An Approach for Capturing and Documenting Architectural Decisions of Reference Architectures

Milena Guessi
Dept. of Computer Systems
University of São Paulo - USP
São Carlos, Brazil
Email: milena@icmc.usp.br

Flavio Oquendo
IRISA Research Institute
University of South Brittany
Vannes, France
Email: flavio.oquendo@irisa.fr

Elisa Yumi Nakagawa
Dept. of Computer Systems
University of São Paulo - USP
São Carlos, Brazil
Email: elisa@icmc.usp.br

Abstract—During the design of software architectures of software systems, it is widely known the relevance of capturing and documenting architectural decisions, i.e., reasons, implications, justification, and trade-offs related to choices made in these architectures. Therefore, it is possible to achieve a more complete documentation of software architectures that also contributes to the success of the software systems. In parallel, reference architectures have become a quite relevant element to the successful construction, evolution, and standardization of software systems. Although architectural decisions are also important for reference architectures, they have not been sufficiently addressed in the context of such architectures. In particular, the use and dissemination of reference architectures may be jeopardized due to lack of appropriate architectural decisions. The main contribution of this study is to present a systematic approach for capturing and documenting architectural decisions of reference architectures. In order to assess the feasibility of our proposal, a case study is presented showing how our approach can be used for enhancing the architecture description of such architectures. With this, we intend to bring attention to the importance of architectural decisions for reference architectures and, as a consequence, to contribute to a more complete documentation, dissemination, and effective use of such architectures.

Keywords—Reference Architecture, Architecture Decision, Architectural Decision, Architecture Description.

I. INTRODUCTION

Software architectures play a determinant role for software systems qualities such as maintainability, dependability, and interoperability, as previously stated by Wasserman [1]. But, most of the success of software architectures relies on its architecture description, which encompasses the set of artifacts expressing the architecture and has several applications, e.g., as basis for system design and development activities, as input to verification and validation automated tools, and as documentation [2].

The documentation of architectural decisions is an important part of architecture descriptions. For example, architecture decisions support the architectural description by recording reasons, implications, justifications, and trade-offs for the most relevant architectural choices. As a result, documenting architectural decisions can add a great value to the software architecture in a medium and long term by preventing knowledge vaporization. Different approaches exist for managing architectural knowledge, e.g., pattern-centric (which aims at establishing a shared vocabulary of reusable, abstract solutions); dynamism-centric (which uses ADLs to codify explicit architectural knowledge into models that can be accessed during runtime); requirements-centric (which intends to bridge the gap between architecture and requirements); and decision-centric (which prevents knowledge vaporization by capturing the reasoning leading to the software architecture) [3].

In the same perspective, we also observe the proposal of reference architectures, which encompass the knowledge about how to design software architectures of a given application domain. To do so, reference architectures address business rules, architectural styles, best practices of software development (e.g., architectural decisions, domain constraints, legislation, and standards), and software elements that support the development of software systems in a given domain [4]. From this, we observe that reference architectures integrate most approaches for architectural knowledge management. Therefore, the documentation of architectural decisions for reference architectures is certainly a prerequisite for their effective role as communication vehicles throughout software systems life cycle. Good examples of reference architectures can be found in the literature, such as AUTOSAR (AUTo motive Open System ARchitecture) [5] for the automotive domain, and UniverSAAAL [6] and Continua [7] for Ambient Assistant Living (AAL) domain.

Even though AUTOSAR is a very complete architecture resulting from several years of research and effort, architectural decisions about the design of the reference architecture are only partially addressed in its documentation, since it lacks rationale and considered trade-offs for example [8]. In fact, to our best knowledge, there is no commonly used approach addressing architectural decisions in reference architectures. If at least the most important decisions were reported, it would be easier to understand why things are as they are in the reference architecture. Moreover, since reference architectures differ from concrete software architectures — e.g., they are more generic and often described at a higher abstraction level, have no clear stakeholders, involve more architectural qualities, and have a larger scope — not all approaches used to describe software architectures are also adequate for reference architectures. In this scenario, several approaches have been proposed for building reference architectures. For example, Muller (2008) [9] provides guidelines for establishing reference architectures. Nakagawa et al. (2014) [10] presents a process to establish, represent, and evaluate reference architectures with focus on improving the separation of concerns in such architectures. Angelov et al. (2012) [11] proposed a classification frame-
work for reference architectures that aims at promoting the analysis and design of reference architectures by assessing their adherence to one of these reference architectures types. Nonetheless, Nakagawa et al. (2012) [12] were the first to introduce the necessity of documenting architecture decisions in the development of reference architectures. However, it is still necessary to determine how this knowledge should be codified and stored in reference architectures, which must be defined by specific methods and architecture frameworks.

In this scenario, the main contribution of this paper is to present an approach for capturing and documenting architectural decisions during the design of reference architectures. We also present a case study in order to show how our approach can contribute to the description of such architectures. As a result, we have observed the importance of capturing architectural decisions during the design of reference architectures as a means to achieve a more complete documentation of such architectures that certainly contributes to their dissemination and effective use. Thus, we also intend this paper motivates further investigations on incorporating an architectural decisions documentation activity throughout reference architectures’ life cycle.

The remainder of this paper is organized as follows. In Section II, we introduce the concept of reference architectures and discuss current approaches for documenting software architectures. In Section III, we present our approach for integrating architectural decisions in the description of reference architectures. In Section IV, we discuss the results from a case study regarding the architectural description of reference architectures. In Section V, we discuss future research perspectives. Finally, we present our final remarks in Section VI.

II. BACKGROUND

Several terms are used to designate software architectures in different abstraction levels. In particular, the term reference model is frequently misused as a synonym for reference architecture. A reference model designates a structure that promotes the understanding on a given domain by sharing a common vocabulary and the parts and its interrelationships without considering implementation details [13]. The OASIS Reference Model for Service Oriented Architectures1 is an example of reference model. Conversely, reference architectures are less abstract than reference models as they provide concrete guidelines for the design of software architectures, such as architectural styles, best practices for software development, and software elements supporting the development of systems. In particular, reference architectures can be derived from the combination of reference models and architecture patterns (see Figure 1). Therefore, different concrete software architectures can be derived from reference architectures.

The architecture description is the main artifact for accessing software architectures. One of the main goals of an architectural description is to present sufficient detail about the software architecture in order to enable its analysis against architectural requirements. Given the relevance of architecture descriptions, the ISO/IEC/IEEE 42010 standard [2] disseminates best practices for creating such artifact. As proposed by


this standard, the usage of multiple architectural viewpoints is a common practice for documenting architectures of software systems. Architectural viewpoints establish the conventions for the construction, interpretation, and use of architectural views framing a particular set of concerns from system stakeholders [2]. As each architectural view conforms to a specific viewpoint, each view provides a particular representation for the system elements and the relationships existing among them [2]. Several viewpoints, such as logical, runtime, data, and physical viewpoints, have been proposed for the description of software architectures [14]. In this context, we can also observe the proposal of architecture frameworks, i.e., a set of viewpoints which establish a common practice for describing the software architecture of a particular domain or stakeholder community, such as “Views and Beyond” [15] and “4+1 Views” [16]. Motivated by the success of architectural viewpoints in the representation of software architectures, architectural viewpoints have also been investigated for the representation of reference architectures [17], [18].

Architectural Description Languages (ADLs) have been proposed for describing software architectures, such as Wright [19] and Rapide [20] which are based on formally defined syntax and semantics. But, even though formal ADLs provide automatic mechanisms for validation and verification of architectural models, they can be difficult for non-technical stakeholders to understand. Therefore, semi-formal languages have been frequently used for describing software architectures as they combine the rigor of formal languages to the understandability of natural languages. For example, UML (Unified Modeling Language) [21] and SysML (System Modeling Language) [22] are semi-formal languages that can be used for describing software architectures. In particular, SysML introduces a Requirements Diagram which captures the hierarchy among requirements and can be useful for guaranteeing traceability among them, and a Parametric Diagram which describes constraints to system property values, such as performance, confidence, or weight, and can be useful for integrating specifications and models of the system into engineering analysis models.

In another context, the main goal of architectural knowledge management is to prevent knowledge vaporization in software architectures. To do so, architecture rationale for significant architectural decisions made in the software architecture should also be included in the architecture description. Significant architectural decisions could be for example the selection of a particular concern or viewpoint, the definition of the most adequate abstraction level for a particular view-
point, or the reason for adopting a particular design pattern. Furthermore, several aspects of an architectural decision can be relevant, such as their implications to the software architecture design, constraints and rules imposed by them, and also the reasoning that lead to them [23]. Therefore, architectural decisions play an important role for education, reuse, and evolution of software architectures as they are used for sharing expertise and best practices. In the context of reference architectures, significant architectural decisions encompass guidelines for deriving the reference architecture into concrete software architectures besides documenting the architectural knowledge of concrete software architectures of a given domain. Thus, reference architectures certainly need to address architectural decisions in their architectural description.

III. INTEGRATING ARCHITECTURAL DECISIONS IN THE DESCRIPTION OF REFERENCE ARCHITECTURES

We included an activity for documenting architectural decisions into the design process of reference architectures defined by Nakagawa et al. (2014) [10]. The first step for designing the reference architecture is to outline its scope which consists in the identification of relevant stakeholders of the reference architecture and their concerns. The scope of the reference architecture guides the selection of a set of viewpoints that frame those concerns. The next activity of this process comprises the creation of selected viewpoints. We propose the documentation of the most relevant architectural decisions in parallel to all design activities. In particular, we propose the creation of a specific repository for architectural decisions which may be directly mapped to other architectural description artifacts (e.g., viewpoints, architecture models). The architecture decisions repository can also be understood as another viewpoint in the reference architecture description. We defined these steps inside a loop to indicate that the reference architecture should be built iteratively. At the end of each iteration, there is an evaluation activity for validating the completeness and/or effectiveness of the architectural description related to the set of concerns defined in the beginning of the loop. This evaluation should be developed by relevant stakeholders and developers of the reference architecture. Moreover, the evaluation approach used depends on the chosen ADL. If this evaluation is not successful, all previous activities should be revised in order to address identified flaws. Finally, a consensus must be reached among the most important stakeholders regarding the necessity of more iterations. Subsequent iterations of this process can include more details, concerns, viewpoints, or stakeholders to the architectural description. This process is described using SPEM in Figure 2.

In our approach, we used the architecture framework for architectural decisions proposed by Heesch et al. (2012) [24]. This framework is composed by four viewpoints, namely: a Decision Detail viewpoint showing information about individual decisions; a Decision Relationship viewpoint showing the relationship between architectural design decisions and their current state in a particular moment in time (e.g., idea, decided, rejected, approved); a Decision Chronology viewpoint showing all versions of an architectural decision, i.e., all states that it assumes over time; and a Decision Stakeholder Involvement viewpoint showing stakeholders’ responsibilities in the decision-making process, which is important to personalize architectural knowledge, i.e., documenting who knows what.

In their study, the authors use UML for creating most of these viewpoints: the decision detail viewpoint only presents a textual representation (detailed in Table I), the design decision relationship and stakeholders involvement apply the use case diagram, and the chronology viewpoint shows design decisions in a state machine diagram using special stereotypes for states and associations. In our approach, we experimented the use of SysML techniques as it is an OMG standard which has also been suggested as an evolution of UML. In particular, SysML has a requirements diagram that can be useful for describing architecture decisions. Moreover, SysML has already been used for describing reference architectures [25]. Therefore, we propose the use of the requirements diagram for complementing the textual representation in the decision detail viewpoint, the block definition diagram and the parametric diagram in the design decisions relationship viewpoint, the state machine.
diagram in the decision chronology viewpoint, and the use case diagram in the decision stakeholders involvement viewpoint.

IV. CASE STUDY

A case study was conducted to evaluate an architectural description for a reference architecture built with our approach. Our purpose with this study was to access the capability of the new architectural description in transmitting knowledge about the reference architecture design. In order to do so, we created an alternative description for the Situated Multiagent Reference Architecture proposed by Weyns (2006) [26]. This reference architecture is composed by:

- one or more Agents, which are autonomous problem solving entities that can dynamically control their own behavior (i.e., perceive and actuate in the environment);
- an Application Environment, which is the medium for agents interaction, such as sharing information and coordinating behavior, that has to be designed for each concrete multiagent system; and
- a Deployment Context, comprised by the hardware, software, and other external resources with which the system interacts (e.g., sensors, actuators, printer, network, database, web service).

In the original architecture description, architectural decisions were presented throughout the reference architecture documentation. In our new architecture description for this reference architecture, we centralized this information in a separate viewpoint, named architecture decisions detail view. We precisely followed the original architectural description in order to guarantee that no errand information was introduced in the new architectural description. As a result, absent information was included in the new architecture description. Table II presents a fragment of the architecture decisions detail view in the new architectural description.

Overall, four subjects participated in our case study. The subjects were separated in two groups and each group analyzed one of the versions of the architecture description. To avoid biasing the evaluation, subjects were not aware of the existence of a second description. The original version of the architecture description is referred to as “original” while the new architecture description created for this case study is referred to as “new”. All subjects received a link to an on-line questionnaire about the reference architecture design including their perception on the architecture description (e.g. facility for accessing a given information, positive and negative aspects of the documentation) and their background on the topics covered in the reference architecture. It is relevant to mention that none of the subjects had prior knowledge on this particular reference architecture and they all spent a couple of hours in their review. In this case study, we focus on the subjects’ overall comprehension about architectural decisions in the reference architecture. In particular, we inquired our subjects about the reasons that motivated the creation of a current knowledge repository. We also asked our subjects to grade from 1 (easy) to 4 (difficult) their difficulty level for answering this question. Table III shows the subjects answers and their perceived difficulty level. Next, we compare subjects’ performance in both architectural descriptions versions. From this table, it is possible to observe that both groups experienced some difficulty in answering this question although there was a small advantage for the original architecture description.

Subjects that analyzed the new architecture description also informed their impressions regarding three aspects of the architecture decision detail view: (i) usefulness, related to the contribution of the view to understand the reference architecture design; (ii) effort, related to the amount of time required to analyze or scan the view for a particular information; and (iii) clarity, related to the amount of time required to understand a particular information in the view. Subjects could grade each aspect from 1 (low) to 4 (high). Table IV presents the subjects answers. We observe that the subjects perception on the usefulness of this view were opposite, but only Subject 1 answered the question correctly. In addition, Subject 1 considered the architectural decisions detail view useful but time consuming, as the whole section needed to be linearly scanned. Since the clarity of this view was poorly evaluated by both subjects, we intend to further investigate the use of SysML and other ADLs in the construction of this view.
To better understand the factors that could have played an important role in this case study, we also asked the subjects about their personal experience on the topics covered in the reference architecture description such as UML, SysML, and Embedded Systems. Table V summarizes subject’s background. We observe that all four subjects present solid knowledge on Software Engineering as three of them are doctorate students and one of them is a post-doctorate student in the field. We also observe that both groups present similar knowledge on most of the topics. In particular, the lack of a solid SysML did not play a determinant role in the analysis of the architecture decision detail view since it presented textual descriptions. However, this factor should certainly be considered in a future evaluation of this view that also uses SysML techniques.

We observed that the group evaluating the new architecture description was more successful since none of the subjects in the other group answered this question correctly. The fact that architecture decisions were scattered in the other views of the original description might explain why subjects evaluating it did not find the correct answer. Therefore, our approach of presenting architecture decisions in a particular repository of the reference architecture seems interesting. Nonetheless, our results also point out that more investigation is certainly needed in order to improve the clarity of architecture decisions. To do so, we will investigate ways for mapping architecture decisions to other elements of the description. Finally, it is important to highlight that this study was supervised by experts who also validated the conclusions drawn from our results.

### V. Brief Discussion and Perspectives of Research

Reference architectures have increasingly importance in the development of a set of software systems while architectural decisions provide the necessary information for deriving the reference architecture. Therefore, architecture decisions should be addressed in parallel to the reference architectures’ creation, maintenance, and evolution for effectively documenting and using the knowledge contained in reference architectures. In spite of the positive results achieved in this work, it is still necessary to conduct additional case studies and/or experiments in order to achieve more evidences on the relevance of our approach. It is also interesting to use our approach in reference architectures of the industry context.

Motivated by the results of our case study and by previous experience in the reference architecture research area, we outline some important research opportunities involving reference architectures and architectural decisions, such as:

- Systematizing the documentation of architectural decisions in reference architectures processes. By taking architecture decisions as an intrinsic part of the development of reference architectures, it will be possible to prevent knowledge vaporization in reference architectures and to create concrete ways for accessing this knowledge in the future. Motivated by Jansen and Bosch (2005) [27] and Kruchten et al. (2009) [28] current processes for creating reference architectures should certainly be revised to add specific guidance in this direction;

- Enabling traceability among architecture decisions and other viewpoints in the architecture description. Architecture decisions may crosscut several viewpoints as they can be related to logical elements (e.g., classes, packages, systems), physical elements (e.g., machines, software installed on these machines, network connections), or functional properties of the reference architecture. In this sense, enabling traceability from architecture decisions to other artifacts in the architectural description provides interesting insights about the reference architecture internals (e.g., dependencies, considered trade-offs, alternatives). To do so, it is necessary to extend architecture decisions with dependencies, constraints, and relationships;

- Adapting ADLs for documenting architecture decisions in reference architectures. Currently, most architecture decisions have been informally described as only a textual explanation is presented and the relationship to other artifacts in the architecture description is indirect. In this sense, the use of formal or semi-formal ADLs can standardize and automatize the documentation of architecture decisions. The automatic support provided by formal ADLs ultimately facilitates the recovery of architecture decisions from knowledge repositories, the reuse of architecture decisions in concrete instances of the reference architecture, and the automatic analysis of their dependencies. Nonetheless, UML and SysML still do not consider architectural decisions as first class entities. Although we used SysML in our case study, it is still necessary to investigate which are the best techniques for codifying architectural decisions into architecture description elements;

- Introducing architecture decisions variability in reference architectures. The variability in reference architectures concerns the ability of a software artifact built from such architectures to be adapted for a specific context in a preplanned manner [30]. In this sense, adding variability to architecture decisions would help to create an even larger knowledge repository as all alternatives, dependencies, and options would be registered in the reference architecture. Moreover, deriving the reference architecture into concrete software architectures could be resumed to the selection of architecture decisions. To do so, it is necessary to investigate in which ways architecture decisions can...
be tailored from reference architectures to concrete software architectures. In this sense, an approach that adds variability to architectural decisions in software architectures, such as the one proposed by Alebrahim and Heisel (2012) [31], could be adapted for reference architectures.

Thus, several interesting opportunities exist regarding architectural decisions in the context of reference architectures. However, each aspect of these topics needs to be investigated in broader and deeper ways; for instance, through qualitative and quantitative evaluations and experimental studies in industry.

VI. CONCLUSIONS

Only through a careful documentation of architectural decisions, it will be possible to have well-documented reference architectures. The main contribution of this study is to propose an approach for documenting architectural decisions in reference architectures. We motivated our proposal with results from a case study. We observed that addressing architectural decisions in the description of reference architectures increases their potential for better communicating and disseminating knowledge contained in these architectures. It is important to say that since reference architectures differ from concrete software architectures, it is still necessary to continue the investigation on this topic. As several research lines need to be addressed yet, we intend to collaborate to the reference community, by providing means for better capturing and documenting architecture decisions in reference architectures.

ACKNOWLEDGMENT

This work is supported by São Paulo Research Foundation (FAPESP) (Grant: 2012/24290-5 and 2011/23316-8), CNPq, and Capes.

REFERENCES