Bridging the Gap among Academics and Practitioners in Non-Functional Requirements Management: Some Reflections and Proposals for the Future

Xavier Franch[1], David Ameller[1], Claudia P. Ayala[1], Jordi Cabot[2]

[1] GESSI research group, Universitat Politècnica de Catalunya, Barcelona, Spain
franch|dameller|cayala@essi.upc.edu
[2] AtlantMod, INRIA - École des Mines de Nantes, France
jordi.cabot@inria.fr

Abstract. The software engineering community has paid a lot of attention to the study of non-functional requirements (NFRs). Along time, framing NFRs into an articulated framework has become an elusive target. As a consequence, practitioners usually integrate NFRs in the different system life-cycle activities in an ad-hoc manner. In this work, we summarise the results of a recent empirical study involving 13 software architects from the Spanish. These results serve as the basis for discussion about possible ways to bridge the gap between academics and practitioners in the management of NFRs.

Keywords: non-functional requirement, NFR, software architect

1 Introduction

Non-functional requirements (NFRs) have triggered not just a lot of research in the requirements engineering community, but also a significant number of debates about their nature, their typology and even their existence. We may even hear researchers arguing that NFRs do not exist since at the end every requirement is related to some system functionality.

Martin Glinz presented at RE'07 an excellent paper that has become a source of inspiration for many of us [1]. I (Xavier) perfectly remember the discussion that it triggered in the conference and the many conversations in the corridors discussing the several aspects proposed. To summarize, Martin Glinz discussed three types of problems with the notion of NFRs:

- Definition problems. Terminological and conceptual discrepancies.
- Classification problems. Existence of many classification schemas.
- Representation problems. Different strategies for representing and refining NFRs.

Before and after that paper, the amount of methods proposed by academics for dealing with NFRs is large. However, it is not clear how well they have transferred to industry. In the last years, we have conducted several empirical studies investigating how NFRs are dealt with in practice [2][3][4][5]. In this paper, we summarize the results of one of them, observe how practitioners hardly adopt proposals made by academics, and discuss some possible ways to mitigate this situation.

2 The Study

In [2], we reported an interview-based study involving 13 software architects from small-to-medium Spanish organizations focusing on they deal with NFRs in their projects. The main findings can be summarised as follows.

Understanding. Architects did not share a common vocabulary for types of NFRs and showed some misunderstandings. During the interviews, architects required additional explanations for some terms, used themselves misleading terms or directly denoted some properties with wrong terms.

NFR elicitation. 77% of architects were the main source of NFRs in their projects. The main explanation seems to be that architects consider themselves to be the real experts when it comes to defining efficiency, reliability, and other similar aspects. Interestingly, the other 23% were the only cases in the study in which the interviewee was working on an outsourced project. On the other hand, all respondents agreed that deciding NFRs is a gradual and iterative process throughout the system lifecycle.

NFR documentation. 69% of interviewees acknowledged that they had not documented the NFRs at all, either intentionally or not. Some interviewees emphasized that documentation is only necessary if the client or the critical nature of the domain so requires. From the rest, 15% declared that they didn't keep the documentation up-to-date.

NFR satisfaction. 85% of respondents claimed that all NFRs had been satisfied by the end of the project. However, when asked how they had validated them, their answers were vague, except in one case that used formal techniques based on statistical analysis and simulation to check the system's reliability.

NFR validation. 61,5% of interviewees performed some validation of applications against NFRs, but each one validated only one to three NFR types. The types considered were: efficiency (46% of respondents); accuracy (23%); usability (15%); and reliability (15%). No other type of NFR was mentioned as explicitly validated.

NFR management. All the architects declared that no specific tools were used for NFR management.

Full details of the study are available at [2]. An extension of this study with some other research questions related to architectural decision-making is also available at [3].

3 Discussion

A general observation that emerges from our empirical study is that there is a gap between the methods that academics propose around NFR management and those that software architects in industry adopt and apply in their daily work. Academics sometimes argue (and maybe this is partly true) that practitioners are reluctant to adopt their proposals. However, as stated in the SEMAT initiative: "Academics are not entirely blameless: some research has concentrated on problems that pose scientific challenges but are hard to relate to the concerns of industry" [6]. In this section we discuss some possible directions for researchers that may help in bridging this gap.

3.1 Improve communication between practitioners and academics

Some of the observations arisen from our study show that assumptions made by academics in their research on NFRs and software architectures do not necessarily hold. For instance, the leading role of software architects as main source of NFRs is not reported in most classical requirements engineering textbooks. It becomes crucial to reinforce, drop or contextualise current assumptions in order to formulate well-founded research proposals adapted to the real needs and context of software architects in practice.

There are three main instruments that we may consider.

Empirical studies. Fostering the execution of more empirical studies and surveys is a key instrument to implement this communication improvement goal. The importance of evidence for improving industrial uptake has been reported by many (e.g. [7]). We have experienced in our own work that semi-structured interviews are a very useful tool for learning about current practices: not just the questions predetermined in the underlying questionnaire are investigated, also unexpected observations arise (e.g., we didn't include NFRs validation in the questionnaire but from the first interview we realized how important was and we decided to include this issue) that although not directly integrated in the main results of the study, help to embrace the full landscape and also trigger ideas for further interviews (e.g., about the role of technological products in architectural decisions). As a beneficial side-effect, after one hour or more of interview, respondents learn about some issues which they are not usually exposed to (e.g., vocabulary about NFRs) and become more receptive to the idea of further collaborating in future studies. A word of caution here: threats to validity need to be clearly stated and especially non-sustainable generalizations need to be avoided. So for instance, this has led us not to claim that our observations are applicable in large-size companies.

Dissemination channels. There is a mismatch on the communication channels that both communities favour. On the one hand, most researchers still base the dissemination of their ideas in conventional means like journals, magazines and research conferences. Being this an indispensable resource, especially for long-term impact through systematic literature reviews, it is a fact that a great deal of practitioners use this means sparely, focusing on a few journals they may know well and that not always are the same than those targeted by academics. Therefore, researchers should think of using also more dynamic channels that are increasingly adopted by practitioners. One particular proposal could be to increase the use of blogs to provide fast and synthetic information of the ongoing research (e.g., a recent entry in one of the authors' blog[3] about the use of NFRs in Model-Driven Development got more than 1.000 visits in less than one month and triggered some interesting feedback from practitioners), including some slide presentation and links for getting more details, one of them being a link to the digital library containing the full scientific presentation. Corporate blogs with a mixed academic and professional profile like SEI's Saturn[4] and IASA[5] may be particularly useful. This problem has

[3] http://modeling-languages.com

[4] http://saturnnetwork.wordpress.com

[5] http://www.iasaglobal.org/iasa/default.asp

also been realized by academic conferences that are day by day seeking new means to bridge the academy-industry gap, in the form of Industrial Tracks or even Industrial Days, or other types of imaginative solutions (e.g., see the so-called Empirical Fair at REFSQ conference).

Adoption of standards. Being difficult to push knowledge from academia to industry, this difficulty increases when academia is not able to provide unified, agreed messages. An example is the lack of consensus and the variety of terminology when it comes to NFRs. This hampers something as basic as the ability to effectively communicate knowledge. It is not really easy to overcome this problem. For whatever classification of NFRs we may adhere to, there will be always some deficiency when applied to a particular context or adopted by a particular research group; in particular, most catalogues are neither universal nor complete. Our position is that even being aware of this fact, both the academic and practitioners communities should make the effort of reaching a consensus on appropriate standards and ontologies, either brand new (e.g., as currently sought by the architectural knowledge community with the WICSA Body of Knowledge) or based on existent ones, (e.g., ISO 25000 for classifying NFRs, ISO/IEC 42010 for architectural descriptions), use it as a unified reference framework for research, and promote its use into the practitioners' community.

3.2 Produce knowledge attractive for practitioners

There are several reports about the lack of adoption of academic proposals by practitioners [8]. In fact, academics cannot realistically expect that practitioners will follow a more systematic approach for dealing with NFRs during the development process until the cost-effectiveness and customizability of the proposed methods has not been demonstrated.

Provide cost-effective techniques. In our study, one of the most striking observations was the lack of proper documentation for NFRs. Surely practitioners must not be completely blamed for it, they live in a world of pressure that makes difficult the adoption of those tools that do not provide a short-term return on investment. Academics should work over the results of fundamental research to turn them into methods that may be effectively transferred to the industry.

To improve the cost-effectiveness of academic methods we need to proceed along two different directions. First of all, researchers need to focus on maximizing the benefit that software architects get when adopting academics' prescribed practices. For instance, to convince architects about the benefits of documenting NFR we could adopt new model-driven development (MDD) techniques to (semi)automatically generate different (prototypical) implementations for the software system according to different architectural styles. These alternatives could be quickly evaluated by the software architect to either validate the input NFRs or to test possible implementation scenarios that will help when choosing the final architecture.

Secondly, researchers must understand that technical excellence is not enough to convince practitioners to use a research tool. Non-core technical aspects like a nice user interface (e.g., software architecture visualization tools adapted to the practitioner's perspective [9]), good documentation, quick support and usability tests must be part of

the tool development. For research groups, creating commercial-quality level tools is a big challenge but it is the only way to get a large user base [10].

Provide highly customizable techniques. The population of our study has a significant characteristic: the involved companies do not have a software architect position; instead, technical personnel are appointed as architects to projects depending on several factors. On the contrary, we may find lots of companies that do have a software architect position. This big difference is just an example of the wide variety of situations that one can find when coming to software architecture and to requirements engineering. It is not realistic then to think that methods and techniques can be one-size-fits-all, we need then either to establish very clearly their applicability conditions, or to be highly customizable.

If we refer to customization, situational method engineering [11] may be used with this purpose. Situational method engineering supports the definition of methods by combination of method fragments that can be tuned to specific situations of concrete organizations and projects. Therefore, instead of trying to propose a single new development process that emphasizes the importance of NFRs we could use situational method engineering to provide ad hoc development processes adapted to the needs of each specific organization. The correct application of such a technique implies the identification of those factors that influence the method under development, for instance the size of the organization, the type of project, etc.

This same variety of situations and contexts that software architects must face suggests to rule out a tool to automatically select "the best" software architecture for a given set of NFRs. However, the current situation based on ad-hoc practices that rely on the developers' own skills and experience, is a risky situation for development companies since the quality of their development projects largely depends on the expertise of the software architects in their payroll. We believe we should evolve towards an intermediate scenario in which all important decisions regarding NFRs are still *architect-driven* but *computer-aided*. In this scenario, the tool is a decision-support tool that may suggest possible architectures that can be examined and customized by the software architect depending on the characteristics of the project [12].

In this sense we envision as a long term goal an expert system for software architects. The expert system could interact with the architects to help them analyze the trade-offs of the different alternative solutions, give recommendations and even to point them to missing pieces of information from the domain or the organization in which the system will operate that are needed to take a better decision. The knowledge base for the expert system should be initially set by senior software architects and then should incrementally learn from representative projects (either successful or not) under human supervision.

We took the chance of the survey to add some questions to the questionnaire asking respondents about their position with respect to this issue. We found that 10 respondents were receptive to such a support system although some concerns and advices immediately arose (e.g., how to capture the knowledge, how to represent the blurry concepts that may be behind NFRs).

As intermediate step towards this ambitious goal, we aim at creating a repository of existing architectural styles together with their well-known advantages and limitations

and, if possible, with full descriptions of industrial projects where that style was applied (successfully or not). Software architects could use this repository to learn from their peers and to search for previous solutions to systems similar to the one they are dealing with now. First steps in this direction have resulted in a prototype for managing and applying architectural knowledge [13].

Apart from these action plan oriented to researchers probably a parallel action plan could be identified more targeted to practitioners. Perhaps a nice conclusion of this paper is to challenge practitioners to complement this paper with a statement on what should be done from their software architect's perspective to improve the current state of practice. It would be interesting to compare both action plans and see if we can agree on a common working agenda for the future.

4 Acknowledgements

This work has been partially founded by the Spanish project TIN2010-19130-C02-01.

References

1. Glinz, M.: On Non-Functional Requirements. In: 15th IEEE International Requirements Engineering Conference (RE'07), pp. 21-26, 2007.
2. Ameller, D., Ayala, C.P., Cabot, J., Franch, X.: How do Software Architects consider Non-Functional Requirements: An Exploratory Study. In: 20th IEEE International Requirements Engineering Conference (RE'12), pp. 41-50, 2012.
3. Ameller, D., Ayala, C.P., Cabot, J., Franch, X.: Non-Functional Requirements in Architectural Decision-Making. IEEE Software, in press, 2013.
4. Ameller, D., Franch, X.: How do Software Architects consider Non-Functional Requirements: A Survey. In: 16th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ'10), LNCS 6182, Springer, pp. 276-277, 2012.
5. Anh, N.D., Cruzes D.S., Conradi, R., Höst, M., Franch, X., Ayala, C.P.: Collaborative Resolution of Requirements Mismatches when adopting Open Source Components. In: 18th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ'12), LNCS 7195, pp. 77-93, 2012.
6. Jacobson, I., Meyer, B., Soley, R.: Software Engineering Method and Theory - A Vision Statement. Available at `http://blog.paluno.uni-due.de/semat.org/wp-content/uploads/2012/03/SEMAT-vision.pdf`, last accessed December 2012.
7. Erdogmus, H.: How Important is Evidence Really?. IEEE Software, 27(3), pp. 2-5, 2010.
8. Torchiano, M., Morizio, M.: Overlooked Aspects of COTS-Based Development. IEEE Software, 21(2), pp. 88-93, 2004.
9. Telea, A.C., Voinea, L., Sassenburt, H.: Visual Tools for Software Architecture Understanding: A Stakeholder Perspective. IEEE Software, 27(6), pp. 46-53, 2010.
10. Brunelière, H., Cabot, J., Jouault, F., Tisi, M., Bévizin, J.: Industrialization of Research Tools: the ATL Case. In Proceedings of the 3rd International Workshop on Academic Software Development Tools and Techniques (co-located with ASE), 2010.
11. Brinkkemper, S., Saeki, M., Harmsen, F.: Meta-modelling based Assembly Techniques for Situational Method Engineering. Information Systems, 24(3), pp. 209-228, 1999.

12. Ameller, D., Franch, X., Cabot, J.: Dealing with Non-Functional Requirements in Model-Driven Development. In: 18th IEEE International Requirements Engineering Conference (RE'10), pp. 189-198, 2010.
13. Ameller, D., Collell, O., Franch, X.: Tool Support for NFR-guided Architectural Decision-Making. In: 20th IEEE International Requirements Engineering Conference (RE'12), pp. 315-316, 2012.