* [5].

a better basis for the retrieval. RSL provides two types of requirement representations: *descriptive* and *model-based* requirement representations. The descriptive type of representation offers scenarios in constrained language and sentence lists. The model-based representations provide activity and sequence diagrams similar to UML. However, both are based on the constrained language sentences that are also used in scenarios.

A *sentence list* contains a list of sentences written in natural language hypertext or constrained language. Two sentence lists are equal if they contain the same sentences, independent from their order. A *scenario* contains a sequence of sentences written in constrained language. The order of sentences is important due to the fact that a scenario tells a story.

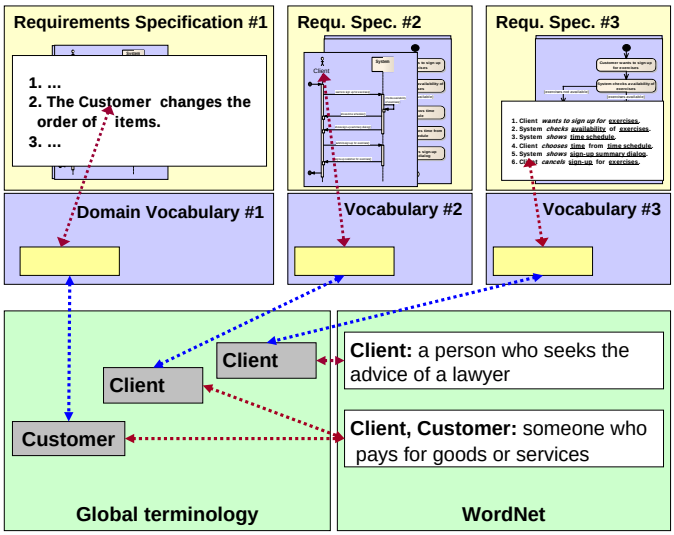

Figure 1. Requirements specifications may use different requirement representations. Every specification has its own domain vocabulary from which there are links to the global terminology. The meaning of words in the global terminology is specified using WordNet.

In RSL constrained language sentences contain links to a *domain vocabulary*, which is a software case-specific collection of notions that acts as a glossary and helps identifying all sentences in which the same notion appears. Links from natural language hypertext sentences to the domain vocabulary are also possible but not required.

In the ReDSeeDS framework, potentially reusable artefacts from former projects are identified based on the similarity of their requirements specifications. However, in different domains the same word may be used with a different meaning. In order to unambiguously define the meaning of specifications and make them available for persons not integrated in the particular development project a case-independent *global terminology* is used (see Figure 1). Links from elements of the domain vocabulary to words of the global terminology allow unambiguously comparing software case requirements from different domains. We use the semantic lexicon WordNet[7] to specify the meaning of words (see Section 3.3).

Besides the introduced elements, RSL provides a variety of additional elements all defined in a coherent metamodel, for example different types of sentences: *SVO sentence, Modal SVO sentence, conditional sentence, precondition sentence, postcondition sentence* and different types of domain vocabulary elements: *actor, notion, domain statement, noun phrase, verb phrase*.

[7] http://wordnet.princeton.edu

2.2 Validation of RSL

Validation Within the ReDSeeDS project, RSL was validated according to its applicability from the viewpoint of the industrial project partners. A workshop was organised in which tutorials were given, explaining RSL and how to specify requirements written in RSL using a prototypical RSL editor. Based on this introduction the participants from the industrial partners specified requirements from a real-life project. Questionnaires and surveys were prepared for querying practitioners at three different stages of the experiments phase: before and after the tutorials and after the participants created a requirements specification for their real-life projects. Using questionnaires and surveys at different stages allowed us to gain an overview of the practitioner's background and improvements during the experiments phase and assessing the applicability of RSL in real-life projects. In the following we briefly summarise the major results of validating the Requirements Specification Language. For more details we refer to [11].

Results Validating RSL has shown that the variety of requirement representations is needed for being applicable in real-life projects and acceptable for requirements engineers. All practitioners deem the possibility to create a domain vocabulary containing notions and domain statements to allow precisely defining the meaning of sentences by linking words to entries of the global terminology, e.g. relying on WordNet, helpful and a significant improvement to support real-life requirements engineering. Additionally, all practitioners think that the metamodel definition of RSL that is close to the well-known Unified Modelling Language (UML) improves understanding and applicability of RSL. But it was also found out that tool support is necessary for being able to efficiently specify requirements using RSL. Such a tool was available as prototype for the workshop and is currently implemented within the project.

3 Assessing Similarity Measures for Requirements written in RSL

One of the key features that distinguishes RSL from other approaches for requirements specification is that it offers different types of requirement representations, i.e. model-based and descriptive with the more formal constrained language sentences and the less formal natural language hypertext sentences. A single requirements specification can contain different types of representation. At the same time two requirements specifications can be conceptually equal, but have different representations.

While these different types of representations are a key feature of RSL, they are a special challenge for determining the similarity of requirements specifications. The measure must be able to handle all different types of representations provided by RSL. When the same requirement is represented differently in two specifications the similarity measure should still indicate the similarity in meaning.

Measures that capture the similarity of artefacts have been developed in many research communities for a variety of different types of artefacts (e.g. text documents, images, graphs). However, these similarity measures are typically developed for one specific type of artefact. Information Retrieval, for example addresses text-based documents while other approaches like SiDiff [10] compare model-based artefacts.

In order to support or disprove our argument that no single similarity measure is suitable for all representation types provided by RSL we reviewed measures from different research areas. In the following subsections we briefly introduce these measures but the main focus is

the evaluation of their applicability for our purpose, i.e. determining the similarity of requirements specifications in RSL. Due to space restrictions we can only discuss some of the assessment criteria used in our review. Even this subset of criteria is sufficient to support our claim that a combination of measures is needed:

- Which types of artefacts can be compared (e.g. text documents, images, graphs, UML models)?
- What is the meaning of a high similarity value?
- Are the ambiguity and paraphrase problems (see below) solved?

The *Ambiguity Problem* emerges when two artefact representations are the same but the actual meaning of the artefacts is different. The *Paraphrase Problem* describes the case where the artefact representations are different but the actual meaning of the artefacts is the same or at least similar [12].

These ambiguity and paraphrase problems are illustrated within Figure 1. Requirements specification #1 uses the word *customer* and Specifications #2 and #3 use the word *client*. But the actual meaning of the words *customer* and *client* in Specifications #1 and #2 is the same (synonyms cause the paraphrase problem); both words point to the same element in WordNet. Specifications #2 and #3 both use the word *client*, but with different meanings (homonym causes the ambiguity problem); the words link to different elements in WordNet.

3.1 Information Retrieval

Information Retrieval (*IR*) provides techniques for comparing text documents [8]. A major application domain of IR techniques is web search engines. Usually, large document collections are considered and the similarity measures are based on automatically generated document representations.

IR techniques can be applied to all artefact types that contain a reasonable amount of text. Traditional IR techniques consider artefacts equal if they contain the same words in the same frequency (stop words like determiner are not considered).

These techniques do not solve the ambiguity and paraphrase problem. Furthermore, structural information contained in the artefacts is not considered. Two scenarios e.g., that use the same words but describe opposed procedures are considered to be equal by this approach. How these problems can be tackled is a matter of current research in IR (see e.g. [18]).

3.2 Case-Based Reasoning

Case-based Reasoning (*CBR*) is the process of solving new problems based on the solution of similar past problems [1, 3]. CBR uses a predefined structure of attributes as case representations, which describe problem and solution of past cases. This structure can be one simple table or a relational data model with several tables. CBR has been applied in different domains such as medical diagnosis, fault diagnosis of technical systems or software reuse.

CBR is able to compare all types of artefacts. However, the information contained in the artefacts needs to be reduced to the fixed structure of the case representations. Due to RSL's flexible structure of requirements specifications (defined in the RSL metamodel), loss of information can not be avoided.

In Case-based Reasoning two cases are considered to be equal if they share the same case representation. The above cited CBR applications each consider one specific domain while our requirements specifications potentially span a variety of different application domains. Thus, the ambiguity problem becomes more important. Basic CBR techniques use traditional IR measures for string comparison. These techniques do not address the paraphrase and ambiguity problem. However, more sophisticated methods include taxonomies to solve these problems [2].

3.3 Using Taxonomies

Semantic lexica and taxonomies can be used to determine the meaning of words used in an artefact. One example for a semantic lexicon is WordNet, which was developed at the Cognitive Science Laboratory at Princeton University [4]. It is based on the concept of synonym sets (called *synsets*) that group synonymic nouns, verbs, adjectives and adverbs and defines, among others, the semantic relations *hypernyms* / *hyponyms* (generalisation) and *holonym* / *meronym* (composition) between synsets. The latest version of WordNet (3.0) contains more than 150,000 words and almost 120,000 synsets, which together define more than 200,000 word-sense-pairs.

WordNet has been applied in combination with different approaches in order to improve retrieval results. A matter of current research in IR is the application of WordNet, or more generally semantic lexica, in order to solve the ambiguity and paraphrase problem [18, 13]. [6] used WordNet in combination with CBR approaches for retrieval of UML class diagrams.

Several similarity measures have been published based on WordNet. For an overview we refer the interested reader to [14]. These measures provide a similarity value for synset pairs, sometimes specifically for noun pairs or verb pairs. Such measures cannot compare sentences or whole paragraphs. Hence, they need to be integrated in other techniques. Most of these measures are based on pathlengths between synsets defined by semantic relations. Two synsets are considered similar when there is a short distance between them. Because the meaning of words is given the paraphrase and ambiguity problem do not emerge – both problems can be handled with this approach.

3.4 Structure-based Similarity

Structure-based similarity measures consider the structure of the artefacts to be compared. *Graph-based* and *Description Logics (DL)* [7] approaches measure similarity of two artefacts by comparing the vertices and arcs or concepts and roles, respectively. Both approaches focus on the artefact's structure and basically compare subgraphs using both taxonomic comparison of elements and their relations to other elements. In contrast to CBR, the structure-based similarity measures can also handle flexible, i.e. only partly known, structures.

Graph-based and Description Logics approaches both compute similarity values for all potential matches between elements within the compared artefacts. The graph-based SiDiff [10] algorithm, for example, additionally outputs a unified artefact that consists of all elements from both input artefatcs, based on the matches. Additional facilities of DL, like classification based on roles, nested logical operators like $\cup$, $\cap$, $\neg$, $\forall$, $\exists$, etc., are currently not considered necessary for RSL. Those facilities are used for in-depth description of terms with attributes and roles, which yields a typically expensive knowledge acquisition process.

Structure-based similarity measures are well suited for comparing artefacts with a flexible structure like the RSL requirements specifications. The measures are not suitable for unstructured artefacts like plain text documents. However, RSL specifications can contain such plain text elements.

When the structure of artefacts is known, not all elements are compared blindly, but only matching elements. For RSL's constrained language sentences this means that nouns are compared to nouns

only, verbs to verbs, etc. Additionally, not all nouns need to be compared to one another but subjects and objects can be distinguished. These approaches consider two artefacts equal when the same elements are represented with the same relations to other elements.

Finally, structure-based similarity measures can evaluate RSL hyperlinks from sentences to the domain vocabulary and global terminology in order to solve the ambiguity and paraphrase problem.

4 Concept for a combined similarity measure

The results described in the previous section show that no single similarity measure is suitable for all elements of RSL requirements specifications. Structure-based techniques are well-suited for model-based representations and constrained language sentences while they cannot handle plain text, like natural language hypertext sentences. On the other hand, Information Retrieval is well-suited for plain text but does not evaluate the structure of RSL elements and cannot solve the ambiguity and paraphrase problems. In the following we describe how we combine different similarity measures in order to compare a query with the requirements specification of software cases stored in the repository. We assume that the query is an initial or partial requirements specification, i.e. both the query and the former requirements specifications are specified in RSL.

We use a set of local similarity measures. The local similarity measures define how the similarity of particular RSL elements is computed. A global similarity measure combines the local similarity measures in order to compute the similarity of a query and the requirements specification of a software case. The global similarity measure computes the weighted normalised sum of the local similarities. The weighting takes into account the different sizes of compared elements. A large sentence list, for example, has a higher impact on the result than a single sentence.

In the following we exemplarily describe the local similarity measures for natural language hypertext sentence, constrained language sentence, sentence list and constrained language scenario, and show how they combine similarity measures from different research areas. For a detailed description that covers suitable similarity measures for all RSL elements we refer to [19].

When comparing two constrained language sentences (CL-Sentences) the structure of the sentences should be taken into account. Comparing the structure can be achieved, for example, by using a structure-based similarity measure. Due to the fact that all major elements of a constrained language sentence are linked with the global terminology, a WordNet-based measure is used to compare the single words. Similarity of sentence's structure and similarity of contained words are integrated into one similarity measure for constrained language sentences. For a combination of values from different measures a weighting of the values is needed. Reasonable values for such a weighting are to be determined in the upcoming experiments (see Section 5).

In contrast to constrained language sentences the natural language hypertext sentences (NLH-Sentence) are not highly structured. For their comparison only the text representation and the potentially contained links to the global terminology are evaluated. Information Retrieval approaches are used to compare the sentence's text and a WordNet-based similarity measure is applied for comparing linked words. This combined measure is used when comparing two natural language hypertext sentences but also when comparing a natural language hypertext sentence with a constrained language sentence. Figure 2 presents an overview of the local similarity measures.

Figure 3 illustrates the advantage of this combined measure for natural language hypertext sentences. The first requirements specification contains the sentence: *The system shall provide the* ***customer*** *the ability to* ***change*** *the* ***order*** *of* ***products***. and the second specification contains the sentence: *The system shall provide the* ***client*** *the ability to* ***modify*** *the* ***purchase order*** *of* ***merchandise***. A traditional IR similarity measure would produce a result reflecting that a significant amount of words is different in the two sentences. However, the words link to identical synsets (e.g. customer and client both link the synset *customer, client* in WordNet; see Figure 3), thus they are synonyms. The WordNet-based similarity measure evaluates the distance between synsets. Combining the IR measure with a WordNet-based measure results in a higher similarity value for the two sentences since the similarity of contained synonyms is taken into account. This reflects the intended meaning of the two sentences more precisely. In general the WordNet-based measure has the advantage of comparing the intended meaning and not only strings. However, it can only be used for the words that have been linked by the requirements engineer. Thus, it needs to be combined with another measure that takes into account the whole sentence text.

	CL Sentence	**NLH Sentence**
CL Sentence	Structure-based measure + WordNet-based measure	Information Retrieval + WordNet-based measre
NLH Sentence	Information Retrieval + WordNet-based measre	Information Retrieval + WordNet-based measre

Figure 2. Overview of the different local similarity measures for comparing constrained language sentences and natural language hypertext sentences.

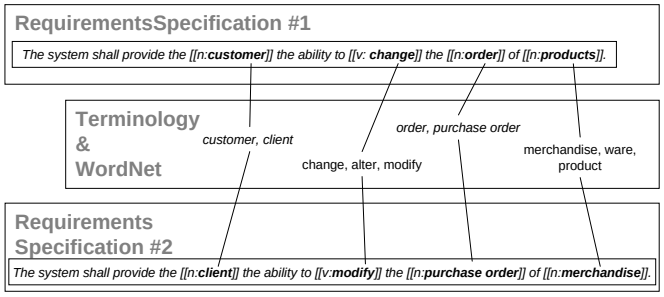

Figure 3. Two different natural language hypertext sentences linked to synsets in WordNet. Domain vocabularies are omitted from the figure for space reasons.

The comparison of sentence lists is based on the measures for sentences. The sentences of both lists are compared pairwise. For each sentence of the one sentence list that is contained in the query the maximum similarity to any sentence in the other sentence list identifies the best match. The similarity between the two sentence lists is the normalised sum of the maximum similarities. The same measure is used when comparing a sentence list with a constrained language scenario. But when two constrained language scenarios are compared to one another, the order of sentences is taken into account. Moreover, constrained language scenarios only contain the well-structured constrained language sentences. Thus, they are compared using a combination of Structure-based measure and WordNet-based measure. Figure 4 provides an overview of the measures required for comparing sentence lists and constrained language scenarios.

5 Summary and Discussion

In this paper we present a novel approach to solve the problem of comparing requirements specifications that use different representa-

	Sentence List	Constrained Language Scenario
Sentence List	Pairwise comparison of constrained language sentences	Pairwise comparison of constrained language sentences
Constrained Language Scenario	Pairwise comparison of constrained language sentences	Structure-based measure + WordNet-based measure

Figure 4. Overview of the similarity measures for comparing sentence lists and constrained language scenarios.

tions. The requirements specification language RSL allows to specify requirements using both model-based and descriptive elements. In order to compare requirements specifications written in RSL we assess similarity measures from different research areas according to their suitability and propose a combination of similarity measures for this task. The approach has following advantages:

- Reuse can be based on similarities of terms by using a global terminology for different software development projects.
- Syntactic-based reuse can be achieved by using Information Retrieval and Case-Based Reasoning methods.
- Semantic-based reuse can be achieved by using Structure-based methods.
- Expensive natural language processing is avoided by using a formal constrained language.
- Expensive knowledge acquisition, modelling, and representation is avoided by using direct links between the used terms and existing terminologies like WordNet.
- The ambiguity and paraphrase problems can be solved by combining similarity measures.

A topic of current research is the question whether two requirements with the same meaning but different representations should be treated equal or not. Apparently, two such requirements specify the same thing but we may want to consider their representation type because requirements engineers may like to work with specific requirement representations better than with other ones and thus react differently on the retrieval of different representation types.

A matter of evaluation is also in how far the ambiguity and paraphrase problems appear in real projects and how well our combined similarity measure solves them. It may happen that requirements engineers use only informal descriptive requirement representations without terminology hyperlinks, in which case IR techniques without terminology would not solve the two problems. First evaluations, which apply the RSL approach in industrial projects, show the usefullness of taxonomies.

Finally, fine tuning will be needed for the selected similarity measures. When numeric values are computed that denote similarity between two elements, then threshold values or weights play an important role. During the evaluation of the approach, our combined similarity measure will be iteratively improved.

The evaluation will be done using industrial application areas with a reasonable number of RSL requirement specifications and a tool that is currently developed within the ReDSeeDS project.

Acknowledgments

This research has been supported by the European Community under the grant IST-2006-33596 (ReDSeeDS). The project is coordinated by Infovide, Poland with technical lead of Warsaw University of Technology and with University of Koblenz-Landau, Vienna University of Technology, Fraunhofer IESE, University of Latvia, HITeC e.V. c/o University of Hamburg, Heriot-Watt University, PRO DV, Cybersoft and Algoritmu Sistemos.

REFERENCES

[1] Agnar Aamodt and Enric Plaza, 'Case-based reasoning: Foundational issues, methodological variations, and system approaches', *Artificial Intelligence Communications*, **7**(1), 39–59, (1994).

[2] Ralph Bergmann, 'On the use of taxonomies for representing case features and local similarity measures', in *6th German Workshop on CBR*, (1998).

[3] Ralph Bergmann, Sean Breen, Michel Manago, Stefan Wess, Mehmet Göker, Klaus-Dieter Althoff, and Ralph Traphöner, *Developing Industrial Case-Based Reasoning Applications - The INRECA Methodology*, volume 1612 of *Lecture Notes in Artificial Intelligence*, Springer, 1999.

[4] *WordNet: An Electronic Lexical Database*, ed., Christiane Fellbaum, MIT Press, 1998.

[5] N E Fuchs, U Schwertel, and R Schwitter, 'Attempto controlled english not just another logic specification language', *Lecture Notes in Computer Science*, **1559**, 1–20, (1999).

[6] Paulo Gomes, Francisco C. Pereira, Paulo Paiva, Nuno Seco, Paulo Carreiro, José L. Ferreira, and Carlos Bento, 'Using wordnet for case-based retrieval of uml models', *AI Commun.*, **17**, 13–23, (2004).

[7] Pedro A González-Calero, Mercedes Gómez-Albarran, and Belén Díaz-Agudo, 'Applying DLs for retrieval in case-based reasoning', in *Proc. of the 1999 International Workshop on Description Logics (DL'99)*, (1999).

[8] David A Grossman and Ophir Frieder, *Information retrieval: algorithms and heuristics*, Springer, 2004.

[9] Hermann Kaindl, Michał Śmiałek, Davor Svetinovic, Albert Ambroziewicz, Jacek Bojarski, Wiktor Nowakowski, Tomasz Straszak, Hannes Schwarz, Daniel Bildhauer, John P Brogan, Kizito Ssamula Mukasa, Katharina Wolter, and Thorsten Krebs, 'Requirements specification language definition', Project Deliverable D2.4.1, ReDSeeDS Project, (2007). www.redseeds.eu.

[10] Udo Kelter, Jürgen Wehren, and Jörg Niere, 'A generic difference algorithm for UML models', in *Proceedings of the SE 2005, Essen, Germany*, Essen, Germany, (March 2005).

[11] Thorsten Krebs, Wiktor Nowakowski, Audris Kalnins, and Elina Kalnina, 'Modelling and transformation language validation report', Project Deliverable D3.4, ReDSeeDS Project, (2007). www.redseeds.eu.

[12] Mario Lenz, Brigitte Bartsch-Spörl, Hans-Dieter Burkhard, and Stefan Wess, eds., *Case-Based Reasoning Technology - From Foundations to Applications*, chapter Textual CBR, 115–137, Springer, 1998.

[13] Shuang Liu, Clement Yu, and Weiyi Meng, 'Word sense disambiguation in queries', in *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pp. 525–532, New York, NY, USA, (2005). ACM Press.

[14] T. Pedersen, S. Patwardhan, and J. Michelizzi, 'WordNet::similarity - measuring the relatedness of concepts', in *Proc. 19th National Conference on Artificial Intelligence (AAAI-04)*, p. 3p, (2004).

[15] Michał Śmiałek, 'Mechanisms for requirements based model reuse', in *International Workshop on Model Reuse Strategies - MoRSe*, pp. 17–20, (2006).

[16] Michał Śmiałek, Albert Ambroziewicz, Jacek Bojarski, Wiktor Nowakowski, and Tomasz Straszak, 'Introducing a unified requirements specification language', in *Proc. CEE-SET'2007, Software Engineering in Progress*, eds., Lech Madeyski, Miroslaw Ochodek, Dawid Weiss, and Jaroslav Zendulka, pp. 172–183. Nakom, (2007).

[17] Michał Śmiałek, Jacek Bojarski, Wiktor Nowakowski, and Tomasz Straszak, 'Writing coherent user stories with tool support', *Lecture Notes in Computer Science*, **3556**, 247–250, (2005).

[18] Giannis Varelas, Epimenidis Voutsakis, Paraskevi Raftopoulou, Euripides G M Petrakis, and Evangelos E Milios, 'Semantic similarity methods in WordNet and their application to information retrieval on the web', in *WIDM '05: Proceedings of the 7th annual ACM international workshop on Web information and data management*, pp. 10–16, New York, NY, USA, (2005). ACM Press.

[19] Katharina Wolter, Thorsten Krebs, Daniel Bildhauer, Markus Nick, and Lothar Hotz, 'Software case similarity measure', Project Deliverable D4.2, ReDSeeDS Project, (2007). www.redseeds.eu.

[20] Katharina Wolter, Thorsten Krebs, and Lothar Hotz, 'Determining similarity of model-based and descriptive requirements by combining different similarity measures', in *2nd International Workshop on Model Reuse Strategies (MoRSe 2008)*, Beijing, China, (2008). Fraunhofer IRB Verlag.