An Initial Approach to Reuse Non-Functional Requirements Knowledge

Rodrigo Veleda, Luiz Marcio Cysneiros
School of Information Technology – York University, Canada
rveleda@yorku.ca, cysneiro@yorku.ca

Abstract. Non-Functional Requirements (NFR) can be seen as qualities that software should deliver to cope with the stakeholders' demands. NFRs are fuzzy in nature and hence hard to identify. Despite the fact that both developers and users may value NFRs, they frequently do not identify the need for an NFR. Even when an NFR is identified as required, possible solutions to implement this NFR may be hard to figure out. Furthermore, interdependencies among NFRs may implicate that a solution for one NFR may, at the same time, bring synergy to one NFR while conflicting with another. One approach to deal with that is to use Softgoal Interdependency Graphs (SIG) to capture knowledge describing alternatives to implement NFRs. We have obtained empirical evidence that using catalogues can help eliciting NFRs despite the fact that catalogues do not scale too well. To address this question, we have investigated the use of ontologies and semantic web techniques to represent SIGs in a machine readable format. We have produced a tool (NDR) that starts to use these concepts. In its current form, the NDR tool only allows very basic queries done manually. The NDR tool is part of the NDR framework which will facilitate the reuse of NFR knowledge on Alternatives to incorporate NFRs into the design of target systems.

Keywords: Non-Functional Requirements, Reuse, Knowledge

1 Introduction.

Requirements engineers have to address both functional and non-functional requirements to develop software systems [1]. Functional requirements are responsible to represent what the system is capable of in terms of available features. On the other hand, non-functional requirements are known to represent quality attributes [2, 3]. These quality characteristics include privacy, performance, usability and other similar aspects related to the quality of a software system.

The first challenge for eliciting NFRs lies on the fact that they are fuzzy in nature and quite frequently are missed both by software engineers and stakeholders. Furthermore, choosing one solution to implement one NFR might bring synergies and perhaps most importantly conflicts to another NFR bringing the perception that one NFR can rarely be expected to be 100% satisfied. We use the term *satisfice* [4, 5] to represent the idea that NFR is satisfied within acceptable limits.

Some works have proposed the use of catalogues representing knowledge to *satisfice* NFRs as a way of helping not only to elicit NFRs but also to reason about the complexity involved in choosing alternatives to *satisfice* an NFR [4], [6]. In fact, empirical work has suggested that the use of catalogues can contribute to avoiding omissions and missed conflicts, despite the fact that SIGs do not scale too well [6].

These catalogues are implemented using Softgoal Interdependency Graphs (SIG) [4]. SIG catalogues promote a graphical representation of essential quality characteristics for

satisficing a given non-functional requirement. SIGs also demonstrate possible tradeoffs among non-functional requirements in the target system.

In this paper, we discuss an ongoing research by introducing the NDR Tool. The NDR Tool is currently under development, and it is part of the NDR Framework which aims to facilitate the reuse of non-functional requirements knowledge captured in SIGs. Our application strives the extraction of present knowledge from SIGs and represents it in a machine readable format using ontologies and semantic web techniques [7, 8] Furthermore, the tool proposes the storage of collected knowledge within an ontology repository that currently follows the proposed Non-functional requirement and Design Rationale (NDR) Ontology [7]. We are developing the NDR tool to store as many alternatives as possible to *satisfice* NFRs. It will also use RDF [9] queries for retrieving alternatives for one specific problem, allowing software engineers to select one alternative and import this alternative to its *i** models representing the target system. This work depicts the tool's currently available features and also denotes the potential challenges and future tasks for implementation.

The remainder of this paper is structured as follows: Section 2 illustrates the related work. Section 3 describes the objectives of our research and its scientific contribution. Finally, Section 4 presents the on-going and future work.

2 Related Work

Some works [10, 11] focus on experienced-based elicitation and recommendation for the use of non-functional requirements in software service. Others [12, 13, 14, 15], aim the use of ontologies to assist non-functional requirements elicitation. Nevertheless, none of these proposed works addresses the challenge of investigating the potential tradeoffs between multiple non-functional requirements. Nor they interact with *i** tools to facilitate the reuse of the knowledge.

Considering the use of SIGs aiming the reuse of knowledge, Sancho et al. [16] proposes an ontological database. Their work consists of two ontologies both written in OWL [17]: The NFR ontology and the SIG ontology. The NFR ontology explains the NFRs concept and relationship among them. The SIG ontology depicts SIG constructs and their relationships. We have identified two shortcomings within this approach. First, the SIG ontology does not define any class to describe the Correlation interdependency between Softgoals therefore limiting reasoning involving more than one NFR. Second, it does not enforce the use of proper kind of Softgoals as parent and offspring of each Refinement. The NDR ontology is based on this work.

Hazeem et. al [13] introduced the ElicitO. ElicitO is an ontology-based tool that supports non-functional requirements elicitation by providing a knowledge base that relates non-functional requirements and its associated metrics. Also, Najera [14] highlights an approach that uses OWL and RDF targeting the representation of *i** variants. Unfortunately, the reuse of non-functional requirements knowledge is not tackled in this work nor is cited as future work.

3 Research objectives and scientific contribution

Our long-term goal is to develop a framework that can help software engineers to elicit and model non-functional requirements empowered by the knowledge that has previously been elicited and validated. We believe that the use of a well-defined knowledge base could play an essential role to achieve this goal. Therefore, our environment will emerge

as the result of further developing our ontology and tools to store and retrieve knowledge on *satisficing* non-functional requirements.

Hence, our first goal is to develop further the NDR tool to efficiently store NFR knowledge while allowing querying at different levels of granularity for retrieving existing information. .

The next step will be to develop mechanisms to import and export knowledge from and to other Tools that support *i**. Techniques to import SIGs from *i** tools will be implemented as well as the ability for choosing alternatives from existing SIGs in the NDR tool to be imported into i* models expressed in tools such as jUCMNav [18].

At first, this environment will only be open to accepting queries from the academic community. However, in a near future we envision to accept contributions from other research groups working with NFR knowledge to enrich the existing knowledge base. At the same time, we will also allow members from the industry to query the knowledge base and submit comments with their perception. At a later stage, contributions to add to the knowledge base will be accepted from a broad audience.

4 Ongoing and Future work

4.1 NDR Ontology

The NDR Ontology proposed in [7] represents NFRs and design argumentative rationale knowledge in a machine-readable format. This representation follows the proposed standards of OWL. Therefore, each examined SIG catalog is converted into semantic graphs, establishing new sets of instances of NDR Ontology and expressing a machine-readable form. In operational terms, RDF is widely used to describe ontologies (mainly at semantically enriched Web sites). RDF encodes information as triplets (resources) that relate a property to other resources or plain literal data. Thus, RDF models are directed labeled graphs that allow representing meaningful contents. RDF Schema (RDFS) [19] allows describing properties and classes of RDF resources and supports a generalization hierarchy for properties and classes. As a short-term goal, we will further develop the ontology and the tool capability of producing the necessary SPARQL queries [20] from user-friendly dialogs.

In our proposed approach, we envision to provide the NDR Ontology available within the NDR Tool in a cloud environment. We believe that making our proposed approach available in a cloud will facilitate not only its use by different audiences but also the opportunity to receive contributions to enlarge the knowledge base that will be available.

Currently, we have implemented a proof of concept version of the NDR Ontology on our cloud ontology repository. In order to have a user-oriented layer displaying details of the target ontology graphically, we have integrated our platform with WebVOWL [21]. The WebVOWL is a web-application that implements the Visual Notation for OWL Ontologies (VOWL) [22]. A graphical visualization of the NDR Ontology version is currently implemented in

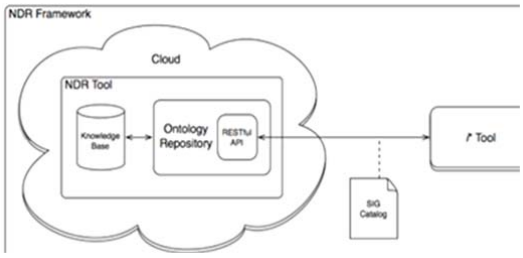

Fig. 1. The NDR Framework Architecture

our platform[1]. We are in the process of importing SIG catalogs into the target environment to also demonstrate the knowledge expansion graphically.

4.2 NDR Framework Conceptual Architecture

To maintain and assure a reliable integration between SIGs developed with *i** Tools and its conversion into ontology knowledge, we propose the NDR Framework. Figure 1 illustrates its conceptual architecture.

The NDR Tool is mainly composed of a knowledge base and an ontology repository. As mentioned in Section 4.1, currently, only the NDR Ontology is implemented within the platform. We aim to develop a generic ontology repository that can be instantiated in domain specific ontologies to provide extensibility of our platform. In other words, the NDR Tool will be able to handle several ontologies, preserving a valuable and vast knowledge base.

Aside from holding ontologies, the NDR Tool will also be in charge of handling the conversion of SIG catalogs into ontology instances. Besides the execution of parsers designed for each type of supported SIG and ontology, our platform will detect if the artefact that is being provided contains relevant knowledge based on definitions manually defined by repository administrators. An approach based on Open-Source concepts will be developed to handle this.

Access to the knowledge contained in the NDR Tool will be possible through the use of web services. We envision to implement RESTful web service endpoints that can be invoked externally by third-party *i** applications. Essential features such as artefact importation and knowledge retrieval will be implemented within these services, facilitating future integrations.

To illustrate an appropriate real-world example of the applicability of the NDR Framework, we portrait a SIG representing the non-functional requirement of Transparency as demonstrated in Figure 2 is uploaded into our framework. The NDR Tool will extract the knowledge in the provided SIG based on the settings manually defined by the repository administrator.

Then, the tool will convert the selected knowledge into a machine-readable format, following the NDR Ontology standards.

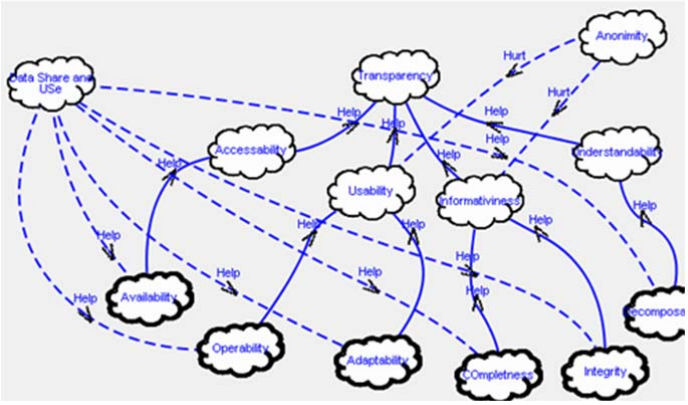

Fig. 2. Transparency SIG

As it is noticeable in Figure 2, the visual information represented in the SIG illustrates the scalability problem mentioned earlier in this paper. As more details such as correlation and decompositions are added into the SIG, its understandability

[1] http://132.206.206.138:8888/

and clarity becomes affected.

By having the knowledge ready in a machine-readable format, the NDR tool will manage to update the ontology with new individual instances and persist the modifications into a database. At this point, the extracted knowledge from the Transparency SIG is available for reuse.

The reuse of stored knowledge will be possible by accessing web service endpoints. Once an endpoint is reached, the NDR Tool will handle the request by applying the SPARQL queries over the stored knowledge. For instance, a software engineer wants to know which non-functional requirements are directly related to the *satisficing* of Transparency. After receiving this request, the NDR Tool will execute an SPARQL query similar to the following:

```
SELECT DISTINCT ?interlinkId ?softgoalParent ?softgoalSpring
?contributionKind WHERE {?interlinkId rdf:type ndr:Correlation.
?interlinkId ndr:correlationHead ?softgoalParent. ?interlinkId
ndr:correlationTail ?softgoalSpring. ?interlinkId ndr:contributionKind
?contributionKind.}
```

Basically, this query is selecting all the correlations that somehow affect the *satisficing* of the Transparency *softgoal*. Table 1 demonstrates the internal result of this query execution.

Table 1. SPARQL query execution result

interlinkId	softgoalParent	softgoalSpring	contribution
ndr:UH_correlation2	ndr:Informativiness	ndr:Anonymity	ndr:Hurt
ndr:UH_correlation7	ndr:Integrity	ndr:Data_Share_and_Use	ndr:Help
ndr:UH_correlation1	ndr:Usability	ndr:Anonymity	ndr:Hurt
ndr:UH_correlation6	ndr:Completeness	ndr:Data_Share_and_Use	ndr:Help
ndr:UH_correlation4	ndr:Operability	ndr:Data_Share_and_Use	ndr:Help
ndr:UH_correlation8	ndr:Decomposability	ndr:Data_Share_and_Use	ndr:Help
ndr:UH_correlation5	ndr:Adaptability	ndr:Data_Share_and_Use	ndr:Help
ndr:UH_correlation3	ndr:Availability	ndr:Data_Share_and_Use	ndr:Help

As an outcome of this process, the NDR Tool retrieves the requested information and the result is ready to be sent back to the user. It is noteworthy to mention that the possibility of having results in a graphical way instead of machine-readable format will depend on the level of integration with a given *i** tool.

4.3 jUCMNav Integration

We aim at integrating the NDR Tool with jUCMNav to have SIG catalogs imported/exported into/from our platform.

jUCMNav is an open-source modeling tool that supports the *i** Framework. One of the reasons that we decided to integrate our approach with jUCMNav is its extensibility. Another reason is that jUCMNav is a cross-platform application endeavor. It is well documented and presents a steady process of growth. Lastly, by integrating the NDR tool to jUCMNav, we can provide a graphical visualization to resulting SIGs from queries. Although we will be mainly focusing on jUCMNav at first, all our efforts will keep in mind the need to develop an interactive approach that can work with as many *i** tools as possible.

References.

1. Chung, L., do Prado Leite, J.: On Non-Functional Requirements in Software Engineering. In: Borgida, A., Chaudhri, V., Giorgini, P., and Yu, E. (eds.) Conceptual Modeling: Foundations and Applications. pp. 363–379. Springer Berlin Heidelberg (2009)
2. Boehm, B.W., Brown, J.R., Kaspar, H., Lipow, M.: Characteristics of software quality. North-Holland, Amsterdam (1978)

3. Keller, S.E., Kahn, L.G., Panara, R.B.: Specifying software quality requirements with metrics. System and Software Requirements Engineering. 145–163 (1990)
4. Chung, L., Nixon, B.A., Yu, E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering. Springer US (1999)
5. Simon, H.A.: The sciences of the artificial. (1996)
6. Cysneiros, L.M.: Evaluating the Effectiveness of using Catalogues to Elicit Non-Functional Requirements. In: Proc. of 10th Workshop in Requirements Engineering, pp. 107–115 (2007)
7. Lopez, C., Cysneiros, L.M., Astudillo, H.: NDR Ontology: Sharing and Reusing NFR and Design Rationale Knowledge. Managing Requirements Knowledge, 2008. MARK '08. First International Workshop on. pp. 1–10 (2008)
8. López, C., Inostroza, P., Cysneiros, L.M., Astudillo, H.: Visualization and comparison of architecture rationale with semantic web technologies. Journal of Systems and Software. 82, 1198 – 1210 (2009)
9. Carroll, J.J, Klyne, G.: RDF concepts and abstract syntax, http://www.w3.org/TR/rdf-concepts/ (2004)
10. Doerr, J., Kerkow, D., Koenig, T., Olsson, T., Suzuki, T.: Non-functional requirements in industry - three case studies adopting an experience-based NFR method. Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on. pp. 373–382 (2005)
11. Zhang, X.-L., Chi, C.-H., Ding, C., Wong, R.K.: Non-functional Requirement Analysis and Recommendation for Software Services. Web Services (ICWS), 2013 IEEE 20th International Conference on. pp. 555–562 (2013)
12. Wang, T., Si, Y., Xuan, X., Wang, X., Yang, X., Li, S., Kavs, A.J.: A QoS Ontology Cooperated with Feature Models for Non-functional Requirements Elicitation. Proceedings of the Second Asia-Pacific Symposium on Internetware. pp. 17:1–17:4. ACM, New York, NY, USA (2010)
13. Al Balushi, T., Sampaio, P.F., Dabhi, D., Loucopoulos, P.: ElicitO: A Quality Ontology-Guided NFR Elicitation Tool. In: Sawyer, P., Paech, B., and Heymans, P. (eds.) Requirements Engineering: Foundation for Software Quality. pp. 306–319. Springer Berlin Heidelberg (2007)
14. Najera, K., Martinez, A., Perini, A., Estrada, H.: An Ontology-Based Methodology for Integrating i* Variants. Presented at the June (2013)
15. Guizzardi, R., Li, F.-L., Borgida, A., Guizzardi, G., Horkoff, J., Mylopoulos, J.: An ontological interpretation of non-functional requirements. Presented at the Formal Ontology in Information Systems: Proceedings of the Eighth International Conference (FOIS 2014) (2014).
16. Sancho, P.P., Juiz, C., Puigjaner, R., Chung, L., Subramanian, N.: An Approach to Ontology-aided Performance Engineering Through NFR Framework. Proceedings of the 6th International Workshop on Software and Performance. pp. 125–128. ACM, New York, NY, USA (2007)
17. McGuinness, D.L., van Harmelen, F.: OWL Web Ontology Language Overview. W3C Recommendation, http://www.w3.org/TR/2004/REC-owl-features-20040210/ (2004)
18. Mussbacher, G., Amyot, D.: Goal and scenario modeling, analysis, and transformation with jUCMNav. Software Engineering - Companion Volume, 2009. ICSE-Companion 2009. 31st International Conference on. pp. 431–432 (2009)
19. Brickley D., Guha,R.: RDF vocabulary description language 1.0: RDF Schema, W3C working draft, http://www.w3.org/TR/2002/WD-rdf-schema-20021112/ (2002).
20. Prud'hommeaux, E., Seaborne, A.: SPARQL Query Language for RDF.
21. Lohmann, S., Link, V., Marbach, E., Negru, S.: WebVOWL: Web-based Visualization of Ontologies. In: Lambrix, P., Hyvönen, E., Blomqvist, E., Presutti, V., Qi, G., Sattler, U., Ding, Y., and Ghidini, C. (eds.) Knowledge Engineering and Knowledge Management. pp. 154–158. Springer International Publishing (2015)
22. Negru, S., Lohmann, S.: A Visual Notation for the Integrated Representation of OWL Ontologies. Proceedings of the 9th International Conference on Web Information Systems and Technologies (WEBIST '13). pp. 308–315. SciTePress (2013)