Information and Software Technology 88 (2017) 37–52

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Benefits and drawbacks of software reference architectures: A case study

Silverio Martínez-Fernández[a,b,*], Claudia P. Ayala[b], Xavier Franch[b], Helena Martins Marques[c]

[a] Fraunhofer Institute for Experimental Software Engineering, Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany
[b] Group of Software and Service Engineering (GESSI), Universitat Politècnica de Catalunya (UPC), BarcelonaTech, Jordi Girona, num.1-3, Omega building, Barcelona, Catalonia, Spain
[c] everis, Diagonal 605, Barcelona, Spain

ARTICLE INFO

Article history:
Received 16 December 2016
Revised 9 March 2017
Accepted 21 March 2017
Available online 22 March 2017

Keywords:
Software architecture
Reference architecture
Empirical software engineering
Case study
Benefits
Drawbacks

ABSTRACT

Context: Software Reference Architectures (SRAs) play a fundamental role for organizations whose business greatly depends on the efficient development and maintenance of complex software applications. However, little is known about the real value and risks associated with SRAs in industrial practice.
Objective: To investigate the current industrial practice of SRAs in a single company from the perspective of different stakeholders.
Method: An exploratory case study that investigates the benefits and drawbacks perceived by relevant stakeholders in nine SRAs designed by a multinational software consulting company.
Results: The study shows the perceptions of different stakeholders regarding the benefits and drawbacks of SRAs (e.g., both SRA designers and users agree that they benefit from reduced development costs; on the contrary, only application builders strongly highlighted the extra learning curve as a drawback associated with mastering SRAs). Furthermore, some of the SRA benefits and drawbacks commonly highlighted in the literature were remarkably not mentioned as a benefit of SRAs (e.g., the use of best practices). Likewise, other aspects arose that are not usually discussed in the literature, such as higher time-to-market for applications when their dependencies on the SRA are managed inappropriately.
Conclusions: This study aims to help practitioners and researchers to better understand real SRAs projects and the contexts where these benefits and drawbacks appeared, as well as some SRA improvement strategies. This would contribute to strengthening the evidence regarding SRAs and support practitioners in making better informed decisions about the expected SRA benefits and drawbacks. Furthermore, we make available the instruments used in this study and the anonymized data gathered to motivate others to provide similar evidence to help mature SRA research and practice.

1. Introduction

Today organizations are faced with the development and maintenance of many complex and business-critical software applications. These software applications are developed at multiple locations, by multiple vendors, and across multiple organizations [1]. Despite this diversity, software applications belonging to the same technology or business domain usually share similar architectural needs. As a response to this situation and in order to speed up software development (with the aim of providing guidelines and inspiration for the design of systems) or achieve standardization (aimed at system/component interoperability), organizations often build a central asset called Software Reference Architecture (SRA). An SRA is "a generic architecture for a class of systems that is used as a foundation for the design of concrete architectures from this class" [2]. SRAs provide supporting artifacts (e.g., software elements, guidelines, and documentation) to enable their use, possibly instantiated partially or completely [3]. Therefore, software engineers use SRAs as templates when designing software applications in a particular domain. For example, there are SRAs defined by industry research centers to inspire architecture design and selection of technologies when constructing big data systems [4]; SRAs defined by software and industry leaders to standardize domains like the Internet of Things [5]; and SRAs defined in-house such as the ones of this study.

* Corresponding author at: Fraunhofer IESE, Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany.
E-mail address: Silverio.Martinez@iese.fraunhofer.de (S. Martínez-Fernández).

http://dx.doi.org/10.1016/j.infsof.2017.03.011
0950-5849/

Several potential benefits of using reference architectures have been claimed. A Gartner's report summarizes them as follows: "Reference architectures reduce the complexity of hardware and software architecture by systematically reducing environmental diversity [...], enables greatly increased speed and reduced operational expenses as well as quality improvements due to lowered complexity, greater investment and greater reuse" [6]. Thus, "IT organizations that lack architecture and configuration standards [...] have higher costs and less agility than those with enforced standards" [6]. In addition, some benefits and drawbacks of SRAs have been reported in the literature. However, most of them are not supported by industrial evidence [7]. Therefore, the perspective of academics regarding SRAs and their benefits and drawbacks is not always in line with industry practice, making industrial uptake of SRAs difficult [7]. In order to envisage realistic and effective solutions, more evidence-based research is needed to understand actual industrial SRA practices, their real value, and their risks [8].

Therefore, considering this scenario, our research goal is: *to gather evidence regarding the benefits and drawbacks of SRAs in a company from the perspective of different stakeholders involved in their design and usage*.

With this goal in mind, we conducted a case study at *everis*, a multinational software consulting company with more than 10,000 employees in 12 countries, which became part of NTT Data in 2014.[1] *everis* offers support to their client organizations (large organizations from diverse business domains) to design and use SRAs. Such SRAs foster the development of high-quality software architectures for new software applications. Thus, the *everis* context offered us an adequate setting for investigating our research question.

The case study was conducted in two stages. First, the data related to SRA engineering at *everis* was collected and analyzed [9]. Second, this case study was designed to understand the benefits and drawbacks of nine SRA projects run at client organizations of *everis*. In this paper, we report the results from the second stage.

This case study shows the main benefits and drawbacks of SRAs in the context of client organizations of *everis*. Moreover, the study provides evidence of some improvement strategies suggested by the respondents for dealing with some SRA drawbacks. In summary, these results aim to help practitioners and researchers to better understand real SRA projects and their associated benefits and drawbacks in their corresponding contexts. It aims at providing insights on the relationship between some benefits/drawbacks and their contextual factors. Of course, more research is needed to understand these relationships. However, the study presented here could serve as a basis for generating hypotheses to be tested and for interpreting the results of such tests.

The work presented in this paper is a follow-up of the poster presented at [10]. It extends and improves the results presented there by reporting definitive and complete results, analyzing different stakeholders' visions regarding the benefits and drawbacks of SRAs and their importance, discussing key novel findings, and showing how to use this information to improve the current practice in SRA use.

The paper is structured as follows. Section 2 provides SRAs examples, their mostly theoretical benefits and drawbacks, and a comparison of SRAs with product line architectures. Section 3 shows the industrial context at *everis* and presents the objectives, methodology, and details of this case study. Section 4 presents the results obtained from the study. Section 5 provides an in-depth discussion of the findings by comparing them with previous research. Section 6 discusses the threats to validity. Finally, Section 7 summarizes the conclusions and sketches future work.

2. Software reference architectures

The next three subsections present some examples of the application domains of SRAs, their mostly theoretical benefits and drawbacks, and the differences between SRAs and product line architectures (which are another asset for managing many software systems).

2.1. Application domains of SRAs

The software industry and academic communities have built many SRAs at various levels of abstraction.

First, there are SRAs that target a technological domain (also called platform-specific SRAs [11]). Examples are *The Open Group standard for SOA reference architecture*, which is a blueprint that provides guidelines for adopting a service-oriented approach to information technology [12]; the set of SRAs presented in the Microsoft Application Architecture Guide, which are supported by the Microsoft technology stack [13]; and *the IBM big data reference architecture*, which provides integrated capabilities for the adoption of information governance in the big data landscape [14]. There are also SRAs from academia for solving well-known technological problems (e.g., web browsers [15], and software testing tools [16]).

Second, there are other types of SRAs that focus on a specific business domain (also called industry-specific SRAs). These SRAs can either target many organizations (whose applications share the business domain) or a specific single organization (which aims to standardize or facilitate the development and maintenance of its own applications). An example of an SRA that targets many organizations is AUTOSAR [17], which is being used by many automotive manufacturers and suppliers in order to standardize the software in modern vehicles. An example of an SRA for a single organization is the SRA for NASA's earth science data systems, which facilitates and homogenizes the development of this type of applications [18].

The above-mentioned second type of SRAs target a single domain (i.e., the automotive or aerospace industry), which makes them hard to apply to other domains. In this direction, the goal of some European industrial research programs [19] is to enable their use across disparate domains. Also, the AUTOSAR consortium plans to adapt its SRA for other commercial sectors, such as railway, agriculture, and forestry machinery [17]. This last type of SRAs covering more than one industry is called industry-crosscutting SRAs.

2.2. Benefits and drawbacks of SRAs

There has not been much effort to review, appraise, and compare the benefits and drawbacks of SRAs based on industrial evidence (a notable exception being [7]). Next, we summarize some of the benefits and drawbacks of SRAs asserted in the literature. We identified the following benefits:

- *(B1)* **Standardization** of concrete software architectures by using the SRA as a template for designing a portfolio of applications fulfilling such a standardized design [2,7,8,11,20,21].
- *(B2)* **Facilitation** of the design of concrete software architectures by providing guidelines and inspiration to application builders [2,7,11,20–22].
- *(B3)* **Systematic reuse** of common functionalities and configurations throughout the generation of applications [1,7,11,20,22].
- *(B4)* **Risk reduction** through the use of proven and partly prequalified architectural elements included in the SRA [1,20].
- *(B5)* **Enhanced quality** by facilitating the achievement of software quality aspects already addressed by the SRA [21,22].

[1] *everis*' site: http://www.everis.com/. NTT Data's site: http://www.nttdata.com.