Quality of Requirements Specifications: A Preliminary Overview of an Automatic Validation Approach

Alberto Rodrigues da Silva
Instituto Superior Técnico
Lisbon, Portugal
alberto.silva@tecnico.ulisboa.pt

ABSTRACT

Requirements specifications describe multiple technical concerns of a system and are used throughout the project life-cycle to help sharing a common understanding among the stakeholders. In spite a lot of interest has been given to manage the requirements lifecycle, which resulted in numerous tools and techniques becoming available, however, little work has been done that address the *quality of requirements specifications*. Most of this work still depends on human-intensive tasks made by domain experts but are time-consuming and error prone, and have negative consequences in the success of the project. This paper briefly proposes an automatic validation approach that, with proper tool support, can help to mitigate some of these limitations and therefore can increase the quality of requirements specifications, in particular those that concerns consistency, completeness, and unambiguousness.

Keywords

Requirements Specification, Quality of Requirements Specification, Requirements Validation.

1. INTRODUCTION

Requirements Engineering (RE) intends to provide a shared vision and understanding of the system to be developed between business and technical stakeholders [1,8,9]. The adverse consequences of disregarding the importance of the early activities covered by RE are well-known [2,3]. System requirements specification, software requirements specification or just requirements specifications (SRS) is a document that describes multiple technical concerns of a software system [1,8,9]. An SRS is used throughout different stages of the project life-cycle to help sharing the system vision among the main stakeholders, as well as to facilitate communication and the overall project management and system development processes. For achieving an effective communication, everyone should be able to communicate by means of a common language, and natural language provides the foundations for such language. Natural language is flexible, universal, and humans are proficient at using it to communicate with each other. Natural language has minimal adoption resistance as a requirements documentation technique [1,9].

However, although natural language is the most common and preferred form of requirements representation [4], it also exhibits some intrinsic characteristics that often present themselves as the root cause of many requirements quality problems, such as incorrectness, inconsistency, incompleteness and ambiguousness [1,9]. From these causes, in this paper we emphasize inconsistency and incompleteness because avoiding – or at least mitigating – them requires significant human effort due to the large amount of information to process when combined with inadequate tool support, namely to perform the typical requirements linguistic analysis. On the other hand, although ambiguity and incorrectness – by definition – cannot be fixed without human validation, we consider that the tasks required to minimize the effects of both inconsistency and incompleteness (and also ambiguity at some extent) can be automated if requirements are expressed in a suitable language, and if adequate tool support is provided.

In our research we consider the RSLingo approach [6] as a starting point for it. RSLingo is a recent and ambitious approach for the formal specification of software requirements that uses lightweight Natural Language Processing (NLP) techniques [5] to translate informal requirements – originally stated in unconstrained natural language by business stakeholders – into a formal representation provided by a language specifically designed for RE. Unlike other RE approaches, which use languages that typically pose some difficulties to business stakeholders (namely graphical modeling languages such as UML or SysML, whose target audience are engineers), RSLingo encourages business stakeholders to actively contribute to the RE process in a collaborative manner by directly authoring requirements in natural language. To achieve this goal, RSLingo provides (1) a language for defining linguistic patterns that frequently occur in requirements specifications written in natural language (the RSL-PL language), (2) a language that covers most RE concerns, in order to enable the formal specification of requirements (the RSL-IL language), and (3) an information extraction mechanism that, based on the linguistic patterns defined in RSL-PL, translates the captured information into a formal requirements specification encoded in RSL-IL [6,7,8].

However, RSLingo does not provide yet any guarantees that the RSL-IL specifications have the required quality. So, this research starts from requirements specified in RSL-IL, and not in natural language, because the challenges to produce RSL-IL specifications from natural language specifications were already discussed and proposed [6,7,10]. Therefore, the main contribute of this research is to propose and discuss that, with proper tool support, we can increase the overall quality of SRSs as well as the productivity associated to documentation and validation tasks. To the best of our knowledge, no further works have been proposed before in relation to this complex approach and the way we support automatic validation of SRSs.

SAC'14, March 24-28, 2014, Gyeongju, Korea.

2. THE GENERAL APPROACH[1]

Figure 1 suggests the operation of the SRS validation process with its main input, outputs, and supporting resources. The major input is the `Requirements Specs` file that is the SRS defined in the RSL-IL concrete syntax [7]. The outputs are the `Parsing Log` file and the `Test Reports` file with the errors and warnings detected by the tool during its execution, respectively during the `Parsing` and the `Validation` processes. Additionally, there are some supporting resources used to better extend and support the tool at run-time, namely: `Quality Tests`, `Configuration`, and `Lexical Resources`.

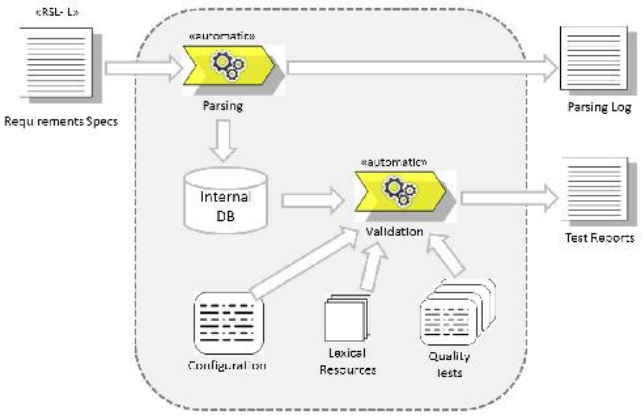

Fig. 1. Overview of the validation process.

`Quality Tests` consist in a set of tests directly implemented in a given programming language (currently in C#) and having additional metadata such as name, description and quality criteria type. `Configuration` is a resource used to support the validation in a flexible way. For example, this resource can allow requirements engineers to configure the level of completeness needed for their purpose, in a project basis. This means that different projects may have different needs regarding completeness of their specifications. Finally, `Lexical Resources` are public domain resources (such as WordNet or VerbNet) that support some tests, mostly those related with linguistic issues.

3. TOOL SUPPORT

The proposed approach has to be implemented by a concrete software tool in order to offer a real interest and utility. Due to that we have implemented the SpecQuA (Requirements Specification Quality Analyzer) tool with the purpose to show and discuss the practicability of this approach.

The SpecQuA[2] has the following objectives. First, provide SRS's quality reports: for a given RSL-IL specification, the system should be able to provide results for quality tests applied to that specification. Second, easily add and configure new quality tests: it should be easy to develop and add new quality tests in order to extend the tool; additionally it should be easy to configure those tests. Third, Web collaborative workspace: the tool should be accessible via a Web browser and should provide the means for multiple users to work together while analyzing RSL-IL specifications.

4. CONCLUSION

This research extends the RSLingo approach by considering that the requirements are represented in RSL-IL automatically extracted from natural language specifications or authored directly by users. However, it was not possible to guarantee the quality of RSL-IL specifications without human-intensive work, which is still time-consuming and error prone. Therefore, this paper proposes a generic approach to automatically validate these specifications and briefly introduces the toolset (i.e., the SpecQuA software tool) that shows the practicability and utility of this proposal. The flexibility of the toolset and the cases studies developed so far allows us to state that the proposed approach helps to mitigate some of the mentioned limitations, in particular in what respect inconsistency, incompleteness and ambiguousness.

For future work we plan to develop other features on the toolset, in particular those related with the support of the collaborative environment, allowing end-users to author and validate directly their requirements [11], eventually in different representations, beyond natural language and RSL-IL. Additionally, we intend to explore the integration of RE with MDE (Model Driven Engineering) approaches [12,13] to increase the quality and productivity of Software Engineering in general.

5. REFERENCES

[1] Pohl, K.: Requirements Engineering: Fundamentals, Principles, and Techniques, Springer, 2010.

[2] Emam, K., Koru, A.: A Replicated Survey of IT Software Project Failures. IEEE Software, 25(5), 2008, 84–90.

[3] Davis, A.: Just Enough Requirements Management: Where Software Development Meets Marketing. Dorset House Publishing, 2005.

[4] Kovitz, B.: Practical Software Requirements: Manual of Content and Style. Manning,1998.

[5] Bird, S., Klein, E., Loper, E.: Natural Language Processing with Python. O'Reilly Media, 1st edition, 2009.

[6] Ferreira, D., Silva, A. R.: RSLingo: An Information Extraction Approach toward Formal Re-quirements Specifications. In: Proc. of the 2nd Int. Workshop on Model-Driven Requirements Engineering (MoDRE 2012), IEEE Computer Society, 2012.

[7] Ferreira, D., Silva, A. R.: RSL-IL: An Interlingua for Formally Documenting Requirements. In: Proc. of the of Third IEEE International Workshop on Model-Driven Requirements Engineering (MoDRE 2013), IEEE Computer Society, 2013.

[8] Sommerville, I., Sawyer, P.: Requirements Engineering: A Good Practice Guide. Wiley, 1997.

[9] Robertson, S. , Robertson, J.: Mastering the Requirements Process, 2nd edition. Addison-Wesley, 2006.

[10] Ferreira, D., Silva, A. R.: RSL-PL: A Linguistic Pattern Language for Documenting Software Requirements. In: Proc. of the of Third International Workshop on Requirements Patterns (RePa 2013), IEEE Computer Society, 2013.

[11] Ferreira, D., Silva, A. R.:Wiki supported collaborative requirements engineering, Proceedings of the 4th International Symposium on Wikis. ACM, 2008.

[12] Silva, A. R., et al.: Integration of RE and MDE Paradigms: The ProjectIT Approach and Tools", IET Software Journal,1(6), 2007.

[13] Ribeiro, A., Silva, A. R.: XIS-Mobile: A DSL for Mobile Applications, Proceedings of SAC 2014 Conference, ACM, 2014.

[1] Acknowledgement: This work was supported by national funds through FCT – Fundação para a Ciência e a Tecnologia, under the PEst-OE/EEI/LA0021/2013 project. Particular thanks to the students David Ferreira and Joao Marques for their strong involvement in this research.

[2] Preliminary prototype available at http://specqua.apphb.com