IJCSE International Journal of Computer Sciences and Engineering Open Access
Review Paper Vol.-7, Issue-1, Jan 2019 E-ISSN: 2347-2693

Secure Software Architecture: A Hybrid Approach Based On Non-Functional Security Requirements

Jameel Ahmad Qurashi[1*], Sanjay Kumar[2]

[1]School of Computer & System Sciences, Jaipur National University, Jaipur, India
[2]Department of Computer Science & Engineering, Vivekananda Global University, Jaipur, India

[*]*Corresponding Author: jameelqureshi41@gmail.com*

Available online at: www.ijcseonline.org

Accepted: 26/Jan/2019, Published:31/Jan/2019

Abstract In the last decades, software engineering has become an important area of research. As researchers, we try to identify a problem, a need, or a hole in some research topic, once identified we make an effort to produce new techniques, methods, and tools that hopefully will help to improve the detected issue. In the present thesis the identified issue was the need of supporting non-functional requirements in the software architecture design where these requirements are the drivers of the architectural decision-making. This paper demonstrates that a relatively new software engineering discipline, model-driven development, was a good place to propose a solution for the detected issue. We envisioned how non-functional requirements can be integrated in model-driven development and how this integration will impact in the architectural design activities. When we started to produce our techniques, methods, and tools for model-driven development we found out that there was a bigger hole in the web of knowledge than what we had initially foreseen. Much of the evidence of how non-functional requirements affect the software architecture design is hidden. This situation caused a turn in this paper. We needed to understand architects, how they think and how they make the architectural decisions, what is the role of non-functional requirements in the architectural decision-making process, and to what extent are the non-functional requirements important in this process.

Keywords: Software, Architecture, Non-functional Requirements

I. INTRODUCTION

Requirements engineering is the part of software engineering which covers all of the activities involved in eliciting, documenting and maintaining requirements. This paper does not deal with the elicitation of requirements since it is assumed that the requirements are already defined. Some of the contributions may help the documentation and the maintainability of requirements [1]. The principal contribution with regard to software requirements is to make them present and drivers of the subsequent development activities. There are several classifications of requirements, the most habitual one being the differentiation between functionality and non-functionality. For example, a functional requirement could be: the system shall produce an inventory every week; while non-functional requirements could be: the system shall load web pages in less than 2 seconds. A second classification is the differentiation between technical and non-technical requirements.

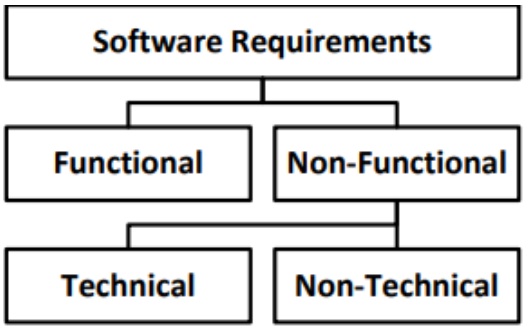

Fig 1 Classifications of software requirements

Functional requirements are by definition technical, but non-functional requirements could be inherent to the software being produced (technical), or could be triggered by external factors such as organizational, laws, software licensing, software providers, in these cases the requirements are classified as non-technical requirements (e.g., the system shall be developed in a language known by the development team). Figure 1 illustrates the explained classification. Non-Functional Requirements Non-Functional Requirements (NFRs) are one of the main research targets in the Requirements Engineering community and their impact has been documented in seminal papers, individual case studies and types of industrial projects. For the present thesis, NFRs are of special relevance.

Software Quality According to M. Glinz, functional and non-functional requirements set the boundary of an important dimension of the software, its quality. For example, in the ISO/IEC 25000, also known as SQuaRE1 quality standard we can find Quality Attributes (QAs) such as: functional suitability, performance efficiency, compatibility, usability, reliability, security, maintainability, and portability. Notice that these quality characteristics are the same that were mentioned in the previous definition of the term NFR. In other words, NFRs and software quality are highly related [2].

Software Architecture

Software architecture is the result of the Architectural Decisions2 (ADs) made during the architecture design. ADs and their recognition as first-class elements are one of the most important advances in software architecture during the last decade. Architectural decisions are the base of Architectural Knowledge (AK). There are many definitions of AK the simplest being: "Architectural Knowledge = Design Decisions + Design".

II. REVIEW OF LITERATURE

Mary Shaw and David Garlan, (2016) [3] we designed the architecture of a SOA monitoring system called SALMon. This architecture was published in the 7th IEEE International Conference on Composition-Based Software Systems (ICCBSS) .The experience gained in Service-Oriented Architecture (SOA) during the design of SALMon has been used many times asexamples of architectural decisions or to identify the key concepts of the Arteon ontology. Later in the same year, Marc Oriol took the lead of the SALMon project, and we made a joint publication with the advances of SALMon during 2008.

Simone Röttger and Steffen Zschaler (2017) [4] the research group (GESSI4) started a new research project about requirements engineering in the context of service oriented systems. In the first year of this project we collaborated in a vision paper of the work plan of this project.

James Robertson and Suzanne Robertson, (2016) [5]in 2016, GESSI started collaboration between the university (UPC) and a software consultancy company (everis). In this collaboration we performed an empirical study about reference architectures. The design on this study was presented in 2013 in the 10th Experimental Software Engineering Track Workshop (ESELAW). In this study we contributed with the experience obtained driving empirical studies to software architects (e.g., helping in the design of the questionnaires and performing the interviews)

Xavier Franch, Angelo Susi, M, (2015) [6]in 2015, GESSI started an European project, RISCOSS5. This project had several tasks related to requirements elicitation and architectural design in which we could use much of the experience obtained during this thesis. As preliminary result of this project we have published a paper with the vision and objectives of RISCOSS.

R.B. Svensson, M. Höst, B. Regnell, (2014) [7]In spite of their acknowledged importance, not so many empirical studies centered on NFRs are available. A recent systematic literature review conducted by Svenssonfound no more than 18 empirical research studies centered on investigating the benefits and limitations on methods around NFRs for five identified areas: elicitation, dependencies, level of quantification, cost estimation, and prioritization. Some findings that are relevant to our aim were: there is no clear view on how to elicit NFRs; quantification of NFRs depends on the market and cost value; different stakeholders may have different views on the importance of NFR types

R.B. Svensson, T. Gorschek, B. Regnell, R. Torkar, A. Shahrokni, R. Feldt, A. Aurum,(2016)[8]The authors of this systematic review themselves have conducted several empirical studies on the topic. In, they focused on the analysis of practices on companies that produce market-driven embedded systems. Svensson et al. targeted several aspects on NFRs in, whilst in they focused on issues related to requirements prioritization. The findings of this last paper suggest that there seems to be a lack of knowledge about managing NFRs in these companies; the authors hypothesize that this could be related to the lower importance given to them with respect to functional requirements (this is a recurrent argument in several studies).Reports a different perception of some NFR aspects depending on the role of the interviewee (e.g., project managers ranked

performance as the most important quality aspect, whilst project leaders ranked usability first), which supports the idea of replicating empirical studies for different role types. Other empirical works from the authors in more general subjects occasionally provided further evidence for NFRs, e.g.,. Borg et al. studied in depth two case studies in two Swedish companies. They interviewed 7 professionals for each case. They reported some common findings in both companies (e.g., vagueness of NFRs and difficulty to test), but also some differences, remarkably in the provenance of requirements, which was different in both cases due to contextual factors. The main conclusion of their study is that although both organizations were aware of the importance of NFRs, still their main focus was on functional requirements. The authors made the hypothesis that methods and tools supporting NFRs throughout the entire development process would be the best way to fight against this situation. Several works focused on the importance of NFR types. In, an e-survey with 31 valid responses was conducted with the purpose of analyzing the importance of the different types of NFRs depending on factors like type and size of project, role of the observer and application domain.

III. RESEARCH OBJECTIVES

1. To know how NFRs and architecture can be integrated in the MDD process
2. To know How do NFRs impact on architectural design
3. To Know Which AK is necessary to make architectural decisions

IV. NFRs IN MDD

The impact of NFRs over software systems design has been widely documented .Consequently, cost-effective software production methods shall provide means to integrate this type of requirements into the development process. The state of practice shows that current MDD approaches do not tackle NFRs satisfactorily and that limits their success and applicability, and hampers their adoption by the industry. In this chapter we analyze this assumption over a particular type of software production paradigm, MDD, and we outline a general framework that smoothly integrates NFRs into the core of the MDD process and provide a detailed comparison among all the MDD approaches considered. In this integration, software architecture emerges with a predominant position. To motivate our findings we use an academic exemplar about the development of a web portal for a travel agency [9].

State of the practice

There are a great variety of MDD-based approaches, many of them following the two-level (PIM and PSM) transformation introduced in the OMG's MDA 2 approach. Among the most popular ones, we find the Executable UML proposals, with as the most popular representative. Executable UML uses a reduced subset of UML that it is directly executable, either using UML interpreters or by providing a direct translation from the models to the final code. Some action is required in this critical situation in order to make the MDD approach more effective, even more considering that this Executable UML method is the basis for the new OMG standard "Semantics of a Foundational Subset for Executable UML Models" that pretends to increase the use of UML in a MDD context.

The adoption of MDD techniques is being slow, even that some industrial studies suggest that they increase the productivity.

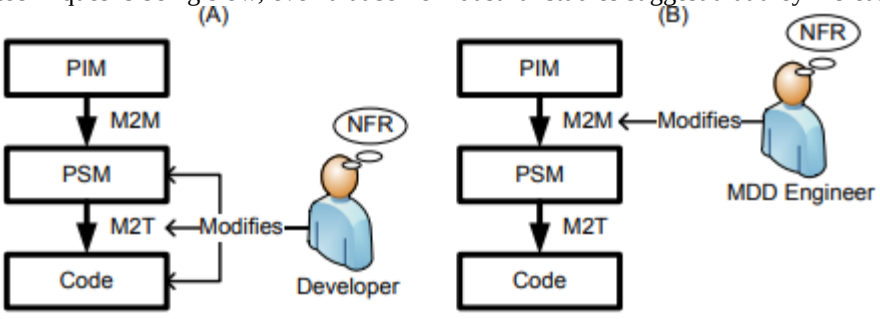

Fig 2 Dealing with NFRs in a classical MDD approach

V. NFRs IN SOFTWARE ARCHITECTURE

As we have seen in the introduction, NFRs express desired qualities of the system to be developed. They refer both to observable qualities such as system performance, availability and dependability, and also to internal characteristics concerning, e.g., maintainability and portability. Other authors use different names, remarkably quality requirement, as a synonymous of NFR, being the diversity of terminology and meaning well-known by the community. Over the years, a common claim made

by software engineers is that it is not feasible to produce a software system that meets stakeholders' needs without taking NFRs into account. As a result, software development projects currently invest a lot into satisfying NFRs. But still it seems to be a lopsided emphasis in the functionality of the system, even though the functionality is not useful or usable when NFRs do not hold. The tight relationship among NFRs and software architectures (SAs) is part of the established body of knowledge in software engineering. As early as in 1994, Kazman and Bass made the point that asserting that SA is intimately connected to the achievement of NFRs should not be controversial [10]. This vision has pervaded over the years and explains why software projects invest a lot into fulfilling NFRs. This influence is mentioned recurrently in the literature: NFRs often influence the system architecture more than functional requirements do; "the rationale behind each architecture decision is mostly about achieving certain NFRs", "business goals and their associated quality attribute requirements strongly influence a system's architecture.

VI. ARCHITECTURAL KNOWLEDGE ONTOLOGIES

Several ontologies have been published to represent AK, each with different nuances, but most of them making a special emphasis on the notion of ADs. One particularly relevant work in this direction is the ontology. The types of ADs (it was previously published in 2014. In this taxonomy ADs are classified into: existence decisions, property decisions, and the executive decisions:

Property decision
"A property decision states an enduring, overarching trait or quality of the system. Property decisions can be design rules or guidelines (when expressed positively) or design constraints (when expressed negatively), as some trait that the system will not exhibit", e.g., "all domain-related classes are defined in the Layer [11].

Existence decision
"An existence decision states that some element / artifact will positively show up, i.e., will exist in the systems' design or implementation", e.g., "the logical view is organized in 3 layers."

Executive decision
"These are the decisions that do not relate directly to the design elements or their qualities, but are driven more by the business environment (financial), and affect the development process (methodological), the people (education and training), the organization, and to a large extend the choices of technologies and tools", e.g., "system is developed using J2EE.

Software architectural design methods
In this section are analyzed some of the Software Architecture Design Methods (SADMs) available in the literature, and more concretely is important for the contents of this thesis how these methods deal with NFRs. One of the principal producers of this type of methods is the Software Engineering Institute (SEI). SEI has created several design and analysis methods: SAAM,ATAM, CBAM, QAWs, QUASAR, ADD, ARID. Documentation for all of them can be found in SEI website. The most relevant ones, in relation to the contents of this thesis, are ADD and ATAM.

Architecture Tradeoff Analysis Method
It is a method to understand the tradeoffs of the architectures of software-intensive systems. This method analyzes the architecture for several quality attributes (e.g., security, performance, etc.). The method guides the design decisions that have an impact on quality attributes. It is a spiral method, consisting in four phases: requirements elicitation including constraints, architectural views, modeling and analysis, and identification of tradeoffs.

Attribute-Driven Design Method
ADD is a method to design the software architecture of a system based on quality goals for the system. The method is extensible for any quality attributes but has been particularly elaborated for the attributes of performance, modifiability, security, reliability, availability and usability. The method considers three architectural views: module view, component and connector view, and deployment view. The method consist in decomposing the system recursively into subsystems and then into components.

We did several searches in academic databases (e.g., Google Scholar, ISI Web of Science, etc.), complemented with other methods that we were already aware of to build a list of SADMs:

Quality Attribute-oriented Software Architecture

It is a method performed in three steps. First, the functional requirements are implemented in components, then the architecture is analyzed to decide whether the NFRs are fulfilled or not, and in the third step the architecture is adapted to be in conformance with the NFRs. The second and third steps are repeated till the whole system is in conformance [12].

Quality-driven Architecture Design and Analysis

It is a set of methods that include a method for selecting an appropriate family architecture approach, a method for quality driven architecture design, a method for evaluating the maturity and quality of the family architecture, and a technique for representing variation points in the family architecture

VII. CONCLUSION

The most valuable outcome of this thesis as a whole is the exploration of different perspectives of the role of NFRs in the software architecture design: in the first part of this thesis, we proposed a way to integrate NFRs in MDD, which made evident the need to support architecture design in MDD; in the second part, we observed how NFRs are understood by architects, and how NFRs are used in practice; and finally, in the third part, we designed Arteon and Quark, an ontology and its companion method, where NFRs are used to drive the architectural decision-making. On the whole, the thesis has served as a way to improve the understanding and the knowledge related to the role of NFRs in software architecture design.

Future Scope

We have better notion of the role of NFRs in software architecture design, we want to produce an implementation of the proposed framework to integrate NFRs in MDD. We have planned collaborations with other researchers in the field to compare our results. For the third empirical study we are planning to produce an extended version with more responses. The research line about how to better integrate NFRs in architectural decision-making process in a way that improves the overall quality of the produced architectures.

REFERENCES

[1]. Lawrence Chung, Kendra Cooper, and Anna Yi. Developing adaptable software architectures using design patterns: an NFR approach. Computer Standards & Interfaces, 25(3):253–260, 2016.
[2]. M. Dal Cin. Extending UML towards a useful OO-language for modeling dependability features. In 9th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems, pages 325–330, 2017.
[3]. Mary Shaw and David Garlan. Software architecture: perspectives on an emerging discipline. Prentice-Hall, Inc, Upper Saddle River, NJ, USA, 2016.
[4]. Simone Röttger and Steffen Zschaler. Model-Driven Development for Non-functional Properties: Refinement Through Model Transformation. In International Conference on the Unified Modeling Language (UML), pages 275–289, 2017
[5]. James Robertson and Suzanne Robertson. Volere. Requirements Specification Template. Edition 15. Technical report, Atlantic Systems Guild, 2016
[6]. Xavier Franch, Angelo Susi, Maria C. Annosi, Claudia Ayala Managing Risk in Open Source Software Adoption. In International Joint Conference on Software Technologies (ICSOFT), 2015.
[7]. R.B. Svensson, M. Höst, B. Regnell, "Managing Quality Requirements: A Systematic Review," EUROMICRO-SEAA 2014.
[8]. R.B. Svensson, T. Gorschek, B. Regnell, R. Torkar, A. Shahrokni, R. Feldt, A. Aurum "Quality Requirements in Practice: An Interview Study in Requirements Engineering for Embedded Systems," REFSQ 2016.
[9]. N.D. Anh, D.S. Cruzes, R. Conradi, M. Höst, X. Franch, C. Ayala, "Collaborative Resolution of Requirements Mismatches when adopting Open Source Components," REFSQ 2017.
[10]. A. Tang, M. Ali Babar, I. Gorton, J. Han, "A Survey of Architecture Design Rationale". Journal of Systems and Software 79, 2016.
[11]. M.A. Babar, L. Bass, I. Gorton, "Factors Influencing Industrial Practices of Software Architecture Evaluation: An Empirical Investigation," QoSA 2017
[12]. A.C. Edmondson, S.E. McManus, "Methodological Fit in Management Field Research," Academy of Management Review 32, 2016.

