Empirical Evaluation of an Approach that Stimulates Architectural Thinking during Requirements Gathering

Preethu Rose Anish
TATA Consultancy Services, India / University of Twente, Netherlands
preethu.rose@tcs.com

Abstract. [Context and Motivation] Requirements specifications often lack the details needed by software architects to make informed architectural decisions. Lacking such details, the architects either make assumptions or go back to business analysts for clarifications or conduct additional stakeholder interviews. This may result in incorrect requirements and project delays. **[Question/problem]** In global software engineering projects, business analysts and software architects are different roles with little communication. **[Principal ideas/results]** The goal of this PhD project is to enhance communication between the two roles by introducing a knowledge base with architectural knowledge to be used by business analysts. Using an empirical approach, we have developed an initial version of such a knowledge base.

1 Introduction

Requirements engineering (RE) activities involve capturing both functional and non-functional requirements of the software system to be developed. The software requirements specifications (SRS) resulting out of these activities often lack the details needed by software architects (SAs) to make informed architectural decisions. In turn, if wrong architectural decisions are made, the intended but unstated requirements will not be satisfied. To compensate, the SAs either make assumptions or go back to the business analysts (BAs) for clarifications or conduct additional stakeholder interviews resulting in project delays. Asking BAs to provide architecturally richer specification may seem like a good idea, but is going to be ineffective, given that BAs lack the technical architectural knowledge needed to ask the kind of questions that extract architectural details from the customer. This is typical in global software engineering and outsourcing projects where communication between BAs and SAs mostly takes place through an SRS and expertise is not shared. This problem has been well acknowledged by other researchers as well [11 - 13]. As a solution to this problem, we have developed an approach [2, 3] that leverages the knowledge of experienced SAs and make it available to BAs to equip them to elicit architecturally richer specification. In this paper, we present the design of a systematic empirical evaluation of our approach. In particular, our goal is to investigate three aspects namely the *ease of use*, *effectiveness* and *relevance* of our approach.

The rest of the paper is structured as follows: Section 2 provides definitions of key concepts. Section 3 presents a summary of the approach. Section 4 provides background on evaluation and its role. Sections 5, 6 and 7 present respectively our research questions, research methodology, plan and initial results. Section 8 concludes the paper.

2 Definitions

As terminology is not uniform across all authors in the field of RE and software architecture, we define some key terms here. We consulted definitions from different sources such as IREB [14] and for the purpose of this PhD research, we use the following working definitions. A requirement is called *architecturally significant* if it has a measurable impact on the architecture of the software system. A *functional requirement* (FR) is a desired behavior triggered by some event or condition change, and delivering some desired output of the system. Any other desired property of the system is called a *non-functional requirement* (NFR). We thus distinguish architecturally significant functional requirements (ASFRs) from architecturally significant non-functional requirements (ASNFRs). While both FRs and NFRs can have an impact on architectural design, for the purpose of scoping this PhD project, we focus on ASFRs only. The questions asked to extract architectural information are called *Probing Questions (PQs)* and PQs when logically sequenced in dialogs are called *PQ-flows* (Probing Question flows). A *Business Analyst* (BA) is the central role responsible for understanding (from the client) the business or functional aspects of the requirements for IT projects. Based on this, the BA is responsible for creating a detailed SRS to be used in the subsequent phases of project development. A *Software Architect* (SA) is a role responsible for converting the business requirements captured by the BA into architecture and design.

3 Brief Summary of the Approach

The earlier years of this PhD work [1-3] were focused on developing an approach to stimulate architectural thinking during requirements gathering. The solution idea is to provide BAs with a knowledge base of ASFR categories and PQ-flows per category to elicit architectural details, plus a tool-supported method that allows SAs to extend the knowledge base with relevant ASFRs. It is worth noting that exhaustive list of requirements and architecture decisions exist in ERP systems but not for the bespoke systems that we focus on. For our kind of systems, we need a dynamic mechanism such as this knowledge base. We acknowledge that both ASFRs and ASNFRs are equally important in this context. However, to scope this PhD research, we focus on ASFRs. We currently have 15 ASFR categories in the knowledge base. Out of the 15 categories, we created PQ-flows for 10 categories. The 10 categories are: *Audit Trail, Batch Processing, Business Process State Alert, Print, Report, Search, Localization/Multilingual, Online Help, Third Party Interaction and Workflow*. These 10 were selected because (a) they occur commonly across systems in many different domains, and (b) they emerged as important topics of architectural significance in our earlier study [2]. Details on the ASFR categories and PQ-flows is published elsewhere [2, 3].

4 Background on Evaluation and its Role for this PhD Project

Drawing on methodological sources in empirical software engineering [5, 8 - 10], the empirical evaluation of our approach is an important research phase in this PhD project. Its role is to gauge (1) the *ease of use*, e*ffectiveness* and *relevance* of the approach, and (2) the generalizability of the approach. In our research design, we first evaluate the *ease of use*, *effectiveness and relevance* from the perspective of practicing BAs. By referring to the *ease of use* concept originally published by FD Davis [14], we measure *ease of use* by gauging how easy it is for the BAs to use the approach as a part of requirements gathering, do they find it easy to adapt to this new way or do they consider this as a paradigm shift that they are not able to relate to. By *effectiveness* of PQs, we intend to investigate the degree to which the PQs are successful in producing a desired result i.e. assist the BAs in unearthing architectural information from the customer during requirements gathering. By *relevance* we mean to investigate whether the BAs find the approach important to their requirement gathering practices and would add value to it. Furthermore, regarding examination of generalizability, we include the perspectives of both BAs and SAs. We need to test generalizability of two things: (1) of the method to fill the knowledge base, and (2) of the ASFRs and PQ-flows in the knowledge base. This includes testing not only the generalizability of the current ASFRs, but also of the ASFRs that will be added by practicing SAs in the future. The central question therefore would be whether our method contains a mechanism by which SAs can test the generalizability of any ASFR or PQ that they add. The target of generalization is the set of all cases in which the communication between BAs and SAs mostly takes place through SRS, and expertise is not shared between them. In other words, we hope that the answer to this is applicable to all such cases, with due allowance made for uncertainty in the answer. For examining generalizability, the strategies described by Wieringa and Daneva [26] would be considered.

5 Research Questions

The overall design goal of this PhD project is to improve the information-content of SRS by means of a knowledge base of ASFRs and PQ-flows that is easy to use by BAs and easy to maintain by SAs. The specific goal of the piece of research presented in this doctoral paper is to validate the proposed approach. Against this backdrop and building upon the discussion in the previous sections, we set out to find answers to the following research questions (RQs):

RQ 1: To what extent does a BA perceive it easy to use the approach?

RQ 1.1: Can the BAs understand the PQs on their own?

RQ 1.2: What kind of effort / training is needed so that the BA can start using the approach on their own?

RQ 2: Can the PQ-flows of the 10 categories help improve architectural relevance of requirements in a SRS?

RQ 2.1: Are all questions in each category architecturally *relevant* (no superfluous questions), and

RQ 2.2: Are all architecturally relevant questions for each category asked?

RQ 2.3: Are all 10 ASFRs architecturally relevant for the system being specified?

We plan to conduct two empirical studies: (1) to answer RQ1 (henceforth referred to as Study 1) and (2) to answer RQ2 (henceforth referred to as Study 2). The two studies, though different in terms of participants and execution style, build upon each other. Each study's research process is organized into three main phases: Design, Execution and Analysis [5]. In the next section, we detail each of the study.

6 Research Methodology and Research Plan

6.1 Research methodology for study 1

Design. We compared the research methodologies that are most relevant to studies in software engineering [5]. We choose a qualitative interview-based evaluation research method by implementing R. Yin's guidelines for case study design [6]. We chose interviews to obtain a detail-rich, holistic and contextualized description from the participants about the approach. The interview technique was selected for two reasons: (1) it is suitable for inquiry like ours, and (2) the resulting data offers a robust alternative [6] to more traditional survey methods. We triangulated the data collected from multiple sources (e.g. participants with varied domain expertise, years of experience, educational background). As we wanted to collect BAs' feedback, we designed our interview study by (1) composing an interview questionnaire to help the participant structure her response (2) testing the questionnaire with an experienced researcher and implement changes to improve it; (3) doing a pilot interview to check the applicability of the questionnaire in a real-life context; (4) carrying out in-depth interviews according to the finalized questionnaire.

Execution. At the time of submitting the paper, this step is work in progress. The 10 ASFR categories were shared with BAs who agreed to participate in the study and they were asked to choose one category that they are most familiar with and one that they are least familiar with. For the two chosen categories, we shared the interview questionnaire and the corresponding PQ-flows. The interview duration was between 30 and 60 minutes. All participants were informed in advance about the research goals and interview process. The interviews were on a one-to-one basis. The questionnaire included three sections designed to collect information about BA's (i) experience and application domain (ii) understanding of ASFRs, and (iii) understanding of PQ-flows.

Analysis. We are using qualitative coding of our data [7], which helps us classify the various reasons as to why BAs perceive a particular category and/or PQs as more difficult or easier than others.

6.2 Research methodology for study 2

Design. We plan to ask volunteer BAs and SAs (5 each) to simulate a process in which requirements are specified by BA and used by SA to design software. We want to observe and analyze simulations in which BAs and SAs use our approach, and use these observations to answer RQ 2. The design would include a volunteer BA and a pseudo-customer who simulates a RE process using our approach, and a volunteer SA who would design an architecture based on the resultant SRS. On the basis of a post-simulation interview with the participants, we will collect their reflections on their experience. Our post-simulation interview questionnaire is developed using the same steps as in Study 1.

Execution. It includes three steps. (1) We provide the participating BA the PQ-flow of a category of her choice along with instructions on how to use it. Based on the outcome of Study 1, it would be decided whether the BA needs to go through some form of training before using the approach. (2) At the meeting between the BA and pseudo-customer, the BA would use the chosen PQ-flow to elicit requirements from the customer and create an SRS. (3) This SRS is given to the participating SA who would use it for designing the architecture of the software system.

Analysis. As we will collect participants' reflections in the form of qualitative data, we will use coding method similar to Study 1. We expect it to yield codes that explain why the approach worked according to the participant or why he would (or would not) use the approach in his next project and what improvements are needed in the approach to make it practically more relevant.

6.3. Threats to Validity

Regarding Study 1, we devised measures to counter the following validity threats [5]: (1) ***Researcher's bias:*** as the researcher is the one who created the PQ-flows, there is an elevated risk of passing bias into data collection and analysis. To reduce this risk, the researcher let the BA select the category to discuss and freely explain the kind of difficulties felt. The researcher took conscious steps to avoid providing any unnecessary information or explanation, except those that the BA asked explicitly. (2) ***Interviewee background:*** BAs could vary in terms of collaboration relationships they established with their respective SAs in a project. Some BAs might be more exposed to SAs' work than others. We think however that this threat is minimal because our participants worked in organizations that have standard project delivery process; where knowledge sharing standards and tools are instrumental in keeping SDLC processes consistent across projects in the same domain.

Regarding Study 2, our biggest concern is that the simulation includes one BA and one SA and the relationship between the two is not known in advance as we rely on volunteers. However, we rely on professional code of conduct and even if the BA and the SA have prior working history, we would ask them to avoid referring to it during the simulation. Following [8], while a simulation in practical settings may be hard to generalize to other context, its key value is in experiencing what in a method works and why it works (or why not). We take this simulation as a pilot and expect the learning from it to be instrumental in improving our approach and its application scenario.

7 Progress

This PhD project has already proposed a solution approach designed to help BAs deliver architecturally richer SRS. Entering the validation phase of this project, we started Study 1. At the time of submitting this paper, we have completed six interviews for study 1. Our very initial reflections on sub-RQs in RQ 1 are as follows:

RQ 1.1: Can the BAs understand the questions in the PQ-flows on their own? The senior BAs (more than 10 years' experience) could understand all the questions on their own. The mid-level BAs (5 to 9 years) needed guidance to understand some questions and junior BAs (less than 5 years' experience) needed relatively more guidance. We found that the domain expertise did not really have any influence on the understanding of the PQs, which indicated that our PQs are generic across business information system domains. Another factor that affected the result was BA's educational background. BAs with a technical background found it easier to understand the PQs than BAs with non-technical background.

RQ 1.2: What kind of effort / training is needed so that the BA can start using the approach on their own? We observed that providing guidance by further elaborating the PQs would improve the understandability. As per our interviewees, such a guidance could take multiple forms: (1) a one hour self-training module for junior BAs, (2) an embedded self-training module in the tool. We await more details to unearth as we progress with the analysis.

8 Conclusion

This PhD project attempted to close the gap between RE and software architecture. It proposes a solution approach [2, 3] that leverages the knowledge of experienced SAs and make it available for the BAs so that they are equipped to elicit an architecturally richer specification. The solution approach detailed in [2, 3] is the key contribution of this PhD project. This doctoral paper is focused on presenting details of the empirical evaluation of our solution approach. The results gained through these evaluation studies would increase our knowledge about our approach and help in improving it further. Our immediate future work includes: (1) finalizing the work on Study 1; (2) include the self-training module in the approach. Our next step will be to execute Study 2.

References

1. P.R. Anish, B. Balasubramaniam, J. Cleland-Huang, R. Wieringa, M. Daneva, S. Ghaisas. Identifying Architecturally Significant Functional Requirements, TwinPeaks, ICSE 2015. IEEE Press, 3-8
2. P.R. Anish, M. Daneva, J. Cleland-Huang, R. Wieringa, S. Ghaisas. What You Ask Is What You Get: Understanding Architecturally Significant Functional Requirements, RE 2015, IEEE Press, 86-95
3. P.R. Anish, B. Balasubramaniam, A. Sainani, J. Cleland-Huang, R. Wieringa, M. Daneva, S. Ghaisas, Probing for Requirements Knowledge to Stimulate Architectural Thinking, ICSE 2016
4. R. Wieringa, Design Science Methodology for Information Systems and Software Engineering, Springer, 2014.
5. R.K. Yin, Case study research, Sage, 2014.
6. Saldaña, J. (2013): The coding manual of qualitative researchers (2. ed.). Los Angeles, London, New Delhi.
7. R. Wieringa, M. Daneva, Six strategies for generalizing software engineering theories. Sci. Comput. Program. 101: 136-152 (2015)
8. V. R. Basili, M. V. Zelkowitz: Empirical studies to build a science of computer science. Commun. ACM 50(11): 33-37 (2007)
9. S. Ghaisas, P. Rose, M. Daneva, K. Sikkel, R. Wieringa: Generalizing by similarity: lessons learnt from industrial case studies. ICSE 2013: 37-42
10. A. Gross, J. Dörr., What do software architects expect from requirements specifications? Results of initial explorative studies, TwinPeaks 2012, pp. 41-45\
11. Z. Li, P. Liang, P. Paris Avgeriou. Application of knowledge-based approaches in software architecture: a systematic mapping study, Information & Software Technology, 55, 2013, pp. 777-794
12. L. Chen, M. Ali Babar, B. Nuseibeh, Characterizing architecturally significant requirements, in IEEE Software, 30(2)2013: 38–45
13. Davis, Fred D., Perceived Usefulness, Perceived Ease Of Use, And User Acceptance of Information Technology, MIS Quarterly; Sep 1989; 13, 3; ABI/INFORM Global pg. 319
14. International Requirement Engineering Board (IREB) Website: https://www.ireb.org/en Last accessed on 09-02-2017