From Architecture to Requirements: Relating Requirements and Architecture for Better Requirements Engineering

Feng Chen
Department of Computer Science and Information Systems, University of Limerick, Ireland
Lero – The Irish Software Engineering Research Centre
feng.chen@ul.ie

***Abstract*—The importance of software requirements is widely acknowledged. However, many software projects still exhibit inadequate Requirements Engineering (RE) practice. More importantly, dealing with Non-Functional Requirements (NFRs) remains as a challenge for software practitioners. This research aims at advancing RE practice through the co-development of requirements and architecture by utilizing the relationship between Architecturally Significant Requirements (ASRs) and Architectural Design Decisions (ADDs).**

***Index Terms*—Architecturally Significant Requirements; Architectural Design Decisions; co-development; relationship**

I. Introduction

"The hardest part of building a software system is deciding what to build … No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later." This quote by Fredrick Brooks nearly thirty years ago has become a truism. Everybody talks how important requirements are; however, requirements practice is often neglected. This neglect is demonstrated by such facts as initial requirements are seldom more than 50% complete for 30 years [1] and about 70-85% of software rework costs are caused by requirements errors [2]. Consequently, academia has targeted research efforts at improving RE practice and has proposed various techniques and methods over the years. Yet these research contributions still haven't visibly improved RE practice in industry. This project is another research effort that aims to improve RE practice.

To improve requirements practice, the role of architecture merits investigation. In 1994 the first international conference on requirements engineering, Garlan [5] pointed out that architectural families induce constraints on requirements in a software product line; and Jackson [5] highlighted that it is difficult to separate requirements from architecture. In 2001, Nuseibeh [19] proposed the twin-peaks model illustrating the co-development of requirements and architecture in an incremental and iterative fashion, which implies that architecture will channel constraining influences back to requirements. But the twin-peaks model provides no detail on how this co-development should be conducted. More recently, Cole [20] emphasized the changing role of architecture in that it can define requirements in a dynamic system-of-systems context. But this change is largely constrained in that particular context and no example or empirical evidence is provided to support that change. Woods and Rozanski [4] argue that architecture can frame, constrain and inspire requirements. Such a three-way relationship from architecture to requirements extends the twin-peaks model with respect to how architecture might influence requirements. But this relationship is too general and high-level to guide the co-development of requirements and architecture. Pohl and Sikora [26] presented the COSMOD-RE (sCenario and gOal based SysteM develOpment methoD) to systematically guide the co-development of system requirements and functional architecture, which emphasizes the use of the proposed architecture solution to validate existing requirements and discover new requirements. But NFRs are not handled in this method, and it lacks a model of the relationship between requirements and architecture necessary to guide their co-development.

Requirements and architecture have a close relationship. To this author's knowledge, research opportunities exist to improve RE practice through the co-development of architectural requirements and architecture by utilizing the relationship between them. What is the nature of this relationship and how one might leverage it to advance RE practice remains a challenge that is the focus of this research.

II. Motivation and Research Questions

Nowadays, real-world software projects are often motivated and driven by non-functional problems, such as slow processing, low scalability, high cost, and dissatisfied customer [3]. These non-functional problems highlight the NFRs, which are critical for the construction of good architecture, in order to achieve high quality software. Hence, requirements engineering and architectural design play a critical role in software development projects.

Very often, requirements elicitation mainly produces functional requirements. Capturing and dealing with NFRs remains a challenge for both academia and practitioners. The software profession has overemphasized functionality at the cost of NFRs [3]. This overemphasis succeeds in obtaining working software in the short term, but fails to achieve sustainability (ability to evolve) in the long term. Furthermore, it incurs a greater amount of technical debts which

978-1-4799-3032-6/14/$31.00 © 2014 IEEE

RE 2014, Karlskrona, Sweden

consequently increase future maintenance costs [9]. Designing better software architecture is key in mitigating against the cost of maintenance efforts. Good architecture promotes software maintainability by exposing the dimensions where change might happen, and hence, make ramifications of change easier to understand and analyze [11].

However, software architecture is not always intentionally designed but emerges over time. One reason for this may be the way NFRs are treated. In 2003 Bass, Clements and Kazman [14] suggested that architects must be actively involved in requirements elicitation early to solicit and understand stakeholders' needs and expectations. However, in 2010, Clements and Bass [12] found that the requirements specification provides very little information needed by architects for architecting. Ameller and Franch [6] found in a survey that most NFRs are discovered solely by architects, but at the same time, architects lack of a shared understanding of NFRs. So how NFRs should be captured and treated remains a challenge, and the architect plays a central role in finding the solutions. The relevant literature [6, 28, 29] focuses on how software architects view, consider and deal with NFRs, but does not mention how they perform RE.

In software architecture, Architectural Design Decisions (ADDs) are the set of design decisions that determine and describe the resulting architecture. Jansen and Bosch [17] even regarded software architecture as a set of ADDs.

In software requirements, Architecturally Significant Requirements (ASRs) refer to those requirements that shape software architecture and drive architectural design. There is a distinction between NFRs that have an impact on architecture and those that don't. For example, some look and feel quality requirements and constraints do not shape the architecture at all, e.g. "Color coding and font size must be used to distinguish a visual element". Meanwhile, some functional requirements do drive architectural design, e.g. "The system shall provide online help", which might imply the establishment of a new component or a new connection between components.

ASRs and ADDs, although not well defined in literature to date, are the most important part of requirements and architecture respectively. Furthermore, de Boer and Vliet [7] argue that ADDs have no fundamental difference from ASRs, and it is supported by Bencomo et al [8]. This close relationship between ASRs and ADDs closes the gap between requirements and architecture. Hence, this research will focus on ASRs and ADDs when looking at the co-development of requirements and architecture.

Based on the problems identified and the knowledge obtained in the previous paragraphs, this research investigates the following questions:

RQ1. With a focus on ASRs, how do software architects perform requirements engineering?

RQ2. What approaches do software architects use to deal with the relationship between ASRs and ADDs?

RQ3. How can we utilize the result of RQ2 to better conduct requirements engineering and architectural design?

RQ1 aims to reveal patterns of the interplay between ASRs and ADDs that arises when architects do requirements engineering, mainly when eliciting and analyzing ASRs. RQ2 necessitates a deeper dive into the interplay between ASRs and ADDs that arises when architects make ADDs with respect to ASRs, and will focus on how architecture and architectural knowledge influence requirements. How ASRs are turned into ADDs and why architects might make changes to proposed architecture solutions will be explored. Real-world problems will be identified that support the co-development of requirements and architecture. RQ3 aims to improve requirements practice by applying the findings from RQ2 to the real world.

The contribution of this research will be a conceptual framework that captures the relationship between requirements and architecture. It will address patterns that depicts how specific types of ASRs could lead to specific types of ADDs, and how specific ADDs could influence (e.g. constrain or inspire) specific ASRs. And this framework might lead to a development model that facilitates the co-development of requirements and architecture with potential time and cost saving benefits for practice.

III. State of The Art

On ASRs. ASRs constitute a large and most important part of requirements; it is the key to promote RE practice. Chen, et al [15] captured four sets of characteristics of ASRs. For example, descriptive characteristics describe ASRs as "tend to be hidden within other requirements" and "situational" etc.; indicators characteristics distinguish ASRs as "targeting tradeoffs" and "assumption breaking" etc. Eeles [13] highlighted reasons why ASRs are often overlooked, such as NFRs are less visible and familiar to most stakeholders, and proposed a conceptual architectural-requirements-questionnaire-based approach to capture ASRs. However, this approach demands extensive human-efforts (stakeholders' involvement) and is time-costly. Cleland-Huang [16] demonstrated a persona driven approach by creating architecturally significant user stories to discover ASRs and evaluate architecture solutions. However, it is hard to guarantee the completeness and accuracy of the ASRs elicited.

On ADDs. ADDs are receiving more attention and popularity now. Most researches concentrate on documenting ADDs to better preserve, manage and reuse architectural knowledge, and proposing models or approaches with regard to ADDs to support architectural design. Kruchten et al [18] described an ontology of ADDs which distinguishes four types of ADDs – existence decisions; bans or non-existence decisions; property decisions; and executive decisions – where attributes of and relationships between ADDs are also articulated.

On relating requirements and architecture. In recent years, there is a surge among researchers to focus on relating software requirements and architecture. The objectives of their researches mainly fall into three categories as follows - our research mainly falls into the last one.

- Facilitating the transition from requirements to architecture. Most research works on relating requirements and architecture fall in this category. For example, CBSP

(Component-Bus-System-Property) provides an approach to refine architectural requirements by leveraging architectural concepts so that requirements statements could be better written to exhibit clearer architectural concerns [21]. It illustrates that a carefully written ASR embodies an early ADD. Dermeval et al [27] described a model which connects requirements (in i*) and architecture (in Acme - an Architecture Description Language [ADL]) through a ADDs metamodel using similar concepts in NFR framework. Such a model enables more precise analysis of how requirements change will influence the architecture through understanding the linkage between them – the ADDs. But this model does not address the relationship from architecture to requirements. Galster et al [22] reviewed and evaluated most transition-focused works, such as rule-based decision making and ADL, and indicated that more fundamental and further research is needed to really understand the relationship between requirements and architecture, and close their gap.

- Proposing models or approaches to promote the co-development and co-evolution of requirements and architecture. The twin-peaks model signals the concurrent and iterative development relationship between requirements and architecture [19]. However, it does not provide a detailed systematic guidance regarding to how this iterative co-development should be developed. Later, Pohl and Sikora proposed the COSMOD-RE method to support the co-development of requirements and functional architecture by providing a systematically structured process. This process addresses activities that should be taken and corresponding artifacts that should be delivered at four different abstraction levels. However, this method mainly focuses on dealing with functional requirements and does not encompass the use of existing architectural knowledge and the relationship between requirements and architecture to better engineer architectural requirements. Pimentel et al [23] utilized i* to represent both requirements and architectural concerns in one single model so that the co-evolution of requirements and architecture can be better managed. However, it provides little explanations and insights on the relationship between requirements and architecture.
- Advancing RE through the influence from architecture. Miller et al [25] conducted a controlled study which found that students with architectural knowledge discovered more architecture-relevant requirements than those students without architectural knowledge, and are more technology-needs focused and less user-needs focused when eliciting requirements. But no empirical study from industry has supported that finding. Schwanke [24] presented an architecture-centered reference process for engineering architectural requirements. This process emphasizes the involvement of architecture concepts and description to ensure the completeness, consistency and validation of architectural requirements by reviewing requirements artifacts against those architecture artifacts. However, this process demands extensive time and human efforts, and thereby may not be sufficiently cost-effective to be fully adopted by industries.

The literature demonstrates that architects influence requirements through their proposed architecture solutions and existing architectural knowledge. However, no detailed systematic theory is developed to elaborate upon the patterns of this influence; and no applicable and time/cost-effective guidance is proposed to advocate practitioners to better engineer requirements by using this influence. Furthermore, very few researches have focused on advancing RE practice through the co-development of requirements and architecture by utilizing the relationship between them.

IV. Proposed Approaches and Progress

This research aims to tackle problems that are not only practical (RE practice in real-world) but also theoretical (relationship between requirements and architecture). To solve these problems, proposed approaches under consideration are outlined in the following steps:

Step 1 – Conduct literature review on how to define, classify, capture and document ASRs and ADDs, and then on the relationship between requirements and architecture. This literature review helps the researcher to identify gaps and resultant research directions that can advance RE practice.

Step 2 – Select and study a software domain to facilitate an in-depth examination on the requirements and architecture of a software system, in order to obtain a deep and thorough understanding of the relationship between them.

Step 3 – Conduct case study for RQ1 and RQ2. Firstly, this case study will investigate how software architects do requirements engineering. How differently software architects perform requirements engineering compared to the way how requirements engineer do it shall be observed. It will reveal what impact the architects' architectural knowledge has on the resulting requirements. More importantly, how architects make tradeoffs on which potential requirement is to keep, refine or discard shall be explored. Comparing the ASRs and early ADDs in each iteration of the development process will contribute to an analysis of the patterns of the interplay between requirements and architecture.

Secondly, this case study will investigate the approaches that architects used to develop ASRs and ADDs. It shall identify problems in their approaches.

Step 4 – Interview architects to obtain their insights into the relationship between requirements and architecture. To fully answer RQ2, the interview data will be analyzed in combination with findings from step 3 and existing literature.

Step 5 – Construct a framework that articulates the relationship between ASRs and ADDs based on findings from step 4.

Step 6 – Synthesize a development model that facilitates the co-development of ASRs and ADDs by utilizing the framework from step 5. Such a model is designed in a way that fully complies with the framework that contains the knowledge of the relationship between ASRs and ADDs. Hence the co-development of requirements and architecture can yield

benefits through the utilization of the relationship. The model shall solve the problems identified in step 3.

Step 7 – Evaluate the framework and the model via a workshop or case study, depending on the scope and nature of the findings. The framework and the model could be applied to a real-life case study or industrial/professional participants in a workshop organized for the purpose who could be given brief tasks which require them to apply the framework/model before being asked for their feedback.

This four-year research project is now in the middle of its second year. The literature review has equipped the researcher with knowledge and directions to further this research. The domain of Contact Center Software Systems (CCSS) has been chosen and studied as the focusing domain. Three criteria guide the selection. Firstly, the kind of software system must be sophisticated and well-established so that the tacit relationship knowledge could be well developed and then embodied in the software system and in its architects' knowledge. Secondly, the selected domain should be accessible in the researcher's environment (Ireland). Lastly, the researcher should be interested in and have some basic understanding of the selected domain. Key functionalities and architecture characteristics of CCSS have been identified. Conducting an empirical case study in a CCSS company is now in progress.

V. CONCLUSIONS

Requirements and architecture have a close relationship. Requirements practice would be better conducted if not isolated from considering architectural concerns, because architecture is a source to validate, evaluate and even discover requirements. In real software projects, requirements engineering is seldom separated from architectural design as delineated by the waterfall software development model. However, the co-development of requirements and architecture is not well supported by current techniques and methods. And the relationship between requirements and architecture is far from clear and at length. This research proposes an approach to improve requirements practice, especially dealing with ASRs. It will address patterns of how architecture influences requirements and how this influence could be embedded in the co-development of requirements and architecture to improve requirements practice.

Future works include studying real projects to investigate the relationship between ASRs and ADDs, interviewing architects to reveal the characteristics of the relationship, and developing a framework and a model to embody this relationship.

ACKNOWLEDGMENT

This research is funded by the Chinese Scholarship Council and also partly supported by Lero – the Irish Software Engineering Centre. I would like to thank my supervisors Dr. Norah Power and J.J.Collins for their support in my research.

REFERENCES

[1] C. Jones. "Positive and negative innovations in software engineering." International Journal of Software Science and Computational Intelligence (IJSSCI) 1.2 (2009): 20-30.

[2] K.E. Wiegers. "Software Requirements 2: Practical techniques for gathering and managing requirements throughout the product development cycle." Redmond: Microsoft Press. 2003.

[3] L. Chung, and J. C. S. d. P. Leite. On Non-Functional Requirements in Software Engineering. Conceptual Modeling: Foundations and Applications. A. T. B. e. al., Springer: 363-379. 2009.

[4] E. Woods, and N. Rozanski. "How software architecture can frame, constrain and inspire system requirements." Relating Software Requirements and Architectures. Springer Berlin Heidelberg, 2011. 333-352.

[5] C. Shekaran, et al.. The Role of Software Architecture in Requirements Engineering. the First Int'l Conf. on Req. Eng. 1994.

[6] D. Ameller, and X. Franch. "How do software architects consider Non-Functional Requirements: a survey." Requirements Engineering: Foundation for Software Quality. Springer Berlin Heidelberg, 2010. 276-277.

[7] R. C. de Boer. and H. van Vliet. "On the similarity between requirements and architecture." Journal of Systems & Software 82(3): 544-550. 2009.

[8] N. Bencomo, P. Grace, and P. Sawyer. "Revisiting the relationship between software architecture and requirements: the case of dynamically adaptive systems." In: Self-Organizing Architectures (SOAR 2009), September 2009, Cambridge, England

[9] N. Brown, et al. "Managing technical debt in software-reliant systems." Proceedings of the FSE/SDP workshop on Future of software engineering research. ACM, 2010.

[10] M.J.C. Sousa, and H.M. Moreira. "A survey on the software maintenance process." IEEE Proc. of Int'l Conf. on Software Maintenance. 1998.

[11] D. Garlan, and D.E. Perry. "Introduction to the special issue on software architecture." IEEE Trans. Software Eng. 21.4 (1995): 269-274.

[12] P. Clements, and L. Bass. Relating business goals to architecturally significant requirements for software systems. TECHNICAL NOTE CMU/SEI-2009-TN-026, Software Engineering Institute, 2009.

[13] P. Eeles. Capturing Architectural Requirements. IBM Rational Developer Works. http://www.ibm.com/developerworks/rational/library/4706.html. 2005.

[14] L. Bass, P. Clements, and R. Kazman. Software architecture in practice. Addison-Wesley Professional, 2003.

[15] L. Chen, et al. (2013). "Characterizing Architecturally Significant Requirements." IEEE Software 30(2): 38-45.

[16] J. Cleland-Huang, A. Czauderna, and E. Keenan. "A persona-based approach for exploring architecturally significant requirements in agile projects." Requirements Engineering: Foundation for Software Quality. Springer Berlin Heidelberg, 2013. 18-33.

[17] A. Jansen, and J. Bosch. "Software architecture as a set of architectural design decisions." In WICSA, pages 109–120, 2005.

[18] P. Kruchten, P. Lago, and H.V. Vliet. "Building up and reasoning about architectural knowledge." Quality of Software Architectures. Springer Berlin Heidelberg, 2006. 43-58.

[19] B. Nuseibeh. (2001). "Weaving Together Requirements and Architectures." Computer 34(3): 115-117.

[20] R. Cole. "The changing role of requirements and architecture in systems engineering." IEEE/SMC International Conference on System of Systems Engineering, 2006.

[21] P. Grunbacher, A. Egyed, and N. Medvidovic. "Reconciling software requirements and architectures: The cbsp approach." Proc. of the 5th IEEE International Symposium on Req. Eng., 2001, 202-211.

[22] M. Galster, A. Eberlein, and M. Moussavi. "Transition from requirements to architecture: A review and future perspective." 7th ACIS Int'l Conf. on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing. SNPD 2006, pp 9-16.

[23] J. Pimentel, et al. "Towards Requirements and Architecture Co-evolution." Advanced Information Systems Engineering Workshops. Springer Berlin Heidelberg, 2012.

[24] R.W. Schwanke. "Architectural Requirements Engineering: Theory vs. Practice." STRAW. 2003.

[25] J.A. Miller, R. Ferrari, and N.H. Madhavji. "Characteristics of new requirements in the presence or absence of an existing system architecture." 17th IEEE Int'l Req. Eng. Conf., 2009.

[26] K. Pohl, and E. Sikora. "The co-development of system requirements and functional architecture." Conceptual Modelling in Information Systems Engineering. Springer Berlin Heidelberg, 2007. 229-246.

[27] D. Dermeval, et al. "On the use of metamodeling for relating requirements and architectural design decisions." Proceedings of the 28th Annual ACM Symposium on Applied Computing. ACM, 2013.

[28] M. Daneva, L. Buglione, A. Herrmann. "Software Architects' Experiences of Quality Requirements: What We Know and What We Do Not Know?," Proceedings of REFSQ, pp. 1–17, 2013.

[29] E. R. Poort, et al. "How architects see non-functional requirements: beware of modifiability." Requirements Engineering: Foundation for Software Quality. Springer Berlin Heidelberg, 2012. 37-51.