NARCIA: An Automated Tool for Change Impact Analysis in Natural Language Requirements

Chetan Arora, Mehrdad Sabetzadeh,
Arda Goknil, Lionel C. Briand
SnT Centre for Security, Reliability and Trust
University of Luxembourg, Luxembourg
{firstname.lastname}@uni.lu

Frank Zimmer
SES TechCom
9 rue Pierre Werner
Betzdorf, Luxembourg
{firstname.lastname}@ses.com

ABSTRACT

We present NARCIA, a tool for analyzing the impact of change in natural language requirements. For a given change in a requirements document, NARCIA calculates quantitative scores suggesting how likely each requirements statement in the document is to be impacted. These scores, computed using Natural Language Processing (NLP), are used for sorting the requirements statements, enabling the user to focus on statements that are most likely to be impacted. To increase the accuracy of change impact analysis, NARCIA provides a mechanism for making explicit the rationale behind changes. NARCIA has been empirically evaluated on two industrial case studies. The results of this evaluation are briefly highlighted.

Categories and Subject Descriptors

D.2.1 [**Software Engineering**]: Requirements/Changes

Keywords

Natural Language Requirements, Natural Language Processing (NLP), Change Impact Analysis.

1. INTRODUCTION

Requirements changes, if left unmanaged, are a major source of software project failures [7]. A key prerequisite for managing change in requirements is the ability to identify which requirements are impacted as the result of a specific change. A manual identification of impacted requirements is laborious and presents a challenge when analysts are faced with large, rapidly-changing requirements documents.

We present a tool, named NARCIA (NAtural language Requirements Change Impact Analyzer), to support automated change impact analysis in Natural Language (NL) requirements. Our focus on NL requirements is motivated by the prevalent use of these requirements in industry.

An important characteristic of NARCIA is that it does not need the relationships between requirements to be specified a priori. For NL requirements, a precise specification of the relationships often entails a significant level of manual effort [6]. NARCIA instead uses Natural Language Processing (NLP) to automatically detect potential relationships between requirements. The main observation that makes such automation possible is that a majority of requirements relationships manifest themselves within the phrases that constitute the requirements statements. NARCIA exploits the phrasal structure of the requirements statements for identifying the relationships between different requirements.

ESEC/FSE'15, August 30 – September 4, 2015, Bergamo, Italy
ACM. 978-1-4503-3675-8/15/08...$15.00
http://dx.doi.org/10.1145/2786805.2803185

R1: The nuclear power plant shall have ~~six~~ five successive degrees of protection for safety functions in the event of a nuclear accident alarm.

R2: The sixth degree of defence shall ensure the leak-prevention of the ducts and raise a nuclear ~~accident alarm~~ hazard warning if an anomaly is detected.

R3: The remaining degrees of protection shall be designated to signal the nuclear power plant operators about a nuclear accident.

R4: The monitoring system for leak-prevention of the ducts shall periodically monitor the amount of radioactive materials in the exhaust air.

Figure 1: Example requirements and changes.

To illustrate the role of phrases, consider $R1$ in the example requirements of Figure 1. The syntactic change in $R1$, namely replacing six with five, does not convey any information about the semantic context in which the change occurred. To analyze the impact of this change in a meaningful manner, one has to consider the phrase(s) that have been affected. At the phrase level, the change in $R1$ is interpreted as six successive degrees of protection being replaced by five successive degrees of protection, rather than six being replaced by five. The ability to delineate this semantic context is key to identifying related requirements that are likely to be impacted by the change.

A second important characteristic of NARCIA is that it provides explicit means for expressing change rationale. To illustrate why capturing the change rationale is important, consider $R2$ in the example requirements of Figure 1. Here, the phrase a nuclear accident alarm has changed to a nuclear hazard warning. Depending on what the rationale for the change is, the change may propagate differently to other requirements. For example, if the rationale is to rename nuclear accident alarm to nuclear hazard warning, then $R1$ and $R3$ will be affected. However, if nuclear hazard warning is a new concept that is specifically related to leak-prevention of the ducts (for raising an early warning as opposed to an after-the-fact accident alarm), the change will not impact $R1$ and $R3$ but will most likely impact $R4$ because the mon-

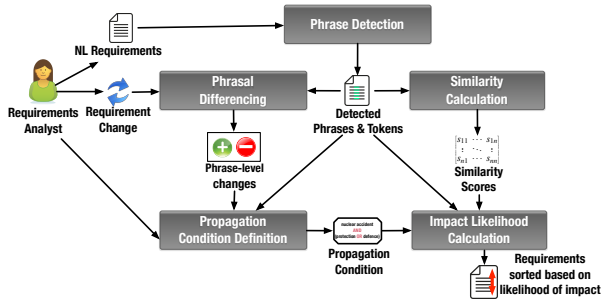

Figure 2: NARCIA components and user interactions

itoring policy in $R4$ may need to be further strengthened to support hazard warnings. NARCIA provides an intuitive mechanism so that analysts can specify the rationale as to why a change has occurred, and uses the specified rationale for more accurate identification of related requirements.

In the remainder of this tool demonstration paper, we outline NARCIA's method of use and the tool's main components. We further highlight key findings from our evaluation of NARCIA over two industrial case studies.

2. RELATED WORK

Limited work exists on change impact analysis for requirements [9]. Goknil et al. [6] propose a change impact analysis approach based on a requirements dependency model, covering, among other relationships, "requires" and "refines". Cleland-Huang et al. [4] develop a semi-automated approach for identifying requirements relationships in the context of change impact analysis. NARCIA takes inspiration from these earlier strands of work. Nevertheless, it does not require analysts to make the relationships between requirements explicit. Instead, NARCIA uses NLP for inferring these relationships based on the semantic context and the rationale for changes.

Some of the NLP techniques used by NARCIA have also been utilized in our previous work on checking conformance to requirements templates [2] and on the identification of requirements glossary terms [1]. NARCIA employs NLP to address a different problem than the ones in our earlier work.

3. TOOL OVERVIEW

NARCIA is a realization of our technical approach for change impact analysis, described in a recent conference paper [3]. Figure 2 presents an overview of NARCIA. First, the NL requirements provided by the user are processed in the *Phrase Detection* step. This step identifies the phrases in the requirements statements and the constituent tokens (words) of these phrases. Next, in the *Similarity Calculation* step, a similarity score is calculated for each pair of phrases and for each pair of constituent tokens.

To enable the analysis of requirements changes at a semantic level, the changes are lifted from words to phrases. This lifting is performed in the *Phrasal Differencing* step. A change, once cast into phrases, assists the user in expressing the change rationale. We take an operational view towards change rationale, defining it as a condition that specifies how a change should propagate in a requirements document. This condition is elicited from the user in the *Propagation Condition Specification* step. Finally, the results from the above steps are used in the *Impact Likelihood Calculation* step. In this step, for every requirement in the requirements

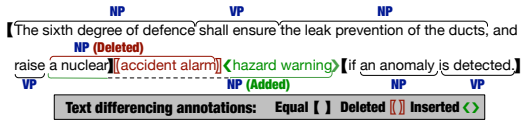

Figure 3: Phrase detection and phrasal differencing

document, a score, designed to predict the likelihood of the requirement being impacted, is computed. For simplicity, we refer to this score as impact likelihood. The output of this step is a list of requirements sorted in descending order of impact likelihood. The list helps the user focus on the requirements that are most likely to be impacted. In the rest of this section, we describe each of the steps in NARCIA.

3.1 Phrase Detection

NARCIA uses an NLP technique known as *text chunking* for detecting phrases in requirements. Text chunking decomposes sentences into non-overlapping segments [8]. In our context, the most important segments are Noun Phrases (NPs) and Verb Phrases (VPs). Figure 3 shows the NPs and VPs in requirement $R2$ of Figure 1 (pre- and post-change). We augment text chunking results through the application of heuristics that merge adjacent NPs in certain situations. These heuristics were developed in our previous work [1] with the aim of avoiding closely-associated NPs from being separated by text chunking. For example, one of the heuristics is to merge atomic NPs related by *of*, i.e., the *NP of NP* pattern. $R2$ has an occurrence of this pattern: the sixth degree of defence. The two atomic NPs in this segment do not provide much contextual information individually. It is therefore advantageous to merge them as a way to make change impact analysis more precise. The output of the *Phrase Detection* step is a list of phrases and their constituent tokens after removing stopwords such as determiners, pronouns, and prepositions. For instance, the phrase the sixth degree of defence, after stopwords removal, would yield the tokens sixth, degree, and defence.

3.2 Similarity Calculation

NARCIA quantifies the relatedness between phrases and between their constituent tokens using *syntactic* and *semantic* similarity measures. Syntactic measures compute a similarity score based on the overlap between the string content of phrases and tokens. Levenstein similarity [8] is an example of a syntactic measure, computing a score based on the minimum number of character edits required to change one string into another. For example, Levenstein similarity between phrases first degree of defence and first defence degree is 0.57. Semantic measures capture relatedness based on the (semantic) relationships defined in a dictionary. These measures are most suitable for relating synonyms as well as terms that are associated with one another through conceptual relationships such as *is-a* and *is-part-of*. Path similarity [8] is an example semantic measure where the relatedness of two words is calculated based on the shortest path from the words to a common word in an *is-a* hierarchy (e.g., defence *is-a* kind of protection). For instance, Path similarity between defence and protection is 0.5. Due to the complementarity of syntactic and semantic measures, combining these two classes of measures often leads to more accurate and robust results [10]. For a given pair of words (or phrases), NARCIA takes the maximum of syntactic and semantic similarities when computing the impact likelihoods.

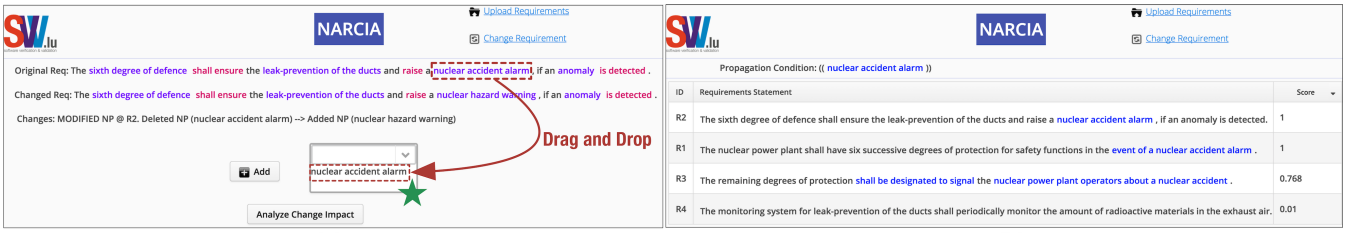

Figure 4: Screenshot of (a) propagation condition page and (b) output page (req.'s sorted by impact likelihoods)

3.3 Phrasal Differencing

Rather than treating changes at a syntactic level, which is the case in existing text differencing tools (e.g., Google's diff-match-patch[1]), NARCIA superimposes the phrasal information of a changed requirement over the syntactic changes. In this way, NARCIA can determine which phrases are affected. For example, when accident alarm is replaced by hazard warning in requirement $R2$ of Figure 3, the change is identified as the deletion of the NP nuclear accident alarm and the addition of the NP nuclear hazard warning. The superimposition of the phrasal information over the syntactic changes identified by a standard text differencing tool[1] is shown in Figure 3. In general, NARCIA treats a requirements update as a combination of additions and deletions of phrases. Adding (resp. deleting) a requirement is treated as the addition (resp. deletion) of the all the NPs and VPs in the requirement.

3.4 Propagation Condition Specification

As we argued earlier, the rationale behind a change can affect the way in which the change propagates. NARCIA provides a mechanism for expressing change rationale. To circumvent the cognitive limitations associated with answering "why" questions [5], NARCIA does not directly ask the user why a change has been made. Instead, the tool attempts to elicit a condition for characterizing how the change should propagate. We call this condition the *propagation condition*. The propagation condition is provided as a Boolean expression over phrases. For example, if the change rationale for $R2$ in Figure 1 is to rename nuclear accident alarm to nuclear hazard warning, then we expect that any impacted requirement should contain nuclear accident alarm, or a syntactic or semantic variation thereof. The condition should thus be: (nuclear accident alarm). Alternatively, if the rationale is to increase oversight over leak prevention of the ducts, the condition would be: (leak prevention of the ducts).

For more complex cases, examples of which can be found in our detailed technical approach [3], the propagation condition may involve multiple phrases related by the AND and OR Boolean operators. To shield the user from writing logical expressions, NARCIA provides a user interface for defining the propagation condition in Conjunctive Normal Form (CNF). A screenshot of this interface is shown in Figure 4(a). The interface displays the original and changed requirements, highlighting their phrases. The changes (detected by *Phrasal Differencing* as discussed earlier) are also shown. From this interface, the user can select the desired phrases by dragging and dropping them into the clause boxes (marked with a $\star$). Each clause box represents a disjunction of phrases. The user can create as many clause boxes as needed. As an alternative to drag and drop, the tool provides in each clause box a drop-down list (populated with the phrases of all requirements). The user can directly type into these lists for specifying the propagation condition.

[1]Google's diff-match-patch - `https://goo.gl/JfMU75`

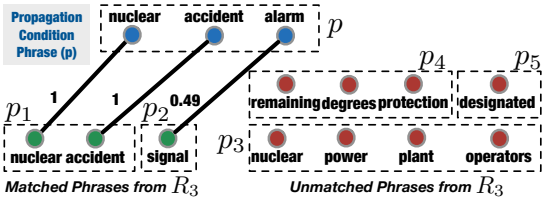

Figure 5: Example for impact likelihood calculation

3.5 Impact Likelihood Calculation

Given a propagation condition C, NARCIA calculates for every requirements statement R a normalized score. The score for every R is a measure of how likely R is to be impacted by the change associated with C. A higher score indicates a higher likelihood of impact. The requirements statements, sorted in descending order of the calculated scores, are then presented to the user. An example output from the tool is shown in Figure 4(b).

Briefly, the score for every R is derived from matching the set of phrases in R against the individual phrases of C. Figure 5 shows an example where the phrases of $R3$ (Figure 1) are matched against the phrase nuclear accident alarm. Stopwords have been removed from all the phrases. The edges in the matching are the optimal token matches. The weights on the edges are the pairwise token similarities, computed as described in Section 3.2.

To improve the quality of the matching, NARCIA applies certain heuristics aimed at minimizing the number of phrases used from a requirements statement for covering a given phrase from the propagation condition. To illustrate, we note that the token nuclear of p in Figure 5 is paired with nuclear in nuclear accident (p_1) rather than with nuclear in nuclear power plant operators (p_3). The latter match would have increased (from 2 to 3) the number of phrases used from $R3$. The impact likelihood for $R3$ is calculated based on the number of phrases used from $R3$ in the optimal match, the number of tokens matched, and the degree of similarity (weights) between the matched tokens. For $R3$ against the phrase nuclear accident alarm, the score is ≈ 0.77.

When the propagation condition has multiple phrases, an impact likelihood is calculated for each of these phrases, with the resulting scores combined according to the Boolean operators in the condition. See [3] for full details.

4. EVALUATION

NARCIA has been evaluated on two industrial case studies [3]. The first study is over the requirements document of

a satellite simulation component that is being developed by our industry collaborator, SES TechCom. This document has 160 requirements and 9 changes. The second study concerns the requirements document of a mobile service platform, containing 72 requirements and 5 changes.

To identify the most accurate similarity measures, we experimented with combinations of 10 syntactic and 9 semantic similarity measures. We observed that the combination of *Levenstein* (syntactic) and *Path* (semantic) yielded the most reliable results across the two case studies.

We have further developed guidelines for assisting analysts in deciding how much of a sorted list generated by NARCIA is worth inspecting, i.e., likely to contain impacted requirements. Our guidelines are based on the internal characteristics of the generated sorted lists, and more specifically, on the pattern of decline in the calculated impact likelihoods [3]. This pattern is automatically analyzable, meaning that recommendations can readily be made to the user as to how much of a list to inspect, without requiring them to do any further analysis on their own.

Using our proposed guidelines and the optimal combination of similarity measures mentioned above, we evaluated NARCIA in terms of: (1) the number of impacted requirements the tool misses and (2) the amount of futile inspection effort, assumed to be a linear function of the number of *non-impacted* requirements that our guidelines would recommend the analysts to inspect. For the 14 changes considered, the sum of the cardinality of the impact sets from the gold standards is 106. Of these, our tool missed only one impacted requirement. This miss was due to a tacit relationship between two phrases, namely tourist attraction and point of interest. The former phrase is a conceptual specialization of the latter; however, since the two phrases have no syntactic or semantic resemblance, NARCIA cannot detect the relationship between them. This case hints at a limitation in our approach which needs to be remedied through additional user-provided information, e.g., a conceptual model.

As for the second measure (futile effort), for 13 out of the 14 changes, the measure ranged between 1% – 8% of the total number of requirements. For one change, the rate was 45% due to the propagation condition not being precise enough. We anticipate that such cases can be improved by having a human feedback loop, whereby the propagation condition is refined by the user if the first few requirements in the sorted list turn out to be irrelevant to the change.

Lastly, we observed that NARCIA produced results within reasonable time (worst case of 11 sec. for interactive user requests in our larger case study), suggesting that the tool will scale to larger requirements documents. For details, see [3].

In summary, our evaluation suggests that NARCIA is accurate and shows promise for use in practical settings.

5. IMPLEMENTATION & AVAILABILITY

NARCIA has been implemented as a Java web application in the Vaadin UI framework[2]. For phrase detection, we use GATE[3] and OpenNLP[4]. Syntactic and semantic similarities are calculated by the SimPack[5] and SEMILAR[6] libraries, respectively. Our phrasal differencing algorithm uses Google's diff-match-patch[1]. NARCIA is approximately 8K lines of code, excluding comments and third-party libraries.

Additional details about NARCIA including executable files, and a screencast covering motivations, tool architecture, and usage are available on the tool's website at:

https://sites.google.com/site/svvnarcia/

[2]Vaadin UI Framework - `http://vaadin.com/`
[3]GATE NLP Framework - `http://gate.ac.uk/`
[4]OpenNLP - `http://opennlp.apache.org`
[5]SimPack - `http://www.ifi.uzh.ch/ddis/simpack.html`
[6]SEMILAR - `http://www.semanticsimilarity.org`

6. CONCLUSION

We presented our tool, NARCIA, for analyzing the impact of changes in natural language requirements. The main novelties of the tool are in: (1) the use of phrases as the basic semantic units for identifying the relationships between requirements, and (2) explicit handling of change rationale for increasing the accuracy of change impact analysis. NARCIA has been evaluated on two industrial case studies with promising results. In the future, we plan to enhance NARCIA with means for handling complex requirements relationships that cannot be detected using NLP alone. We further intend to extend the tool so as to support multiple simultaneous changes. Finally, we plan to conduct user studies to better evaluate the practical utility of the tool. The most important question that we need to answer through user studies is whether users are able to conveniently specify the propagation conditions required by our approach.

7. ACKNOWLEDGMENT

We gratefully acknowledge funding from Luxembourg's National Research Fund (FNR/P10/03 - Verification and Validation Laboratory and FNR-6911386).

8. REFERENCES

[1] C. Arora, M. Sabetzadeh, L. Briand, and F. Zimmer. Improving requirements glossary construction via clustering: Approach and industrial case studies. In *ESEM'14*, 2014.

[2] C. Arora, M. Sabetzadeh, L. Briand, and F. Zimmer. Automated checking of conformance to requirements templates using natural language processing. *IEEE TSE*, 2015. (to appear).

[3] C. Arora, M. Sabetzadeh, A. Goknil, L. Briand, and F. Zimmer. Change impact analysis for natural language requirements: An NLP approach. In *RE'15*, 2015. (to appear).

[4] J. Cleland-Huang, C. K. Chang, and M. J. Christensen. Event-based traceability for managing evolutionary change. *IEEE TSE*, 29(9), 2003.

[5] A. Dutoit, R. McCall, I. Mistrik, and B. Paech. *Rationale Management in Software Engineering*. Springer, 2006.

[6] A. Goknil, I. Kurtev, K. van den Berg, and W. Spijkerman. Change impact analysis for requirements: A metamodeling approach. *IST*, 56(8), 2014.

[7] P. Jönsson and M. Lindvall. Impact analysis. In A. Aurum and C. Wohlin, editors, *Engineering and Managing Software Requirements*. Springer, 2005.

[8] D. Jurafsky and J. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing*. Prentice Hall, 2000.

[9] S. Lehnert. A review of software change impact analysis. In *Technical Report - Technische Universitat Ilmenau*, 2011.

[10] S. Nejati, M. Sabetzadeh, M. Chechik, S. Easterbrook, and P. Zave. Matching and merging of variant feature specifications. *IEEE TSE*, 38(6), 2012.