Using Semantic Relatedness and Locality for Requirements Elicitation Guidance

Stefan Farfeleder
Institute of Computer Languages
Vienna University of Technology
Vienna, Austria
stefan.farfeleder@tuwien.ac.at

Thomas Moser
CDL Flex
Vienna University of Technology
Vienna, Austria
thomas.moser@tuwien.ac.at

Andreas Krall
Institute of Computer Languages
Vienna University of Technology
Vienna, Austria
andi@complang.tuwien.ac.at

***Abstract*—Requirements engineers strive for high-quality requirements which are the basis for successful projects. Ontologies and semantic technologies have been used successfully in several areas of requirements engineering, e.g., analysis and categorization of requirements. We improve a semantic guidance system which derives phrases from a domain ontology by taking two observations into account: a) domain terms that are semantically related are more likely to be used together in a requirement, and b) the occurrence of domain terms is usually not distributed uniformly over the requirements specification. We define suggestion orders that make use of these properties which results in automatically proposing semantically and spatially related domain terms to the requirements engineer. We implement these orders in our tool and provide an evaluation using three projects from the embedded systems domain. We achieve significant suggestion quality improvements over previous work.**

***Keywords*—domain ontology, semantic relatedness, requirements specification, requirements elicitation, guidance**

I. Introduction

Any specification errors that propagate from the requirements definition phase into later development phases such as design or testing have a much higher impact and cost to fix them. Therefore requirements engineers work hard to find errors in requirements and to have complete, correct and consistent requirements.

Requirements specified using natural language text have inherent ambiguities due to different ways to interpret them by stakeholders. There have been many strategies to reduce the ambiguity in requirement statements. They include restricting the allowed grammar to a subset, e.g., Attempto Controlled English [1], and using predefined vocabularies, e.g., the Language Extended Lexicon [2].

Ontologies have been used for requirements engineering in several ways. Körner and Brumm [3] use domain-independent ontologies to detect linguistic problems in specifications. Other works capture knowledge about the problem domain in the ontology [4][5]. There the ontology acts as a vocabulary of domain terms with additional links between the terms that define their relationships. This enables the analysis of requirement statements with regards to the domain knowledge represented in the ontology, e.g., to find missing requirements. The support for inferring knowledge and reasoning allows automating tasks that otherwise would have to be done manually.

Additionally to the analysis of already specified requirements, [6] showed that a domain ontology can also be used as a guide during the specification of new requirements. By directly using ontology information during requirements definition this combines the advantages of ontology-based analysis techniques with a reduction of specification effort. In this work we build upon this guidance system and add a method that actively tries to estimate what the requirements engineer intends to type by taking semantic relations and nearby requirements into account.

This work is structured as follows. Section II discusses related work; section III motivates our research and presents the research questions. Then our proposed approach is described in section IV, and its evaluation follows in section V. Finally section VI concludes and lists future research topics.

II. Related Work

PROPEL [7] is a tool that provides guidance for defining property specifications which are expressed as finite-state automata. For the definition of a property the user is guided by a *question tree*, a hierarchical sequence of questions. There are separate question trees for a property's behavior and its scope. Based on the answers the tool chooses an appropriate property template. The tool is used to elicitate requirements in the medical domain.

Kitamura et al. [4] present a requirements elicitation tool that improves requirements quality by ontology-based analysis. The tool analyzes natural language requirements and maps words to domain ontology concepts. According to these occurrences and their relations in the ontology, requirements are analyzed in terms of completeness, correctness, consistency and unambiguity. The authors argue that with the help of the domain ontology the requirements engineer needs less knowledge about the problem domain. Hints are provided to the requirements engineer to help solving found problems, e.g., if the tool determines that a domain concept is not defined in the requirements set, the tool suggests adding a requirement about the missing concept.

Recommendation systems filter items from a large set of data and recommend them to a user based on information about the user's preferences. Maalej and Thurimella [8] provide an overview of possible applications of recommendation systems to requirements engineering. Additionally to the requirement statements themselves and vocabularies - both related to our work - they foresee usage in the area of recommending quality measures, requirements dependencies, people, etc.

SRRS [9] is a recommendation system which supports choosing a requirements elicitation method for security requirements. The requirements engineer needs to assign priorities to ten key characteristics (e.g., unambiguity, traceability, scalability) according to which the system then suggests the most appropriate method.

Chen et al. [10] use a concept hierarchy which is derived from a web directory to improve the quality of an advertising keyword suggestion system. Concepts are considered to be important in a hierarchy node if they do not occur in other parts of the hierarchy as well. Semantically similar concepts are grouped into clusters and one representative concept per cluster is presented to the user. The semantically enriched system performs better than traditional systems that are based on statistical co-occurrence.

Literature distinguishes between semantic *relatedness* and semantic *similarity*. Similarity is a specific kind of relatedness. Reusing an example from Resnik [11], a *car* is *similar* to a *bicycle* - both are a kind of vehicle - but the *car* is *related* to *gasoline*, because it *requires* gasoline to drive. Semantic similarity only considers subclass-of relations between concepts while semantic relatedness takes all kinds of relations into account, e.g., meronymy (part-of) and functional relationships (*require* in the example above).

Budanitsky and Hirst [12] compare five measures of lexical semantic relatedness and similarity that are based upon WordNet. WordNet contains synonymy, hyponymy (is-a, subclass-of), several kinds of meronymy (part-of) and antonymy (opposite-of) relations. The measures are compared to human judgment and in the context of an application to detect malapropism, the usage of lexically similar but semantically incorrect words. Unfortunately four out of five measures only handle similarity.

Yeh et al. [13] use the Wikipedia data set to compute semantic relatedness between words. They build a graph from the Wikipedia data, the articles being the nodes and links to other articles being the edges. They differentiate between links in infoboxes, categorical links and links in the content itself. Combining the *Personalized PageRank* and the *Explicit Semantic Analysis* techniques they achieve good results on two established test data sets.

III. Research Issues

Previous work [6] presented a semantic guidance system for requirements elicitation. It proposes phrases derived from the knowledge contained in a domain ontology. While the system works well for small ontologies, we found practical issues when applying it to more requirements and bigger ontologies due to the large number of generated proposals:

- The requirements engineer needs to type more characters until filtering limits the proposals to a manageable amount.
- A requirements engineer not familiar with the domain does not benefit from seeing a very long list of proposals, e.g., when a domain term slipped his mind.

Clearly there must be a better method than showing all matching proposals in alphabetical order. During the study of these problems we thought about what constitutes "good" proposals. We made the following observations:

- While writing a requirement statement it is possible to use already specified requirement parts to, combined with the ontology knowledge, predict the remainder of the statement, at least to a certain degree. Consider a requirement starting with "The Safing Controller shall be able to". We can support the requirements engineer by suggesting concepts that are semantically related to the concept *Safing Controller* for the following parts of the requirement, e.g., *SafeAct mode* in Fig. 1.
  We identified the following common types of semantic relations in requirements (this list is intended to be exemplary and not exhaustive):
  - Function: Requirements of the form "*subject* shall [be able to] *verb object*". Given an ontology link between *subject* and *object*, we can suggest *object* if the requirements already contains *subject*.
  - Restriction: Requirements of the form "if *event* then *subject* shall [...]" or "during *state subject* shall [...]". Given an ontology link between a concept in *event/state* and *subject*, we can suggest *subject* if the requirements already contains *event/state*.
  - Architecture: Requirements of the form "*system* shall have *subsystem*". Given an ontology link between *system* and *subsystem*, we can suggest *subsystem* if the requirements already contains *system*.

  In this work we assume having a suitable domain ontology containing such links. Ontology extraction techniques (e.g., [5]) can be used to extract domain ontologies from text documents.
  We call this observation the *semantic relatedness property*.
- The occurrences of many domain terms used in a requirements specification are not randomly distributed over the entire set of requirements but are clustered around a certain location in the document. This is only natural considering that most requirements specifications are organized into chapters each describing an individual part of the system. We call this observation the *locality property*.
  As a quick check for this observation we measured the distributions of the requirement indices for concepts that occur at least twice and computed the standard deviations. For example, a concept occurring in two subsequent requirements has a standard deviation of 0.5, one occurring in requirement 1 and in requirement 100 a standard deviation of 49.5. On average, the standard deviations of those distributions were 11.60, 35.03 and 9.87 for the

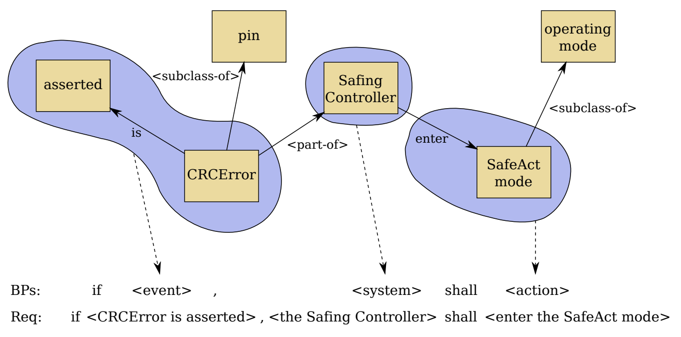

pin
operating mode
asserted
<subclass-of>
Safing Controller
<subclass-of>
is
<part-of>
enter
CRCError
SafeAct mode
BPs: if <event> , <system> shall <action>
Req: if <CRCError is asserted> , <the Safing Controller> shall <enter the SafeAct mode>

Fig. 1. Suggestions derived from Domain Ontology, Boilerplates and Requirement

three requirements sets used in our evaluation. This is considerably lower than for randomly distributed indices (28.58, 81.69 and 24.82).

We call two concepts that occur in nearby requirements *spatially related*.

The idea of this research is to find out whether we can make use of those properties and whether this actually improves the guidance system. In particular we want to answer the following research questions:

- *RQ1*: Does the guidance system improve if we preferably suggest semantically related concepts?
- *RQ2*: Does the guidance system improve if we preferably suggest spatially related concepts?

We are optimistic that both questions can be answered positively and have the following hypotheses:

- *H1*: Suggesting semantically related concepts improves the guidance system.
- *H2*: Suggesting spatially related concepts improves the guidance system.

IV. Guidance System

This section briefly summarizes the existing semantic guidance system and then goes on to introduce the new contributions, starting with section IV-D.

The goal of the guidance system is assisting the requirements engineer with specifying requirements. It does that by proposing textual phrases which are derived from the ontology information. Fig. 1 depicts a part of the ontology and how the guidance is related. The beige boxes at the top and the arrows between them are the ontology information while the blue areas represent the derived phrases (“CRCError is asserted”, “the Safing Controller” and “enter the SafeAct mode”). Axioms are labelled with brackets to make the distinguishable from named relations. A requirement using these suggestions can be seen at the very bottom.

A. Boilerplate Requirements

Our approach uses *boilerplates* for requirement statements. This term was coined by J. Dick [14] and refers to a textual requirement template. A boilerplate consists of a sequence of attributes and fixed syntax elements. A common boilerplate is “⟨*system*⟩ `shall` ⟨*action*⟩”. In this boilerplate ⟨*system*⟩ and ⟨*action*⟩ are attributes and `shall` is a fixed syntax element. It is possible to combine several boilerplates by means of concatenation (in Fig. 1 the boilerplates “`if` ⟨*event*⟩`,`” and “⟨*system*⟩ `shall` ⟨*action*⟩” are combined). This allows keeping the number of required boilerplates low while at the same time having a high flexibility. During instantiation textual values are assigned to the attributes of the boilerplates; a boilerplate requirement is thus defined by its boilerplates and its attribute values.

We use boilerplates for our approach due to two reasons: a) the usage of different attributes allows providing context-sensitive guidance (section IV-C), i.e., proposing different suggestions depending on the current attribute, and b) using boilerplates helps specifying requirements that are syntactically uniform when using a small number of templates (compared to the number of requirements).

There are numerous other template-based approaches for requirements specification, most of them being more formal than boilerplates. Post et al. [15] report on successfully applying a formal specification pattern system defined by Konrad and Cheng [16] on automotive requirements. However, the authors limit themselves to a certain class of requirements, the behavioral requirements. Our approach aims at covering all kinds of textual requirements.

B. Domain Ontology

A nowadays broadly accepted definition of the term *ontology* is that it is a *formal, explicit specification of a shared conceptualization* [17]. A domain ontology focuses on a specific subject, here the system under construction. For requirements engineering the aspect of sharing is of particular interest, it means that stakeholders agree on a terminology and the relations between the terms. By making use of this information we lower the risk for requirements ambiguity.

The following ontology entities are used for the guidance system:

- Concepts: Ontology concepts are the terms the requirements engineer uses in the requirements. This includes actors, components, events, states, etc. of the system under construction. In Fig. 1 the concepts are “asserted”, “CRCError”, “pin”, “Safing Controller”, “SafeAct mode” and “operation mode”.
- Relations: Relations are links between concepts. We use two kinds of relations:
  - A named relation represents a functional relationship between a subject concept and an object concept. The relation name is expected to be a transitive verb. Named relations are used to derive suggestions.
  - Anonymous relations simply indicate that two concepts are related. This is used to prefer semantically related suggestions.

  In Fig. 1 there are two named relations: “is” between concepts “CRCError” and “asserted”, and “enter” between “Safing Controller” and “SafeAct mode”.

- Axioms: Axioms are relations between concepts with a special meaning. An *equivalence* axiom represents the knowledge that two concepts refer to the same phenomenon in the domain (synonyms). A *subclass-of* axiom states that one concept is a subclass of another one. Both kinds of axioms lead to an inheritance of relations from the equivalent or parent concept. The *part-of* axiom imposes a hierarchical structure on the ontology contents which facilitates navigation.

Additionally the ontology contains links that classify concepts into one of the boilerplate attributes (the link between "Safing Controller" and ⟨*system*⟩ in Fig. 1).

C. Suggestions

From the ontology knowledge the guidance system infers three kinds of suggestions:

- Concept: The simplest kind of suggestion simply consists of the name of an ontology concept, optionally prefixed with the determiner "the".
- Verb-Object: From a named ontology relation our system proposes the relation's verb in infinitive form followed by the name of the relation's destination concept (again optionally prefixed with "the"). This suggestion is intended to follow "shall" or "shall be able to" formulations often found in requirements.
- Subject-Verb-Object: From a named ontology relation our system proposes the relation's source concept, followed by the relation's verb in third person singular form, and the relation's destination concept - both concept names optionally prefixed with "the". This suggestion is intended to be used in clauses starting with "if", "while" or similar words.

In Fig. 1 an example is provided for each suggestion kind: "the Safing Controller" for Concept, "enter the SafeAct mode" for Verb-Object and "CRCError is asserted" for Subject-Verb-Object. It is not always grammatically correct to add a determiner before a concept name, e.g., before "asserted". Thus our approach checks the part-of-speech (a classification of a word into one of several types, e.g., noun, verb, adjective) to determine this.

The three suggestion kinds are used for different boilerplate attributes. Table I shows the mapping between suggestions and attributes. For the attributes *event* and *state* both Concept suggestions and Subject-Verb-Object are used due to grammatical reasons, e.g., a noun should be used for `during` ⟨*state*⟩ but `if` ⟨*state*⟩ requires a clause. The Verb-Object suggestions are used for the ⟨*action*⟩ attribute which generally follows modal verbs like "shall"; the remaining boilerplate attributes use only the Concept suggestions.

D. Semantic Relatedness of Suggestions

We mentioned functional, restrictional and architectural relations in requirements earlier. These relations have in common that they do not correspond to simple is-a ontology links. While there exist requirements where is-a links are useful, e.g., *"The system shall have the following states: … "*, they

TABLE I
ATTRIBUTE SUGGESTION MAPPING

Attribute	Con	VO	SVO
⟨*system*⟩, etc.	✓		
⟨*action*⟩		✓	
⟨*event*⟩, ⟨*state*⟩	✓		✓

TABLE II
PROJECT MEASURES

Item	DMS	Airbag	Power-train
Reqs.	99	283	86
Concepts	233	591	267
Relations	164	425	115
Axioms	100	571	163

are rare. From this point of view semantic relatedness is more important to us than semantic similarity. Using the car example mentioned earlier, a possible requirement is *"The driver shall be able to refuel the car with gasoline."* When proposing phrases to requirements engineers, we want to make use of semantic relatedness between concepts, e.g., given a requirement that already contains *car* we would rather suggest *gasoline* than *bicycle*.

Unfortunately from the review of related literature it seems that researchers concentrate more on similarity than on relatedness. We were able to find complex functions to compute semantic similarity using up to six different factors [18]: link type, node depth, local density, link strength, node attributes and cluster granularity degree. Most of them only make sense in an is-a taxonomy and do not apply well to the more generic graph we are using. We experimented with using different weights depending on link types but that had no measurable effect. In the end we decided to go for simplicity: We measure the relatedness of two concepts simply by using the edge count of the shortest path between two concepts.

We define C to be the set of concepts, T to be the set of link types in the ontology, $L \subseteq C \times C \times T$ to be the set of all ontology links and the direct distance δ_l between two concepts s and d to be

$$\delta_l(c, c') = \begin{cases} 1 & \text{if } \exists\, t \in T : \langle c, c', t\rangle \in L \vee \langle c', c, t\rangle \in L, \\ \infty & \text{otherwise.} \end{cases}$$

We use ontology links in an undirected manner to be more flexible with the order in which concepts are stated in a requirement, e.g., we can handle "*subject* shall *action* if *event*" even though the link might be directed from *event* to *subject*. Moreover we do not require instantiating boilerplate attributes from left to right. Using δ_l we can compute the shortest path δ_p between two concepts. In Fig. 1 the distance between *Safing Controller* and *SafeAct mode* is one, the distance between *operating mode* and *assert* is four.

We define R to be the set of requirements and S to be the set of suggestions. Further we define $cr : R \to \mathcal{P}(C)$ and $cs : S \to \mathcal{P}(C)$ to map a requirement or, respectively, a suggestion to the concepts it uses. Next we define the distance δ_s between a requirement r and a suggestion s to be

$$\delta_s(r, s) = \max(\min_{c \in cr(r)} \min_{c' \in cs(s)} \delta_p(c, c'), 1).$$

It is the minimum distance between any concept used in the requirement and any concept used in the suggestion. We cap it at one to avoid first proposing suggestions containing concepts already used in the current requirement.

E. Locality Property

Requirements documents are often structured into sections each covering a different aspect of system functionality. Terms specific to that functionality occur mostly there. In order to measure the spatial relatedness of two requirements, we consider the requirements to be a list and define an index function $ind : R \to \mathbb{N}$. Requirements that have similar indices are spatially related. When specifying a new requirement, the requirements engineer must indicate the index where the new requirement should be inserted into the list. This is similar to choosing a section where the requirement should be added to.

We define the locality factor loc of a requirement r and a suggestion s in the following way:

$$loc(r, s) = \min_{r' \in R} \begin{cases} |ind(r) - ind(r')| & \text{if } cr(r') \cap cs(s) \neq \emptyset, \\ \infty & \text{otherwise.} \end{cases}$$

It is the index distance to the nearest requirement that shares at least one concept with the suggestion.

F. Suggestion Algorithm

Based on the previous observations we define four different suggestion rankings. The first one (alpha) sorts suggestions alphabetically and serves as a baseline. The second (sem-rel) and third one (locality) sort using the semantic relatedness function δ_s and locality function loc, respectively. The fourth order (ML) uses a linear combination of both functions. A machine learning approach is used to obtain good coefficients for the combination.

Our algorithm for suggestions in a requirement r given a (possibly empty) word w the requirements engineer is currently typing for boilerplate attribute a consists of the following steps:

1) Compute all suggestions and filter out those not starting with w.
2) Sort remaining suggestions by either
    a) Matching boilerplate attribute a (suggestions marked by "✓" in Table I are ranked before suggestions lacking one), and then alphabetically, using δ_s or loc, or
    b) Using the weight computed by ML coefficients

Using these orders we achieve that semantically and spatially related suggestions are ranked and shown before other suggestions.

For the ML ranking we use linear regression models, one for each boilerplate attribute. The models are trained using the following features: suggestion type, matching boilerplate attribute, value of δ_s and loc and whether a suggestion is correct (prediction value 1.0) or not (0.0) in the context of the other features. A higher value computed by such a model is considered to be a better match. We use linear regression models because they are very efficient at computing the weights of all suggestions.

V. Evaluation

To test hypotheses H1 and H2 we implemented the enhancements in our tool DODT (Domain Ontology Design Tool). We set up an evaluation with three different industrial projects: DMS (Doors Management System) which controls aircraft doors, an airbag controller that decides if an airbag should be deployed and a power-switch used in the powertrain domain. For each project a domain ontology was extracted from domain documents using term and relation extraction techniques and some manual pruning and refinement. Table II gives information about the requirements and the domain ontologies of the projects.

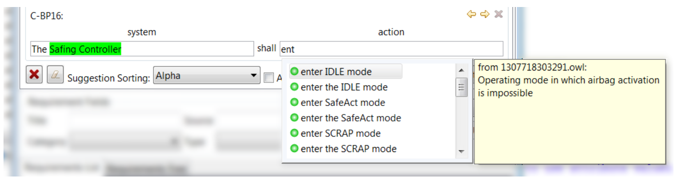
We took the following approach to measure the difference between the baseline suggestion order (alpha) and the orders evolving from the research questions RQ1 (sem-rel), RQ2 (locality) and the ML order. First we looked for all places where a requirement phrase p (one or more words of the requirement) matched one of our suggestions. In case several suggestions matched at the same position, we used the longest one. Then we computed the suggestion list our algorithm yielded and measured at which position in this list the expected phrase p was ranked. This we repeated for all substrings of p of length zero to five and for all suggestion orders. The requirements were added to our guidance system in the same order that is used in the projects. We argue the average index is a good measure for the performance of a suggestion order as the user sees the first N entries (depending on the window size) and has to scroll down for further entries. Fig. 2 shows a screenshot of our tool presenting suggestions. To avoid overfitting to the evaluation data we used a three-fold cross-validation for the ML order, i.e., for each project we trained the model using data from the other two projects.

Table III presents our results. It shows the arithmetic means and standard deviations for all suggestion orders, project and lengths. For example: If an Airbag project user enters the first two letters of a phrase, the expected suggestion will be on average at index 5.94 for the alpha order and at index 1.91 for the ML order. In Fig. 2 the index of the suggestion "enter SafeAct mode" is two.

All three suggestion orders using semantic relatedness and/or locality are an improvement over the baseline order with regards to the index of the correct suggestion when comparing means. The ML order generally performs best except for DMS at lengths 4 and 5 where sem-rel performs better. We tested statistical significance using a Student's t-test with a confidence level of 95% (rejecting the null hypothesis of identical distributions). The improvements are significant for ML (DMS: len$\leq$4, Airbag: all, Powertrain: all), for sem-rel (len$\leq$3, all, len$\leq$2) and for locality (len$\leq$1, all, len$\leq$2). For longer lengths the improvements are not significant. With these restrictions in mind, we claim our hypotheses to be true. We achieve the biggest improvements at length zero (start of a new phrase) because here the alpha suggestion order is basically random while our proposed orders rank related suggestions first. At longer phrases this advantage decreases

TABLE III
SUGGESTION RESULTS

	alpha		sem-rel		locality		ML	
len	avg	σ	avg	σ	avg	σ	avg	σ
				DMS				
0	134.9	135.1	81.43	101.4	102.7	124.9	69.39	103.9
1	6.99	14.32	4.02	9.09	5.50	14.09	3.69	7.22
2	1.66	3.00	1.03	2.56	1.34	2.92	0.89	1.95
3	0.98	2.44	0.70	2.29	0.85	2.35	0.61	1.26
4	0.68	1.39	0.48	1.19	0.61	1.32	0.56	1.13
5	0.57	1.33	0.43	1.18	0.53	1.28	0.50	1.16
				Airbag				
0	353.0	332.7	161.9	228.9	204.7	264.0	100.1	196.5
1	29.00	67.16	13.74	31.79	18.54	55.27	8.53	31.40
2	5.94	28.01	2.77	8.06	3.56	16.76	1.89	7.35
3	3.15	5.40	1.81	4.02	1.79	3.77	1.15	2.96
4	1.18	1.98	0.90	1.79	0.88	1.72	0.61	1.42
5	0.64	1.42	0.49	1.28	0.50	1.23	0.33	0.89
				Powertrain				
0	224.9	200.7	154.4	175.2	153.8	176.1	107.5	154.8
1	28.95	56.13	19.50	48.31	15.55	38.69	9.42	28.99
2	2.80	6.02	2.03	5.18	2.36	5.48	1.15	2.86
3	1.95	5.27	1.40	4.68	1.65	4.84	0.84	2.27
4	1.74	5.30	1.19	4.62	1.45	4.80	0.86	2.39
5	1.67	5.37	1.15	4.68	1.37	4.81	0.83	2.36


C-BP16:

system action

The Safing Controller shall ent

Suggestion Sorting: Alpha

- enter IDLE mode
- enter the IDLE mode
- enter SafeAct mode
- enter the SafeAct mode
- enter SCRAP mode
- enter the SCRAP mode

from 1307718303291.owl:
Operating mode in which airbag activation is impossible

Fig. 2. DODT Screenshot showing Suggestions

because filtering often reduces the suggestions to a small set of very similar ones which is more difficult to rank in a good way, especially if there is a long common prefix.

A. Threats to Validity

The quality of the suggestion order depends on the ontology quality, especially on the links for the semantic relatedness. The process to create the domain ontology involved a selection and validation by a domain expert.

VI. CONCLUSION AND FUTURE WORK

In this paper we presented two enhancements to our semantic guidance system. The first one is taking the semantic relatedness between concepts in a requirement statement into account. The probability that we want to use concepts that are semantically related to concepts already occurring in the requirement is increased. The second enhancement is taking the locality of domain terms with regards to their occurrence location in the requirements document into account. For our guidance system this means that the probability for reuse of spatially related concepts is increased. We defined orders on suggestions to exploit these properties and implemented them in our tool. An evaluation using three sets of industrial requirements showed that both properties lead to an improvement of the suggestion quality.

More research needs to be done on deriving phrases from ontology concepts and relations. There are grammatical forms in our requirements, e.g., gerunds, which could be supported additionally. However, this needs to be evaluated carefully as there is a trade-off between convenience for the requirements engineer and achieving requirements consistency by using the same phrases in all requirements.

Finally we plan to research ontology extraction methods that are specialized to extract exactly the kind of concepts and relations we are interested in.

ACKNOWLEDGMENTS

The research leading to these results has received funding from the ARTEMIS Joint Undertaking under grant agreement N° 100016 and from specific national programs and/or funding authorities.

REFERENCES

[1] N. Fuchs, U. Schwertel, and R. Schwitter, "Attempto Controlled English - Not Just Another Logic Specification Language," in *LOPSTR 1999*. Springer, 1999, pp. 1–20.
[2] J. Leite and A. Franco, "A Strategy for Conceptual Model Acquisition," in *Requirements Engineering, 1993., Proceedings of IEEE International Symposium on*. IEEE, 1993, pp. 243–246.
[3] S. Körner and T. Brumm, "Natural Language Specification Improvement with Ontologies," *International Journal of Semantic Computing (IJSC)*, vol. 3, no. 4, pp. 445–470, 2010.
[4] M. Kitamura, R. Hasegawa, H. Kaiya, and M. Saeki, "A Supporting Tool for Requirements Elicitation Using a Domain Ontology," in *ICSOFT 2009*. Springer, 2009, pp. 128–140.
[5] I. Omoronyia, G. Sindre, T. Stålhane, S. Biffl, T. Moser, and W. Sunindyo, "A Domain Ontology Building Process for Guiding Requirements Elicitation," in *REFSQ 2010*. Springer, 2010, pp. 188–202.
[6] S. Farfeleder, T. Moser, A. Krall, T. Stålhane, I. Omoronyia, and H. Zojer, "Ontology-Driven Guidance for Requirements Elicitation," in *ESWC 2011*. Springer, 2011, pp. 212–226.
[7] R. Cobleigh, G. Avrunin, and L. Clarke, "User Guidance for Creating Precise and Accessible Property Specifications," in *14th International Symposium on Foundations of Software Engineering*. ACM, 2006, pp. 208–218.
[8] W. Maalej and A. Thurimella, "Towards a Research Agenda for Recommendation Systems in Requirements Engineering," in *2009 Second International Workshop on Managing Requirements Knowledge*. IEEE Computer Society, 2009, pp. 32–39.
[9] J. Romero-Mariona, H. Ziv, and D. Richardson, "SRRS: A Recommendation System for Security Requirements," in *Proceedings of the 2008 International Workshop on Recommendation Systems for Software Engineering*. ACM, 2008, pp. 50–52.
[10] Y. Chen, G. Xue, and Y. Yu, "Advertising Keyword Suggestion Based on Concept Hierarchy," in *Proceedings of the International Conference on Web Search and Web Data Mining*. ACM, 2008, pp. 251–260.
[11] P. Resnik, "Using Information Content to Evaluate Semantic Similarity in a Taxonomy," in *Proceedings of the 14th International Joint Conference on Artificial Intelligence*. Morgan Kaufmann Publishers Inc., 1995, pp. 448–453.
[12] A. Budanitsky and G. Hirst, "Evaluating WordNet-based Measures of Lexical Semantic Relatedness," *Computational Linguistics*, vol. 32, no. 1, pp. 13–47, 2006.
[13] E. Yeh, D. Ramage, C. Manning, E. Agirre, and A. Soroa, "Wikiwalk: random walks on wikipedia for semantic relatedness," in *Proceedings of the 2009 Workshop on Graph-based Methods for Natural Language Processing*. Association for Computational Linguistics, 2009, pp. 41–49.
[14] E. Hull, K. Jackson, and J. Dick, *Requirements Engineering*. Springer, 2005.
[15] A. Post, I. Menzel, and A. Podelski, "Applying Restricted English Grammar on Automotive Requirements - Does it Work? A Case Study," in *REFSQ 2011*. Springer, 2011, pp. 166–180.
[16] S. Konrad and B. Cheng, "Real-time Specification Patterns," in *Proceedings of the 27th International Conference on Software Engineering*. ACM, 2005, pp. 372–381.

[17] R. Studer, V. Benjamins, and D. Fensel, “Knowledge Engineering: Principles and Methods,” *Data & knowledge engineering*, vol. 25, no. 1-2, pp. 161–197, 1998.

[18] X. Xu, J. Huang, J. Wan, and C. Jiang, “A Method for Measuring Semantic Similarity of Concepts in the Same Ontology,” in *2008 International Multi-symposiums on Computer and Computational Sciences*. IEEE, 2008, pp. 207–213.