The final publication can be accessed under the DOI: **http://dx.doi.org/10.1016/j.infsof.2017.03.011**

Benefits and Drawbacks of Software Reference Architectures: A Case Study

Silverio Martínez-Fernández[1]
UPC-BarcelonaTech,
Fraunhofer IESE
Spain, Germany

Claudia P. Ayala, Xavier Franch,
UPC-BarcelonaTech
Jordi Girona, 1-3, Barcelona
Spain

Helena Martins Marques
everis
Diagonal 605, Barcelona
Spain

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.

Key Words: software architecture; reference architecture; empirical software engineering; case study; benefits; drawbacks.

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

[1] Corresponding author: Silverio.Martinez@iese.fraunhofer.de
Address: Fraunhofer IESE, Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany

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.

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

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].
- *(B6)* **Interoperability** among different applications and their software components by establishing common mechanisms for information exchange [1,7,11,20].
- *(B7)* **Creation of a knowledge repository** as the SRA inherently acts as a repository of applied knowledge such as architectural and design principles [1,8].
- *(B8)* **Improvement of communication** in the organization and with multiple suppliers because stakeholders share the architectural mindset established in the SRA [1,7].

- *(B9)* **Elaboration of an organization's mission, vision, and strategy,** as designing the SRA might imply reasoning about the organizational goals to be fostered by the SRA [1].
- *(B10)* **Promotion of company-wide best practices** as an SRA provides good practices for the organization, such as prior project artifacts, company standards, design patterns, and commercial frameworks [7,23].
- *(B11)* **Use of the most novel design solutions.** Preliminary SRAs are usually designed to provide innovative design solutions with respect to the existing state of the art [7].

On the other hand, although the benefits of SRAs have seen widespread consideration, their drawbacks have scarcely been documented. Below we describe some drawbacks reported in the literature:

- *(D1)* **The need for an initial investment** to create the reusable assets that compose the SRA [24].
- *(D2)* **Inefficient support for adaptation and instantiation** from the SRA to applications, as SRAs usually lack annotations with attributes and rules describing variability issues [11].
- *(D3)* **Too much abstraction**. The SRA might end up providing an inadequate level of abstraction, leaving the specific choice of specific elements fully open [7].
- *(D4)* **Lack of common interpretation** of SRA, coming from a lack of terminology conventions among different types of stakeholders [1,7].
- *(D5)* **Inadequate documentation** of SRA, which greatly hampers its overall understanding and use by its stakeholders. [7,25].
- *(D6)* **Poor quality** of SRA, mainly in terms of correctness and coverage of the needs of the organization, which hamper its use. SRA quality depends on whether it can be transformed into a meaningful organization-specific architecture [7,11].
- *(D7)* **SRA too specific or limiting**. The SRA specifies the choice from the class of options for each element, which can limit innovation and novel ideas [7,26].

In Section 5, we will review these theoretical benefits and drawbacks of SRAs in the light of the results from our case study.

2.3 Software Reference Architectures Related Concepts

There are several SRA-related concepts, such as architectural styles, architectural patterns, and Product Line Architectures (PLA). Although several studies have described the difference among SRA-related concepts [2,27], it is important to emphasize their similarities and differences here in order to better place and understand the results from this study.

An architectural style focuses on the large-scale of a system, providing a vocabulary of design elements (e.g., pipe and filter, client and server). An architectural pattern is a well-established small-scale solution to a commonly occurring architectural problem. Architectural styles and patterns are domain-independent and abstract. Contrary, SRAs target the definition of functionalities required in a domain and their interaction with the domain environment [2], and provide supporting artifacts to enable their use [3]. Having said that, an SRA can follow an architectural style, and use architectural patterns to achieve desired architectural qualities [28].

The terms SRA and PLA, as well as their underlying concepts, are sometimes used indistinctly in the software engineering literature [21,24]. SRAs and PLAs refer to architecture-centric concepts that aim to capture the high-level architectural design for many software applications. As such, they both describe common architectural elements that can be used in the design of a concrete architecture of an application domain [25]. However, SRA and PLA are different concepts [21,24]. In this respect, Deelstra et al. provide a classification of architecture-centric concepts for the development of many software applications [29]: 1) standardized infrastructure; 2) platform; 3) software product line; 4) configurable product family. Under this classification, Graff

et al. explicitly stated that SRAs are the basis for standardized infrastructures or platforms, while PLAs are the basis for software product lines and configurable product families [30]. We agree with this vision of SRAs, as the results from our study confirm this position because the studied SRAs (see Section 3) referred to standardized infrastructures prescribing functional components, or to platforms providing reusable components with common functionality.

There are mainly three differences between SRAs and PLAs: their purpose, the systems they target, and their supporting artifacts (see Table I).

First, SRAs aim at facilitating or standardizing concrete architectures of a domain, whereas PLAs aim at defining the functional variability of a product family [2,27]. In other words, SRAs are not about an organization, but about a domain, a type of problem, or a type of system, independent of the organizational context. On the other hand, PLAs tend to be for products produced by a single organization.

Second, SRAs provide standardized solutions for a broader domain (i.e., a "spectrum of systems in a technology or application domain" [27]), whereas PLAs provide standardized solutions for a smaller subset of the software systems of a product range domain [21] (i.e., a "group of systems that are part of a product line" [27]).

Third, because they have different purposes, SRAs and PLAs provide different artifacts. SRAs aim at providing guidelines for application development [2], while PLAs provide a more concrete view of the products in a project [25]. On the one hand, SRAs provide artifacts focusing on commonalities [2,21,25], such as common software elements, guidelines, and documentation for the applications of a domain (to give inspiration for the design of systems of an application domain or to achieve interoperability). On the other hand, PLAs provide processes (domain engineering, application engineering), and describe the variability and variants among the products of a software family by specifically and explicitly addressing points of variability and more formal specifications in order to ensure clear and precise behavior specifications at well-specified extension points [2].

Despite these differences between SRA and PLA, some benefits and drawbacks of SRAs from the literature (see Section 2.2) are similar for PLAs [31]. This is due to the main similarity between SRAs and PLAs: commonalities of software systems and their reuse.

Table I. Differences between SRAs and PLAs.

Different characteristics	SRA	PLA
Purpose	Facilitation or standardization of applications from a domain	Formal specifications and defined variability for configurable products of a family
Targeted systems	Spectrum of systems in a technology or application domain	Subset of the whole spectrum of systems of a domain: group of systems that are part of a product line
Support artifacts for	Commonalities	Commonalities and variability

3. RESEARCH METHODOLOGY

This research has its origin in a long-term action-research collaboration between the software consulting company *everis* and the GESSI research group at the UPC. This collaboration aims at supporting the improvement of SRA engineering at *everis* by enhancing the internal reuse of architectural knowledge among their employees, and improving services offered to its clients. To do so, it was important for *everis* to understand and analyze the benefits and drawbacks of the SRAs designed for its clients in order to envisage adequate actions. The next subsections present details of the methodology followed to elicit these benefits and drawbacks.

3.1 Research Setting

everis offers solutions for large organizations from diverse business domains, e.g., banks, insurance companies, public administrations, utilities, and industrial organizations. Such organizations usually manage a wide portfolio of complex and business-critical applications that integrate bespoke software with commercial packages. The complexity of these applications

requires high-quality software architectures. The strategy adopted by *everis* to reach such quality is to foster the adoption of SRAs in the client organizations as a baseline for developing concrete software architectures for new applications. In summary, *everis* supports its client organizations to design and develop their own SRAs, and to build applications on top of such SRAs.

To support the design of SRAs for client organizations, *everis* uses a corporate reference model. A reference model is "a standard decomposition of a known problem into parts that cooperatively solve the problem" [28]. As such, the reference model of *everis* gathers and centralizes the architectural knowledge and practices of the company that have worked in the past. Such a corporate model is continuously shaped to the common business values and services shared by the clients of *everis*. Thus, the *everis* reference model supports reuse of architectural knowledge in different client organizations.

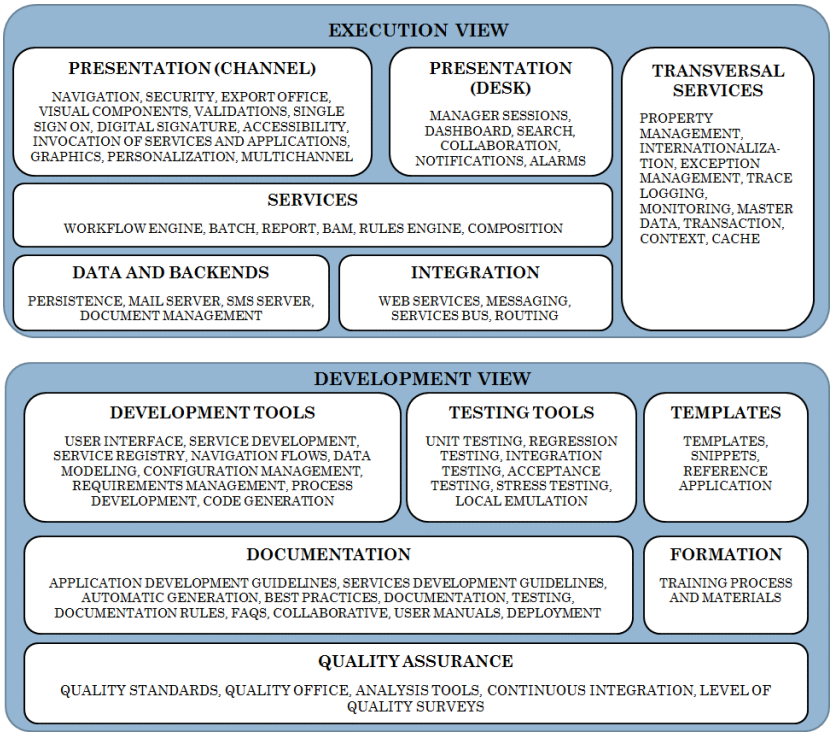

Fig. 1. An excerpt of the *everis* reference model for complex and business-critical information systems.

Fig. 1 shows two different views of the *everis* reference model that is used as a starting point for the design and development of SRAs: the execution and the development views. The execution view describes a generic set of potential modules/functionalities that can be mapped to the SRAs of the clients, depending on their specific needs. The development view describes a wide set of software artifacts (i.e., guidelines, tools) that could be used to support the development of applications. Other views, such as the physical view, are not the responsibility of SRA designers, so they are not shown here.

The *everis* reference model establishes a unique corporate vision about the typical elements that compose complex and business-critical information systems. The reference model is used as a basis for analyzing architecturally significant requirements in order to decide which functionalities should be mapped to the SRA of their clients. For the sake of space and confidentiality, Fig. 2 shows an excerpt of one SRA, which belongs to a public administration (Table III lists all the SRAs from the client organizations of *everis* that were studied). Further information about this SRA project is publicly available at http://canigo.ctti.gencat.cat/canigo/framework/. Fig. 2 shows

that some functionalities of the *everis* reference model were mapped to a Java-based SRA. For instance, we can find the "validations" functionality at the "presentation (channel)" layer, "web services" in the "integration" layer, and four software elements (e.g., i18n and logging) for the "core transversal services". Besides, we can see software components in the SRA that are not considered by the reference model, such as connectors to existing services of the public administration. This shows that each SRA should be personalized for each client because the reference model cannot cover their specific architecturally significant requirements.

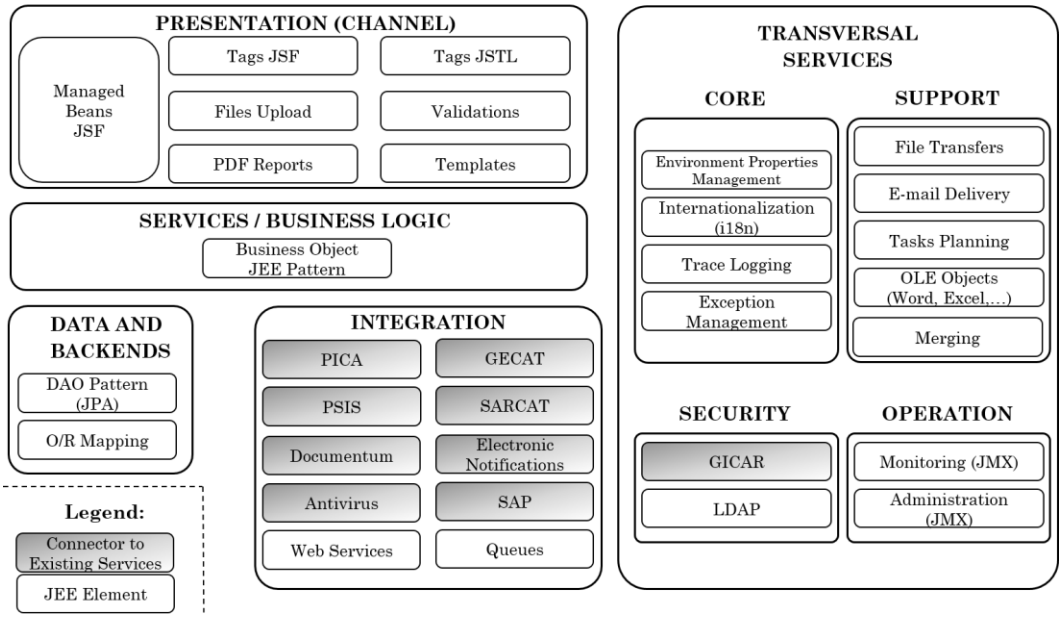

Fig. 2. An excerpt of the execution view of an SRA (called Canigó 3) designed with the help of the *everis* reference model.

There are three types of key stakeholders related to the design and use of SRAs at *everis*:

1. **software architects**, who work together to figure out an SRA based on the *everis* reference model to accomplish the desired quality attributes and architecturally significant requirements of the client organization;
2. **architecture developers**, who are responsible for coding, maintaining, integrating, testing, and documenting the software components and other artifacts of the SRA previously designed by the software architects for the client organization; and
3. **application builders**, who use an SRA by instantiating reusable elements and components developed by architecture developers to build concrete software architectures for the client organization's applications. To do so, the SRA provides them with supporting artifacts such as common reusable software elements, guidelines for the homogeneous development of applications, and documentation describing the logical solution for creating a set of applications [32].

Fig. 3 shows these stakeholders. It shows how software architects and architecture developers are considered **SRA designers**, whereas application builders are **SRA users**. In our context, both SRA designers and users are mainly professionals from *everis*, but sometimes they may also come from the client organization.

3.2 Research Questions

Once we got a substantial understanding of the SRAs at *everis*, we elaborated the Research Questions (RQ) driving this study (see Table II).

RQ1 focuses on the main benefits of SRAs for the client organizations of *everis*. To get an unbiased view from all stakeholders, RQ1.1 analyzes the perception of different stakeholders.

RQ2 focuses on the main drawbacks of SRAs for the client organizations of *everis*. RQ2.1 deepens the perception from different stakeholders. RQ2.2 looks for potential improvements to overcome the drawbacks of SRAs.

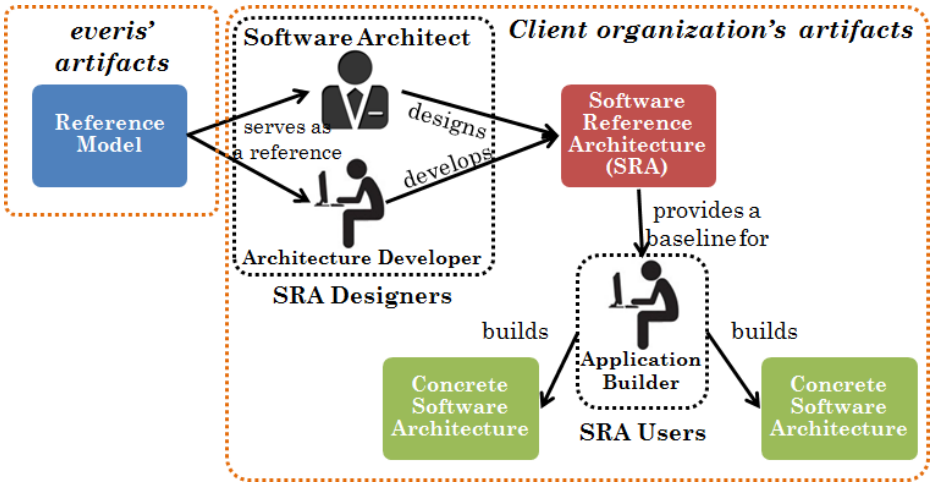

Fig. 3. Key stakeholders in SRA projects at *everis*.

Table II. RQs of our case study.

RQ1	What are the main benefits of SRAs in the context of the client organizations of *everis*?
RQ1.1	What are the main similarities and differences among the stakeholders' perception of such benefits?
RQ2	What are the main drawbacks of SRAs in the context of the client organizations of *everis*?
RQ2.1	What are the main similarities and differences among the stakeholders' perception of such drawbacks?
RQ2.2	What are the potential improvements that stakeholders would be willing to perform to overcome the drawbacks of SRAs?

3.3 Research Design and Sampling

In line with the exploratory nature of our RQs, we decided to use a case study approach to gain a deep understanding of SRA benefits and drawbacks in the context of the client organizations of *everis*. The effort invested in the case study was approximately 21 person-month, considering all its steps (i.e., definition, design, implementation, execution, analysis, and packaging).

We devised a flexible case study protocol from the very beginning, as suggested by [33], to register and update our procedures, instruments, decisions, and deviations. This protocol was devised and agreed on among the researchers and two *everis* managers who participated in the study. These two *everis* managers had extensive experience with SRAs, as they had previously participated in several SRA projects covering all potential roles. Such experience was crucial for tackling this research.

The target population of the study was *everis* SRA projects. The SRAs of these projects fulfilled two criteria: they were created by software architects and architecture developers, and they were used by application builders to build concrete software architectures. For this reason, we targeted these three different stakeholders involved in the SRA projects, as each of them might have different concerns about certain architectural aspects and as this might influence the perceived benefits and drawbacks [20]. We decided to approach several projects in different client organizations as this allowed us to better interpret and assess their goals, and thus the perceived benefits and drawbacks of SRAs in their own context. It would otherwise have been very difficult to interpret certain aspects of the considered benefits and drawbacks in each context.

The two *everis* managers selected nine SRA projects from different client organizations on the basis of their suitability (i.e., SRAs already used) and feasibility to contact least one person playing each of the targeted stakeholder roles. Then the *everis* managers contacted potential participants to get them to agree to participate. We ultimately ended up with 28 people who participated in the selected projects.

Table III summarizes the projects selected for the study. It can be observed that most of the studied SRA projects are from the public sector domain, with banking, insurance, and industry also represented. In all but one of the studied organizations, we had the opportunity to approach stakeholders who covered all the related roles, namely: software architect (SA), architecture developer (AD), and application builder (AB). In two organizations (F and H), we had access to more than one application builder, and in organization G, we could not contact any application builder as some time had passed since the end of the project and none of the application builders who had participated in the project were still at the company. The main goals of the studied SRAs and the applications based on them are also stated in the table. We can observe that most of the stated goals range from improving productivity by reusing components, homogenizing applications and easing the development of applications based on the SRA, via ensuring the fulfillment of certain functionalities and requirements, to enabling the adoption of new technologies by the organization.

Table IV reports the role, education, and experience of the stakeholders who participated in the study.

3.4 Data Collection and Instruments

Once the projects had been selected, *everis* managers provided us the so-called SRA project card containing a summarized description, documentation, and metrics about the effort invested in each project. Whenever we needed clarification, they contacted the corresponding technical project manager or suitable people to handle our questions. We held two informal meetings with *everis* managers to confirm whether the SRA projects and the experience of the participants were suitable for the study.

In order to gather and assess the different perceptions of the targeted roles regarding benefits and drawbacks, we designed and piloted different data collection instruments following the guidelines stated in [33,34] and the corresponding literature background. All the instruments were designed mainly considering the literature about the benefits and drawbacks of SRAs, the practical experience of the two *everis* managers who participated in the study, and the background of our research team. The mapping between the RQs and the corresponding instrument is explicitly stated in Annex A, Section A.1 [3]. It is important to mention that we performed weekly discussion sessions between our research team and the *everis* managers to improve our understanding of the projects' context and to design and polish our instruments. We prioritized the *everis* vision to make a more practitioner-oriented list.

For software architects, we designed semi-structured interviews based on an interview guide as a data gathering instrument (see Annex A, Section A.1). We chose semi-structured interviews mainly because the software architects had a wider vision of the SRA goal and its design, so it was important for us to have the possibility to approach them face-to-face to fully enquire about these details. Semi-structured interviews provided us with the ability to elicit details for each of the analyzed projects while enquiring and understanding their particularities, as follow-up questions were allowed when required. Prior to the interview, we asked each software architect for their personal information (to shorten the meetings) and documentation of the SRA (to prepare the meetings). The interviews were conducted face-to-face by two researchers, in Spanish.

[3] Annex A (A.1, A.2, and A.3) available at http://www.essi.upc.edu/~smartinez/files/ist16-attachment.pdf

Table III. Overview of the selected *everis*' SRA projects.

Id org.	SRA project composition			Main domain	SRA project goal	Applications based on the SRA	Software development methodology	Evolu-tion stage	Approx. effort (hours)
	SA	AD	AB						
A	2	1	3	Industry	To create a minimal SRA to develop homogeneous applications.	Web-based applications to allow vendors to update information about clients in a department store.	Scrum	No	≈5,000
B	3	6	1	Banking	To create an SRA to cover the functionality and requirements needed for the applications.	Multi-platform applications that are fast, satisfy practices of the market and support transaction processing.	Iterative or incremental ("a bit Agile but not formally")	Yes	≈97,000
C	6	3	1	Banking	To provide an SRA and its guidelines to application builders so that they are more productive and can develop applications easier.	Multi-platform applications of a bank with improved usability.	everis' COM (see Table note)	Yes	≈37,000
D	1	10	3	Insurance	To create an SRA that improves productivity and supports new functionalities to migrate applications to new technologies.	Applications that satisfy internal requests for proposals.	Own methodology of the client ("similar to waterfall with many check points")	Yes	≈29,000
E	2	1	1	Public sector	To provide a component-based SRA and its guidelines that supports the development of applications.	Java web applications, with flexible front-end, integration and batch processes.	Agile software development	Yes	≈6,500
F	1	3	2	Public sector	To evolve the existing SRA with new technologies and functionalities.	Web-based applications for the different departments of a public administration.	Agile software development	Yes	≈20,000
G	2	2	1	Public sector	To evolve the existing SRA to standardize the development of applications.	Applications with enhanced reusability and reduced development costs.	Scrum	No	≈4,500
H	3	2	1	Insurance	To create a component-based SRA with the latest technologies that allows reuse in the development of applications.	Applications integrated with the services of an insurance company.	everis' COM (see Table note)	No	≈4,000
I	1	3	1	Public sector	To create an SRA with the latest technologies that support business processes.	Applications that include the business processes of a utility organization.	Lean Six Sigma to define and optimize business processes. Then, waterfall.	No	≈6,500

Note: SA; Software Architect; AD: Architecture Developer; AB: Application Builder.
Note 2: everis' COM (COM (COrporate Methods) is a methodology developed in-house that combines experiences obtained from the delivery of actual projects. More information at:
http://www.everis.com/welcome/Documents/welcome/eng/en_Com.htm

Table IV. Overview of the respondents.

Id. respondent	Id. org.	Education level	Education area	Job position	Role	Years of role experience	Experience with SRAs (Likert)
A-SA	A	Master	Telecommunications	Team/Project Leader	SA	2	5
B-SA	B	Master	Computer Science	Team/Project Leader	SA	3	5
C-SA	C	Master	Computer Science	Team/Project Leader	SA	2	5
D-SA	D	Master	Computer Science	Senior Analyst	SA	1	5
E-SA	E	Master	Computer Science	Team/Project Leader	SA	1	5
F-SA	F	Master	Computer Science	Team/Project Leader	SA	3	5
G-SA	G	Master	Electronics and Telecommunications	Team/Project Leader	SA	10	5
H-SA	H	Master	Telecommunications	Team/Project Leader	SA	1	5
I-SA	I	Master	Telecommunications	Team/Project Leader	SA	3	5
A-AD	A	Master	Computer Science	Program Analyst	AD	1	3
B-AD	B	Master	Computer Science	Program Analyst	AD	0.5	1
C-AD	C	Master	Computer Science	Senior Analyst	AD	3	4
D-AD	D	Master	Computer Science	Senior Analyst	AD	0	3
E-AD	E	Master	Computer Science	Team/Project Leader	AD	1	4
F-AD	F	Bachelor	Computer Science	Senior Analyst	AD	2	3
G-AD	G	Master	Computer Science	Senior Analyst	AD	2	4
H-AD	H	Bachelor	Computer Science	Senior Analyst	AD	1	3
I-AD	I	Bachelor	Computer Science	Team/Project Leader	AD	1	3
A-AB	A	Superior Technician	Computer Science	Junior Programmer	AB	0	1
B-AB	B	Master	Computer Science	Junior Programmer	AB	0	2
C-AB	C	Master	Computer Science	Team/Project Leader	AB	1	1
D-AB	D	Bachelor	Telecommunications	Program Analyst	AB	1	4
E-AB	E	Master	Computer Science	Junior Programmer	AB	1	3
F-AB1	F	Superior Technician	Computer Science	Junior Programmer	AB	2.25	4
F-AB2	F	Bachelor	Computer Science	Junior Programmer	AB	1	3
H-AB1	H	Bachelor	Computer Science	Program Analyst	AB	0	1
H-AB2	H	Master	Computer Science	Junior Programmer	AB	0	1
I-AB	I	Master	Computer Science	Senior Analyst	AB	1.5	1

Note: SA; Software Architect; AD: Architecture Developer; AB: Application Builder.

Each interview took about one hour and was audiotaped and prepared for analysis by manually transcribing the audio records into text documents (this was done by an external company and reviewed by the researchers).

To gather information about the SRA vision from architecture developers and application builders, we used online questionnaires to gather data. The reason for this decision was that they used to work at different locations depending on the client organizations, some of which are located in other cities or even different countries. Furthermore, given the differences in the SRA-related responsibilities associated with architecture developers and application builders, we decided to design different online questionnaires for each role. For architecture developers, the questions of the online questionnaire mainly focused on aspects related to coding, maintaining, and documenting all the artifacts created to operationalize the SRA. For application builders, their questionnaire focused on the use of the SRA for building concrete software architectures in the clients' contexts. The resulting online questionnaires mostly included closed questions. The lists of possible answers for the closed questions was based on our aforementioned discussion meetings with *everis* managers, the benefits and drawbacks from the literature studied in Section 2.2, and the responses of the software architects in their face-to-face interviews. For instance, after the face-to-face interviews with the software architects, we added as a possible benefit the use of the latest technologies. To partially mitigate the rigidness of closed questions in the online questionnaires, we also included open questions to enable the interviewees to add any comment or observation about the closed questions therein.

To enable architecture developers and application builders of the assessed projects to fill in the questionnaire, we prepared an invitation e-mail that was sent through the *everis*' managers with cc to us. Their project leaders had previously agreed that they could spend some time on this activity. We gave them a period of two weeks to complete their answers. After this invitation e-mail, we got all responses on time. The data gathered via the online questionnaires was automatically prepared for subsequent analysis using the functionalities of LimeSurvey.

3.5 Data Analysis

To perform the data analysis, the research team held several discussion meetings during and after the data collection and established specific protocols and templates for the data analysis.

It is important to mention that given the diversity of the instruments used to gather data, we provide more detailed information from software architects than from architecture developers and application builders. This is due to the fact that information from software architects was gathered through face-to-face interviews instead of through online questionnaires.

To process the data gathered from the interviews with the software architects, we used an Excel-based template to organize each participant's answer to each question. To do so, we used the interview transcripts and individual notes taken by the researchers during the interviews. The approach followed for processing the open questions from the interviews was a tailored thematic analysis as suggested in [35] for case-study synthesis. It consisted of the following steps:

1) Extracting data from the original interviews and individual notes.

2) Grouping the data into fundamental groups based on the questions of the interview guide.

3) Identifying and coding interesting concepts and findings from each group.

4) Translating codes into categories. The template used to gather and process the information of the devised categories included the following columns: the identifier of the category, a detailed description of the category and the cases included there, the participant, and explicit sentences from the interview that support the category.

5) Discussing the codes and categories and linking relevant categories together.

Two researchers performed steps 1 and 2. Two members of the research team performed step 3 individually. The resulting codes were discussed between these two members and were translated into categories in step 4.

The aim of step 5 was to discuss the codes and categories identified in the previous steps with the rest of the team in order to ensure correct interpretation of each category and the supporting evidence, and to find potential associations among them. This activity was supported by the use of the software tool Weka[4] to generate and visualize clusters, which helped us to better relate and interpret our qualitative data. Cluster analysis is an explorative analysis that tries to identify structures within the data in order to identify characteristics and homogenous groups of cases [36]. We used the simplest cluster analysis algorithm, named simple k-means, to identify groups that were subsequently assessed and discussed to confirm their meaningfulness. Details of the cluster analysis we performed are provided in Annex A, Section A.3. Furthermore, the raw data is available in an Excel file (see Annex B[5]) so that the interested reader can corroborate our observations, further assess the data, or create new clusters at their convenience. In order to be exhaustive with the analysis of the gathered evidence, we first discussed our findings with respect to the contexts of the projects to understand the contextual influence. Afterwards, we analyzed the evidence with respect to the role of the participants in the SRA projects. This led us to a better interpretation of the contextual factors of our results and thus improved our understanding when it came to devising the categories. Consequently, our discussion led us to split, modify, discard, or add categories to ensure that all answers and their contexts were represented well. We tried to be thorough with the codes and categories in order to include as much detail provided by the respondents as possible. Processing the answers of each question to envisage their categories had its own peculiarities, which will be summarized in the context of the description of the results.

In the case of the online questionnaires, the closed questions were easy to process as we took each option given in the questionnaire as a category. These categories were automatically reported by the Lime Survey tool. However, to process the answers to the open questions, we assessed each answer in the context of its corresponding question to analyze its effect on the existing categories (i.e., those from the options given in the questionnaire).

To provide a global understanding of the benefits and drawbacks of SRAs (as shown in the Results section), we also used Weka to assess all categories gathered from each role's instrument, and then proceeded to assess the results by role.

It is important to emphasize that, in line with the qualitative nature of our approach, the generated categories were intended to give us a way to describe our findings rather than to provide a quantitative vision of the *everis* context.

4. RESULTS

This section details the main findings from the case study. The following elements are used to report these findings:

1. non-mutually exclusive categories created from the analysis of all stakeholders' responses as indicated in the data analysis section;
2. representative quotes of these categories from software architects' responses, indicating their project in square brackets.
3. tables and bubble charts, respectively, showing the frequency with which various stakeholders mentioned each category. They indicate the most popular SRA benefits and drawbacks, and show how perception differ among SRA designers and users.

Moreover, we provide further details about the categories, the stakeholders' representative quotes, and cluster analysis to facilitate comprehension of different contextual aspects (such as respondent experience and different application domains) in an additional document, available in Annex A. Furthermore, the raw data is available in Annex B.

[4] http://www.cs.waikato.ac.nz/ml/weka/

[5] Annex B available at http://www.essi.upc.edu/~smartinez/files/ist16-data.xlsx

4.1 RQ1: Benefits from using SRAs in *everis'* client organizations

In this subsection, we report the resulting categories of SRA benefits, encompassing the perception of all stakeholders. Table V shows representative quotes of these categories and the details of how many respondents of each type considered each of the benefits. The categories of benefits mentioned were:

- (*Ben-A*) **Reduced development costs**. 23 of the 28 participants emphasized the perception that SRAs reduce development effort and costs by enabling the reuse of common assets, facilitating functionality, and speeding up application development. Regarding this, some respondents also commented on the appropriateness of investing time into common software elements that would be reused, such as cross-cutting elements (e.g., persistence and logging modules) that appear in all applications and are time-consuming without an SRA.
- (*Ben-B*) **Improved maintainability and reduced maintenance costs**. 22 of the 28 participants perceived improved maintainability and understandability of applications derived from SRAs mainly because of the modularity of the SRAs.
- (*Ben-C*) **Easier application development and increased productivity of application builders.** 21 of the 28 participants stated that the SRA artifacts make it easier for them to build applications because artifacts abstract them from most technical problems (e.g., communication, back-ends…). This is because architecturally significant requirements were already addressed by the SRA artifacts, facilitating the development of applications.
- (*Ben-D*) **Incorporation of state-of-the-art technologies**. 15 of the 28 participants agreed that SRAs were being used as a way to foster the use of the latest technologies in the applications of their organization. Among other things, using the latest technologies instead of older ones facilitates the recruitment of professionals with the required technological skills.
- (*Ben-E*) **Alignment with business needs**. 12 of the 28 participants mentioned that the design of the SRA inherently considers important organizational business processes, so that the applications that are based on it are better aligned with business needs, e.g., by supporting the particular workflow in a process.
- (*Ben-F*) **Homogenization of the development and maintenance of a portfolio of applications**. 9 of the 28 participants believed that the standardization promoted by SRAs implies higher control over what is being done (supporting distributed teams in different locations) and helps to create a corporate style for all applications.
- (*Ben-G*) **Increased reliability of SRA software elements that are common for a set of applications**. 9 of the 28 participants stated that the SRA elements have been tested and matured. As a consequence, common SRA elements have fewer errors.
- (*Ben-H*) **Others benefits**. In this category, we include some specific benefits not mentioned by any stakeholder type more than once. Software architects indicated: application of best practices; easy distribution of the SRA through the web; support for application builders in case of problems. Architecture developers indicated: improved decision-making; reduced license costs; ability to incorporate more functionality into applications. Application builders indicated: improved agility when requirements are changed; improved decision-making; good documentation of SRAs.

As we can see in Table V, about 80% of the participants agreed that client organizations mainly benefit from reduced development costs (Ben-A) and improved maintainability (Ben-B). By means of the interviews done with software architects, we got useful details for understanding how a reduction of development costs is achieved. The reasons include: reuse of software elements, such as cross-cutting modules and services that implement business logic (in 5 out of 9 projects); agile and automated development (4); improved configuration of software elements, e.g., fast set-up of the modules to be reused by the application (3); and technological and architectural decisions were already taken, i.e., time is saved during the architecture design of every new application and reliability is improved (2). However, although they provided qualitative

explanations regarding why the development costs were reduced, they did not provide quantitative estimates to those reductions. Still, some software architects commissioned us to make such estimation. Therefore, we conducted another case study in a client organization [24].

Table V. Quotes from respondents about the benefits of SRA use.

Code	Representative quotes from software architects	# SA	# AD	# AB	% Total
Ben-A	"If the developers need to use common software, they know that the SRA offers software elements that facilitate functionality and speed up the process" [F]. "The cost of developing a new application is lower when it is based on the architecture" [A].	7	8	8	82%
Ben-B	"The cost of maintaining an application based on the SRA is lower because SRA-based applications are more comprehensible and easier to evolve and maintain" [A].	5	7	10	78%
Ben-C	"The SRA abstracts you from the most technical problems" [B]. "SRA improves productivity in the development of applications" [C].	7	5	7	68%
Ben-D	"Technological updates facilitate the recruitment of professionals" [H].	3	5	7	53%
Ben-E	"The business process of reviewing records was dramatically improved" [I].	3	4	5	43%
Ben-F	"The SRA offers procedures and a methodology about how to make applications" [C]. "Homogeneity helps to have a distributed team in different locations" [E].	6	2	1	33%
Ben-G	"SRA software elements have been tested and matured, which implies reliability" [F].	5	3	1	33%
Ben-H	"We used good practices like remove 'dead code' and wrappers of the SRA" [F]. "The SRA can be found on the Web, which saves distribution costs" [H]. "If the application builder has problems, there is a support team that helps them and solves problems if necessary" [F].	2	3	2	25%

Note: SA; Software Architect; AD: Architecture Developer; AB: Application Builder.

To a lower extent, but still with strong support (68% and 53%, respectively), the participants also mentioned easier development (Ben-C) and incorporation of the latest technologies (Ben-D) as relevant benefits.

Next, we report the different stakeholders' perception regarding these benefits.

4.1.1 RQ1.1: Stakeholders' perception of the benefits of using SRAs in client organizations.

In order to show the different perception of stakeholders about the benefits of using SRA in client organizations, we graphically report such benefits as a bubble chart in Fig. 4. The X-axis contains the frequency in which SRA designers (i.e., software architects and architecture developers) mentioned the benefits whereas the Y-axis represents the same frequency for application builders. We divided Fig. 4 into four quadrants. The bubbles contained in the up-right side represent relevant aspects for SRA designers and users. The bubbles included in the up-left and down-right side are only important for SRA users and designers respectively. Finally, the bubbles contained in the down-left side were not strongly worded by neither designers nor users. The size of the bubble corresponds to the overall percentage of stakeholders that mentioned it.

Both SRA designers and users agree that they benefit from reduced development costs (Ben-A) and easier development and maintenance of applications (Ben-C). These two benefits were mentioned by a similar percentage of SRA designers and users.

However, there was less agreement among the stakeholders about other benefits.

On the one hand, application builders were more concerned than SRA designers about improved maintainability and reduced maintenance costs (Ben-B) because they are responsible for evolving the applications and benefit from better understandability of SRA-based applications. Application builders seem to be more concerned about the use of the latest technologies (Ben-D), since they use the selected technology stack on a daily basis. Moreover, application builders considered the fact that applications are better aligned with business needs (Ben-E) as a more relevant benefit compared to the other two roles. It is important to mention that this benefit appeared in SRA projects that allow modeling and executing business processes (C, H, I projects).

The reason we posit is that application builders focus on the domain of a client organization and have greater knowledge of its business processes.

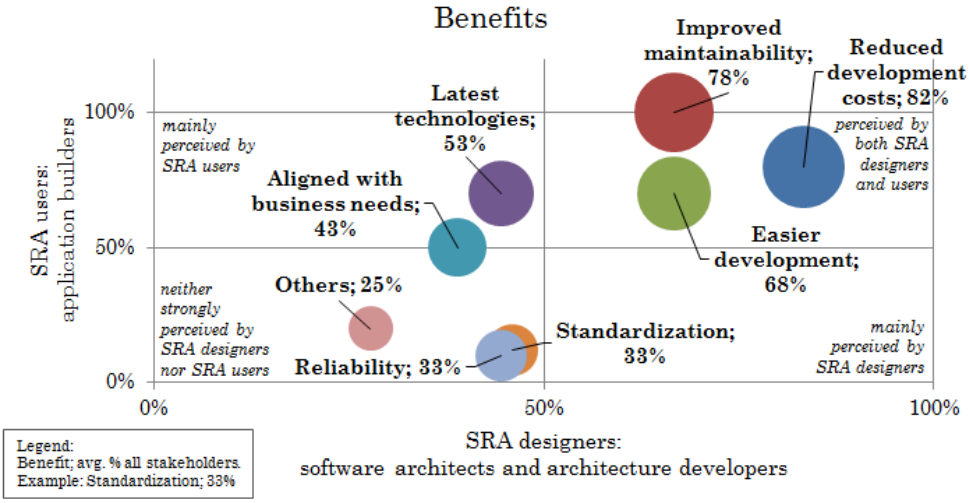

Fig. 4. Comparison of SRA benefits between SRA designers and SRA users.

On the other hand, SRA designers disagreed about the standardization and reliability of SRAs as a benefit (see Table V). These two last benefits received more attention by far from software architects (6 of them mentioned standardization and 5 reliability) than from architecture developers (2 of them mentioned standardization and 3 reliability). The reason may be that, even if both types of stakeholders are SRA designers, software architects have a more global vision of the whole project and have more experience in other SRA projects.

4.2 RQ2: Drawbacks and risks from using SRAs in *everis'* client organizations

The three types of stakeholders mentioned the following drawbacks (see Table VI):

- *(Dra-A)* **Additional high or medium learning curve for using the SRA**. Application builders need to learn how to develop and maintain applications with an SRA. Even though SRAs may be based on standards and de-facto technologies, there are extra features that need to be mastered.
- *(Dra-B)* **Limited creativity by giving regulative guidelines to develop applications**. "Rare" applications will seldom be developed since SRAs standardize developments. For instance, if the SRA supports specific technologies, application builders are only allowed to use such technologies.
- *(Dra-C)* **Dependency of applications on the SRA**. When applications have requirements that the SRA does not offer yet, their development is stopped until the SRA implements these requirements.
- *(Dra-D)* **Complexity**. Architecture developers and application builders mentioned that the use of the SRA might be complex, especially when it grows.
- *(Dra-E)* **None**. Some of the responders indicated that the adoption of SRAs does not present any drawback for them.
- *(Dra-F)* **Wrong decisions about the technologies to be used in all the applications** (e.g., adopting technologies not mature enough to be productive).
- *(Dra-G)* **Other drawbacks**. In this category, we include the drawbacks that were mentioned only once by any type of stakeholder. Software architects indicated: difficulty to measure time-to-market reduction due to the SRA; time-to-market ultimately depends on the skills of

the application builder; initial investment in the SRA. Architecture developers indicated: SRA maintenance. Application developers indicated: use of old technologies; conflicts between technologies.

Table VI. Quotes from respondents about the drawbacks of SRA use.

Code	Representative quote from software architects	# SA	# AD	# AB	% Total
Dra-A	"The SRA is very specific. Although it is based on standards and de-facto technologies, there are extra features to be learnt" [B]. "Although the organization had experts in Oracle Forms, they did not have knowledge about developing Java applications." [H]	5	4	9	63%
Dra-B	"The architecture restricts the development, and indicates how to do it. Thus, some developers would prefer not to use some part." [A]	2	1	5	28%
Dra-C	"When applications have requirements that the SRA does not offer yet, there are dependencies. Until the SRA will satisfy them, application development is blocked" [C].	4	2	0	22%
Dra-D	There is no representative quote because this category came up from the online questionnaires.	0	2	2	14%
Dra-E	There is no representative quote because this category came up from the online questionnaires.	0	1	3	14%
Dra-F	"The ESB used was not mature enough to be productive" [G].	2	0	0	7%
Dra-G	"The SRA allows having the structure of the application and a few screens working in one day, but it always depends on the application builders and the business logic that they put inside" [F].	2	2	2	21%

The most frequently mentioned drawback of using an SRA, mentioned by 63% of the participants, was that application builders need time to attend training courses and learn how to use the SRA (Dra-A). Software architects mentioned in the interviews that they usually provide training sessions. Some of the artifacts used in these sessions are: user manuals or documentation about how to use the SRA (in 6 out of 9 projects); practical workshops for application builders (6); training sessions and follow-up meetings for the project managers of the client organization (5); description document of the architecture (2); a wiki with material (e.g., how-to guides, configuration files) to support application builders (2); support office and service (2); and continuous training when there were also SRA designers from the client organization (2).

To a lesser extent, the second and third most popular drawbacks, respectively, were: limiting application builders (Dra-B), mentioned by 28% of the participants; and dependencies on the SRA (Dra-C), mentioned by 22% of the participants.

Besides analyzing the main risks and limitations when using SRAs, we report the different visions of the stakeholders below.

4.2.1. RQ2.1: Stakeholders' perception of the drawbacks of SRA use for client organizations.

The findings about drawbacks clearly reflect the daily work of each role (see Fig. 5). Software architects are more worried about decisions they make about technologies (Dra-F) and about offering common SRA software elements to application builders as soon as possible, so that they do not block them (Dra-C). On the other hand, application builders are more worried about the learning curve (Dra-A) and the restriction of having to follow SRA standards and procedures that force them how to do things (Dra-B).

Surprisingly, only 14% of the participants indicated that the use of an SRA is complex (Dra-D), whereas 63% mentioned a high learning curve. No software architect mentioned the complexity of an SRA as a drawback. The main reasons could be that they think that SRAs facilitate the development of applications and that application builders just need time to learn the extra features of an SRA, which from their point of view may be time-consuming but not complex.

One architecture developer and three application builders indicated that the use of SRAs does not have any drawback (Dra-E). It could be important to mention that these three application builders stated that they had "no experience with SRAs" before that project, so they were not yet experienced then. Although the architecture developer had "medium experience with SRAs", his SRA project was in an early phase at the time we did the interview.

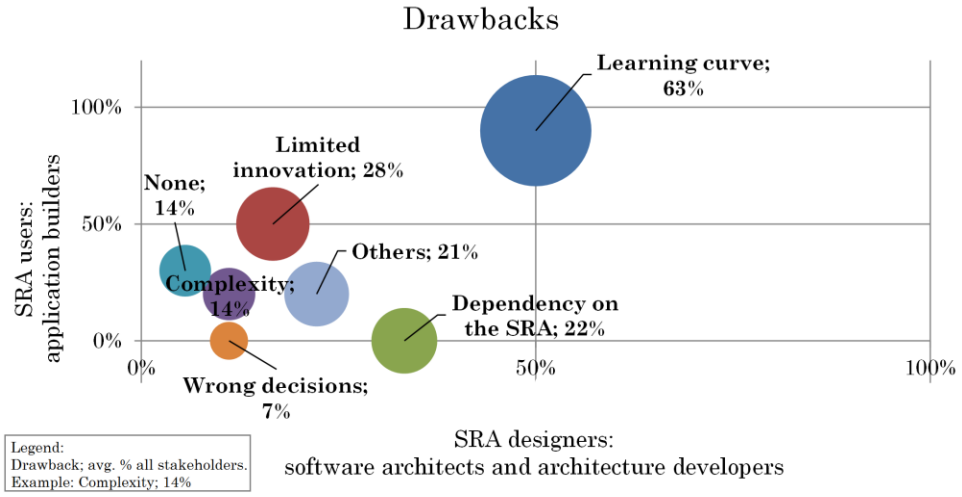

Fig. 5. Comparison of SRA drawbacks between SRA designers and SRA users.

In addition, the stakeholders were also asked about potential improvements they would make to the SRA.

4.2.2. RQ2.2: Improvements that stakeholders would make.

We asked the stakeholders what they thought should be changed, included, or updated in future versions of the SRA. The stakeholders mentioned the following improvements to be made (see Table VII):

- *(Imp-A)* **Add functionality or modules to the SRA**. For instance, one interviewee suggested developing a visual plugin to facilitate the development of components and automate the job more for application builders.
- *(Imp-B)* **Technology change,** because the current one is not mature enough or appropriate enough, or needs an upgrade to the latest version, e.g., more up-to-date BPM engines or migration from JSF to allow mobile technologies.
- *(Imp-C)* **Simplify modules**, e.g., when they cover too many functionalities.
- *(Imp-D)* **Add new practices or guidelines to the SRA**, e.g., move towards a continuous integration approach.
- *(Imp-E)* **Migrate from legacy applications**.

The most frequently mentioned improvement, stated by 64% of the participants, is that they would add functionalities or components to the SRA (Imp-A). This means that typically SRAs are not definitive, and they are always evolving. Indeed, successful SRAs need to be evolved after they have been designed for as long as they are used.

29% of the participants considered it necessary to update some technology (Imp-B), which is related to Ben-D. One possible reason could be that new stakeholders who enter an SRA project would have made different decisions. Obviously, changing technologies requires extra effort, which cannot always be spent.

Table VII. Quotes from respondents about improvements.

Code	Representative quotes from software architects	# SA	# AD	# AB	% Total
Imp-A	"Our aim is to foster a visual plugin that makes the development easier and automates the development of components for the application builders more" [C].	6	4	8	64%
Imp-B	"We are limited to JSF; migrating to another framework, like Sencha, would allow offering presentation not only for web browsers, but also for mobile devices" [G].	5	2	1	29%
Imp-C	"There is a module that is more complete than what is really asked for. Thus, it is not aligned with the client needs and therefore it should be simplified" [E].	1	2	0	11%
Imp-D	"The main point is to move to continuous integration" [H].	2	0	0	7%
Imp-E	"There are very old software elements that we have inherited. It would be good to update them because you do not know them well since they are black boxes" [B].	1	0	0	4%

5. DISCUSSION OF MAIN FINDINGS

The aim of Section 4 was to show the results from this case study and to discuss the vision of different types of stakeholders (not addressed in previous studies). In this section, we compare the main findings of our study with respect to the literature. It is important to analyze how such findings support the claims made by researchers. Sections 5.1 and 5.2, respectively, focus on the benefits and drawbacks of SRA usage. In Section 5.3, we analyze whether these benefits and drawbacks are more common in certain application domains. In Section 5.4, we discuss how to use these results. In Section 5.5, we discuss the motivation and utility of this case study for *everis*.

5.1 Discussion of SRA benefits

Table VIII compares the theoretical benefits of SRAs (see Section 2.2) with the results of this case study. The first column indicates the benefit. In total, there are eleven benefits that have been previously discussed in the literature. For them, the second column shows the extent to which the results from our study confirm (√), partially support or help to understand (±), do not explicitly mention (°), or refute (×) these benefits. If the benefit was mentioned while asking about the benefits of using the *everis* reference model (instead of SRAs), we indicate this with an asterisk (*), see Section 5.1.1. The third column represents the percentage of stakeholders who mentioned that benefit. Finally, some comments are made based on the findings of this study. Some benefits classified as Ben-H have not been listed because they were mentioned only once.

The results of the study show that the practitioners at everis have different perceptions regarding SRA benefits. For instance, in this study, the participants attributed more importance to reuse and facilitation than to other benefits.

The most commonly perceived benefit of SRAs in client organizations is systematic reuse (B3), leading to the reduction of the time and cost required to develop and maintain applications and thus to shorter time-to-market.

The second most frequently perceived benefit was facilitation due to the provision of artifacts for the design and development of applications (B2), which is specifically discussed in [32].

In our study, interoperability (B6) and best practices (B10) were not frequently mentioned as a benefit of SRAs. This might be because the stakeholders we approached did not have this in mind explicitly as these were not the main goals of their projects.

5.1.1. The use of a reference model, a confounding factor.

As a vendor, *everis* designs SRAs for its client organizations using its corporate reference model, see Fig. 3. In this context, two theoretical SRA benefits (B7, B8) were only mentioned when we asked about the advantages of designing many SRAs (i.e., reference model benefits). Therefore, in the respondents' point of view, they are not benefits of SRA use (i.e., the context of *everis*' client organizations), but rather of reference model use (i.e., the context of *everis* itself).

We were very surprised that our respondents did not mention having a shared architectural mindset (B8) as a benefit of using an SRA, and that only 11% highlighted it as a reference model

benefit. In our opinion, improving communication among multiple stakeholders who develop and maintain a wide portfolio of applications is a key benefit of SRAs. Also, creating a knowledge repository (B7) could be a benefit for the client organization that uses an SRA when this SRA has matured enough and has been evolved. To sum up, we think that B7 and B8 could be benefits of SRAs in contexts in which the organization that uses the SRA is also the one that designs it (i.e., there is no vendor).

Besides knowledge repository and improved communication, reputation was uncovered to be an important benefit for SRA vendors. Client organizations rely more on vendors who have already tested their experience in other organizations. As a software architect mentioned: "It gives prestige to announce yourself as the provider of successful SRAs" [D]. Also, vendors of the SRA are more likely to also be the provider that develops the applications, e.g., "once you define the SRA for a client organization, you have more options of developing applications on top of that SRA, since you know it" [H]. For this reason, the use of reference models is becoming popular among SRA vendors [9].

Table VIII. Summary of benefits of using SRAs.

Benefits	Diagnostic	%	Some further findings from this study
Standardization (B1)	±	33%	(Ben-F) Unlike other stakeholders, more than half of software architects indicated as a benefit that SRAs homogenizes the development and maintenance of applications.
Facilitation (B2)	√	68%	(Ben-C) Stakeholders claimed that SRA artifacts make the development of applications easier.
Reuse (B3)	√	82%	(Ben-A) (Ben-B) Stakeholders indicated cost savings in SRA-based application development and maintenance because of systematic reuse of both architectural knowledge and common elements.
Risk reduction (B4)	±	33%	(Ben-G) Stakeholders (mainly software architects) pointed out increased reliability of applications because the software elements of SRAs have been previously developed, tested, and matured.
Enhanced quality (B5)	±	82%	The improvement of architecturally significant requirements was highlighted. However, this is not due to the use of an SRA, but to that of any software architecture. Quality attributes clearly promoted by SRAs are reusability and maintainability due to reuse.
Interoperability (B6)	◦	-	Although some SRAs integrate new applications with services, legacy applications, and other back-ends, stakeholders did not explicitly mention this as a benefit.
Knowledge repository (B7)	√*	67%	SRA designers, as vendors of an SRA, mentioned this only when asked about the reference model. They pointed out the importance of harvesting experience from previous successful projects and making it explicit.
Improved communication (B8)	±*	11%	SRA designers, as vendors of an SRA, mentioned this only when asked about the reference model. SRAs help to share an architectural mindset among all stakeholders, even when they are from multiple vendors or work at multiple locations.
Elaboration of mission, vision and strategy (B9)	×	7%	No participant mentioned this. Rather, two software architects remarked that this is a benefit from enterprise architectures: "Enterprise architectures are more ambitious than SRAs; they do not only cover the technological part, but also the business level." [A]. Yet, they pointed out that applications are better aligned with business needs (Ben-E).
Best practices (B10)	◦	-	Stakeholders did not highlight the use of best practices as a benefit (only one software architect did, see Ben-H). However, SRAs provided best practices: "If provided best practices are not followed, the use of the SRA is not going to be positive" [C].
Novel design solutions (B11)	±	53%	(Ben-D) SRAs are a way to use the latest technologies in a portfolio of applications.

5.2 Discussion of SRA drawbacks

Table IX shows the drawbacks from using SRAs following the same format used in Table VIII above. In this study, three new drawbacks of SRAs were found, which are shown in the last three rows because they could not be matched to the theoretical ones. These three drawbacks are:

learning curve, dependency on the SRA, and complexity. It is important that practitioners are aware of them so that they can focus on lowering the risk arising from them when designing or evolving their SRAs.

First, the learning curve is the most common problem in SRA projects. This learning curve mainly consists of training the application builders in new technologies and specific design decisions. Typical responses were, e.g.: "Although the SRA is based on standards and de-facto technologies, it is very specific and there are extra features that need to be learned" [B]; "SRA requires knowledge in all the layers, not only in the business layer" [E]; "although the organization had experts in Oracle Forms, they did not have knowledge about developing Java applications" [H]; "the learning curve is low as long as the underlying basic technologies are already known" [F-AD]; "you can learn the SRA essentials in two weeks, but to gain a deep understanding, it requires more than one year" [I-AB].

Second, software architects and architecture developers highlighted as an important activity the management of the dependencies that an SRA creates for the development of applications.

Third, a factor that can really jeopardize the success of an SRA is its complexity. If an SRA is complex and its goal is to facilitate the daily work of application builders, it would be a failure. SRA designers should be aware of this risk in order to create easy-to-use SRAs.

Table IX. Summary of drawbacks of SRAs.

Drawbacks	Diagnostic	%	Some further findings from this study
Initial investment (D1)	±	4%	(Dra-G) Stakeholders mentioned the necessity of an initial investment in the SRA, but they did not strongly word it. One reason may be that we did not include as stakeholders the upper management of client organizations.
Inefficient instantiation (D2)	±	11%	(Imp-C) They mentioned the problem of designing common software elements without bearing in mind business needs, which may lead to inefficient instantiation of the SRA.
Too abstract (D3)	∘	-	None of the participants mentioned that the SRA of their project was too abstract or not abstract enough. We believe that since the studied SRAs were used in industry and applications have been implemented based on them, they were practical and provided common software elements and guidelines.
Term confusion (D4)	±	43%	5 architecture developers and 7 application builders reported problems with some term confusion (e.g., they did not give a definition compliant with the SRA concept).
Bad documentation (D5)	×	4%	No one reported problems with bad documentation. Indeed, documentation was described as a key asset of the SRA to help application builders, and one application builder explicitly mentioned documentation as a benefit.
Bad quality (D6)	±	7%	(Dra-F) Software architects were concerned about the consequences from making a wrong decision in the SRA. In this context, this may be a very risky problem since the quality of an SRA is propagated to the applications.
Limiting (D7)	±	28%	(Dra-B) Limiting the creativity of developers by making the development of applications less flexible.
Learning curve	new	63%	(Dra-A) Additional learning curve for application builders.
Dependency on the SRA	new	22%	(Dra-C) Applications depend on the common elements provided by the SRA.
Complexity	new	14%	(Dra-D) Even considering that SRAs aim to be easy to use, a minority of stakeholders indicated that it was complex.

5.3 SRA benefits and drawbacks by application domain

After the comparison of these results with the literature, we analyzed the differences among the benefits and drawbacks of SRAs from different application domains. In this case study, the SRA application domains were banking, insurance, public sector, and industry (see Table III).

Regarding contextual aspects, we could see that in SRA projects from the banking domain the effort invested was higher, both in terms of person-months and duration of the projects (i.e., years). With respect to specific benefits and drawbacks, we could not detect any pattern or correlation to determine that some benefits or drawbacks are exclusive of an application domain. However, some effects caused by SRAs were perceived more strongly in some application domains than in others. In the banking domain, maintenance costs (Ben-B) and the learning curve

(Dra-A) of SRA projects were higher in comparison with the other application domains. In the public sector domain, standardization (Ben-F) was perceived more often as a benefit whereas easier development (Ben-C), the use of the latest technologies (Ben-D), limitations (Dra-B), and dependency on the SRA (Dra-C) were perceived as higher in the rest of the application domains. In other domains (i.e., industry and insurance), no correlations were found between the application domain and specific benefits/drawbacks.

5.4 How to use these results

This paper aims to help SRA practitioners as follows.

First, for organizations that need to decide whether or not to go for an SRA program, understanding the benefits and drawbacks associated with real SRAs can help them to realize important situations and make industrial uptake of SRA research efforts easier.

Second, organizations that have already adopted an SRA can use these empirical results as a point of reference to assess their own benefits and drawbacks. For instance, they may see that an additional learning curve is a commonly mentioned drawback in the *everis* SRA projects.

Third, organizations can gain insights from the participants' responses about how to improve their own SRAs. Table X shows the main improvements mentioned by the stakeholders, the type of role that is interested the most in each improvement, and the benefits and drawbacks it affects.

Table X. Summary of improvements and trade-off analysis of the benefits they promote and the risks that need to be managed.

Improvement	Requested by	Promotes	Need to manage
Add functionality in the common SRA elements (Imp-A)	Application builders	Facilitation of application development and evolution (Ben-C)	Dependency on the SRA (Dra-C)
Change of SRA technologies (Imp-B)	Software architects	Update to the latest technologies (Ben-D)	Wrong decisions (Dra-F)
Simplify SRA modules (Imp-C)	Architecture developers	Easier development (Ben-C) and shared mindset	Complexity (Dra-D)
New SRA procedures (Imp-D)	Software architects	Standardization (Ben-F)	Limitation of innovation (Dra-B)

To sum up, the organizations in our case study experienced particular benefits at different degrees. However, the achievement of specific benefits depends on the issues the organization wants to resolve with the help of the SRA (see Table III). For instance, one organization may aim, with different weights, at standardizing the development of their applications, easing the application's development or interoperability. As a consequence, we think that every organization should clearly state the benefits they aim to achieve with the SRA (a subset of the benefits of Table VIII). Then they can manage the SRA project in order to achieve them. Moreover, it is important to note that these goals might not be static and can evolve over time.

Regarding researchers, the results from this study (summarized in Tables VIII, IX, and X) may be used to generate hypotheses and test them in other empirical studies. In this paper, we aim to provide as much contextual data as possible to allow the reader to draw up their own hypotheses or make their own observations based on our data as in other qualitative works [37,38].

5.5 Motivation and utility of this case study for *everis*

As aforementioned, this case study was conducted inside an industry-academia collaboration between *everis* and UPC. From the research perspective, the initial motivation for everis was: "promoting training in information technology (IT) by conducting research, innovation, knowledge transfer and dissemination" [39]. More specifically, the goal of the collaboration was to provide a solution to the challenges that *everis* faced in SRA projects. Examples of such challenges were to assess if an SRA is a good approach for software development in a client organization, or to align current SRA projects to the client organization's goals.

In a previous work [39], we evaluated the value of these research results for *everis* following the model of Sandberg et al. [40]. Next, we discuss the deployment impact and industry benefit factors of the model. The results of this case study were deployed as knowledge (i.e., in the form of internal reports, presentations, and executive summaries) in the internal collaboration site of *everis*, and were disseminated in internal meetings with the employees. As a consequence, these results have had an impact on SRA projects involving employees who participated in this collaboration, both in pilot studies executed together, and in similar SRA projects. On the other hand, the results have been mainly used in the context of the architecture group located in Barcelona and are expected to be transferred to other *everis* locations by their own teams. To increase the industry benefit, we have proposed to the *everis* management the following activities: promotion of lightweight materials and tool support for employees, and celebration of workshops and training courses.

6. LIMITATIONS OF THE STUDY

This section discusses possible threats to validity in terms of construct, internal, and external validity. It also emphasizes the mitigation actions applied.

Construct validity. This refers to issues that affect our ability to reflect the constructs under study using adequate instruments. To strengthen this aspect, we performed rigorous planning of the study and established a rigorous protocol [33]. We designed our data collection instruments in such a way that they were fully understood by the respondents.

We are aware that the online questionnaires used for gathering responses from architecture developers and application builders limited our ability to further enquire about their perceptions compared to the software architects' perceptions, which were gathered through face-to-face interviews. To mitigate this threat, we selected the options provided in the questionnaires together with *everis* managers and with input from the software architects interviewed. In addition, we added open questions to the questionnaires to gather the participants' opinions that did not match any of the given options.

In all cases, we made sure to polish the instruments with suitable vocabulary that the participants were familiar with. This was particularly relevant in our case as the different stakeholders used different terms to refer to the same thing. Thus, all instruments were revised by *everis* managers, piloted, and enhanced to ensure their effectiveness. Furthermore, we included specific actions to mitigate evaluation apprehension by ensuring confidentiality and aggregating the answers. Hence, the respondents could freely share their real perceptions, either in the interviews or in the open questions of the questionnaires.

Internal validity. This refers to factors that might affect our conclusions. For instance, when the researcher is investigating whether a certain factor affects an investigated factor, there is a risk that the investigated factor may also be affected by a third factor [33]. We are aware that the *everis* managers chose SRAs that were already used by their respective organizations to build concrete software architectures and that they might have picked the most successful projects for the sampling. To minimize this issue, we explained to them the importance of having a representative sampling of the SRA projects in order to obtain reliable data that reflect the perceptions of different roles. In addition, the fact that various roles from these projects were chosen as the unit of analysis allowed us to better interpret and assess contextual information.

It is important to emphasize that our results are based on the stakeholders' perceptions from the specific project in which they participated. Therefore, even if an SRA benefit/drawback was not explicitly mentioned by these individuals, there might be several factors affecting this, for instance that our instruments did not explicitly request some potentially influential information, or cultural issues. In addition, regarding the individuals who participated in the study, there is always the possibility that they may have forgotten something or failed to explicitly state something when asked about it. To reduce this risk, we took the following precautions: 1) In the case of the interviews, we discussed some potential topics that might be omitted by the respondents and paid

particular attention to asking for clarifications, if necessary; 2) in the case of the online questionnaires, we designed them in such a way that the respondents had to answer all the corresponding questions while being allowed to complete the questionnaire at any time, which gave them the chance to consult registries and documentation if they did not remember something; 3) in all cases we also had the opportunity to contact the participants after the interview/questionnaire to send them their responses. Two software architects provided small clarifications after checking the transcription of their interviews. For instance, B-SA corrected his response about reduced time-to-market, clarifying that it is not reduced in the case of new common SRA functionalities because they could only release an evolution of the SRA every three months. We also made sure to design our data collection instruments in such a way that tricky questions had related questions that helped to confirm the correctness of the answers. For instance, we had two questions asking about the benefits of SRAs and differentiated between the benefits from an SRA and those from the corporate reference model.

The study investigated SRA benefits and drawbacks, but since SRAs are just one asset used by *everis* in their software process, it might be the case that some of these benefits and drawbacks attributed to the use of SRA are, in fact, related to other software engineering practices. When designing the instruments of the study, we discussed with the *everis* managers (who have extensive experience in SRA design, development, and building) the potential benefits to be included in the instruments, and none of them suggested emphasizing this separation. Therefore, we took no special measures to mitigate this threat. In fact, we observed that, when performing the interviews with the software architects, they naturally mentioned the reported benefits without making any difference either.

Another threat to validity is related to the experience of the participants. Since the goal of this case study is to gather the perception of different stakeholders, we included stakeholders with diverse experience characteristics, such as application builders who had no previous professional experience in developing software without SRAs. Although this could deviate the results, we analyzed their answers separately and reported their experience in Table IV.

Other mitigation strategies included the recording and transcription of all interviews aimed at contributing to a better understanding and assessment of the collected data. Also, to reduce potential researcher bias, several meetings were held among the researchers and *everis* managers to discuss the course of the case study and the preliminary results.

External validity. This is concerned with the extent to which it is possible to generalize the findings and to what extent the findings are of interest to other people outside the investigated case. We recognize that our results are tied to the context of the client organizations of *everis* and should therefore be interpreted as such. Our Results section provides representative categories obtained through analytical generalization of the gathered evidence. We exhaustively analyzed the data together with *everis* managers, using Why questions to fully understand and explain the results. To strengthen the correct understanding of this analytical generalization, in this paper we aim to provide as much information as possible about the context and quote sentences from the participants themselves.

We recognize that our results cannot be generalized to other organizations without further work. However, we claim that our study can serve as a basis for other similar organizations to study their SRA's benefits and drawbacks [9]. Other IT consulting firms that could be considered to some extent similar to *everis* are, for instance, Accenture [41] and Capgemini [42], as they use an industry-specific reference model to provide support to their clients to adopt SRAs, and they have similar professional roles to perform the associated tasks (see Fig. 3). Besides IT consulting firms, other companies have reported similar use of SRAs without using a corporate reference model, such as Volvo [43]), Océ [30], Credit Suisse [44], and the Dutch e-government [38]. It is also important to note that all aforementioned SRAs are based on practical experience in industry.

To enable replication, we designed our instruments in a way that would enable other researchers and practitioners to use them and compare the results (see Annex A). We expect that our results will strengthen the evidence regarding SRAs and encourage others to provide similar evidence to help mature SRA research and practice.

7. CONCLUSIONS AND FUTURE WORK

An SRA provides a blueprint for developing concrete software architectures for applications that share a common base. The scientific literature has discussed the benefits and drawbacks of this concept. However, the perspective of academics regarding SRA and its benefits and drawbacks are not always in line with industry practice.

With the goal of supporting organizations that plan to adopt or have adopted an SRA, this paper has addressed the benefits and drawbacks of SRAs in large organizations. A case study was conducted to analyze how SRAs are perceived by industrial practitioners from nine client organizations of *everis*. The results help to increase the empirical evidence about SRA as follows.

First, this study supports several SRA benefits already identified by researchers, mainly cost savings in the development and evolution of applications, and facilitation of the design of concrete software architectures. Also, it shows that a single organization does not necessarily realize all the theoretical benefits of SRAs. On the contrary, our study has uncovered a scenario in which an organization realizes a subset of these benefits, depending on the goals of the SRA.

Second, the study revealed one main drawback of adopting SRAs: the additional learning curve experienced by the application builders who use an SRA. Besides, two risks not previously reported were uncovered. These risks are higher time-to-market for applications when their dependencies on the SRA are managed inappropriately, and high complexity of SRAs when they do not properly aim to facilitate the development of applications. These results show that experience reports about negative experiences are also needed.

Third, we analyzed how important SRA benefits and drawbacks are for various SRA stakeholder types, which had not been addressed in previous research. As a consequence, we can see the differences in their concerns. For instance, software architects claim that standardization and reliability are key benefits of SRAs, whereas application builders are worried about the use of the latest technologies.

Finally, yet importantly, we discuss how practitioners can use the results of this case study to decide whether to adopt an SRA, compare their SRA projects with the industrial evidence from this paper, or evolve an SRA, taking into account the improvement actions recommended by the participants of this case study.

The results of this case study offer useful evidence that may mostly serve organizations with a similar context. Organizations that have previously reported a context similar to that of *everis*' client organizations were analyzed in Section 6 (external validity). As Seddon et al. suggests: "If the forces within an organization that drove the observed behavior are likely to exist in other organizations, it is likely that those other organizations, too, will exhibit similar behavior" [45].

As further work from this case study, we plan to consolidate the available evidence about SRA benefits and drawbacks in various contexts by means of the structured synthesis method [46]. Some preliminary results of this endeavor can be found in [47].

ACKNOWLEDGMENTS

This work has been supported by the ERCIM Fellowship Programme, the Spanish grant FPU12/00690, and the Cátedra *everis* contract. We thank all participants of the case study. We also thank Samuil Angelov, Matthias Galster, and Elisa Nakagawa, who gave feedback about the comparison between SRAs and PLAs.

REFERENCES

[1] R. Cloutier, G. Muller, D. Verma, R. Nilchiani, E. Hole, M. Bone, The Concept of Reference Architectures, Syst. Eng. 13 (2010) 14–27.

[2] S. Angelov, P. Grefen, D. Greefhorst, A framework for analysis and design of software reference architectures, Inf. Softw. Technol. 54 (2012) 417–431.

[3] P. Kruchten, The Rational Unified Process: An Introduction, Addison-Wesley Professional, 2004.

[4] P. Pääkkönen, D. Pakkala, Reference Architecture and Classification of Technologies, Products and Services for Big Data Systems, Big Data Res. 2 (2015) 166–186.

[5] M. Weyrich, C. Ebert, Reference Architectures for the Internet of Things, IEEE Softw. 33 (2016) 112–116.

[6] D. Scott, Gartner Hype Cycle for Real-Time Infrastructure, (2012). https://www.gartner.com/doc/2098715/hype-cycle-realtime-infrastructure-.

[7] S. Angelov, J. Trienekens, R. Kusters, Software reference architectures-exploring their usage and design in practice, in: ECSA 2013: 17–24.

[8] G. Muller, P. van de Laar, Researching Reference Architectures, in: CSER 2009.

[9] S. Martínez-Fernández, C.P. Ayala, X. Franch, H. Marques, D. Ameller, Towards Guidelines for Building a Business Case and Gathering Evidence of Software Reference Architectures in Industry, J. Softw. Eng. Res. Dev. 2 (2014).

[10] S. Martínez-Fernández, C.P. Ayala, X. Franch, H. Martins Marques, Benefits and drawbacks of reference architectures, in: ECSA 2013: 307–310.

[11] M. Galster, P. Avgeriou, Empirically-grounded reference architectures: a proposal, in: QoSA 2011: 153–157.

[12] The Open Group, SOA Reference Architecture, ISBN 1-937218-01-0. (2011).

[13] J. Meier, D. Hill, A. Homer, T. Jason, P. Bansode, L. Wall, R. Boucher Jr, A. Bogawat, Microsoft Application Architecture Guide. Patterns and Practices, Microsoft Corporation, 2009.

[14] C. Ballard, C. Compert, T. Jesionowski, I. Milman, B. Plants, B. Rosen, H. Smith, IBM Redbooks | Information Governance Principles and Practices for a Big Data Landscape, (2014).

[15] A. Grosskurth, M.W. Godfrey, A reference architecture for Web browsers, in: ICSM 2005: 661–664.

[16] L. Bueno, E.Y. Nakagawa, A Service-Oriented Reference Architecture for Software Testing Tools, in: ECSA 2011: 405–421.

[17] AUTOSAR, The Worldwide Automotive Standard for E/E Systems, ATZextra Worldw. 18 (2013) 5–12.

[18] NASA, ESDS Reference Architecture for the Decadal Survey Era, 2012. https://wiki.earthdata.nasa.gov/download/attachments/49447036/ESDS Reference Architecture v1.1.pdf?version=1&modificationDate=1428611037665&api=v2.

[19] S. Mazzini, J. Favaro, T. Vardanega, Cross-Domain Reuse: Lessons Learned in a Multi-project Trajectory, in ICSR 2013: 113–126.

[20] B. Gallagher, Using the Architecture Tradeoff Analysis MethodSM to Evaluate a Reference Architecture: A Case Study, 2000. http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA388852.

[21] E. Nakagawa, P. Antonino, M. Becker, Reference architecture and product line architecture: A subtle but critical difference, in: ECSA 2011: 207–211.

[22] L. Dobrica, E. Ovaska, Analysis of a cross-domain reference architecture using change scenarios, in: ECSA 2011: p. 10:1--10:9.

[23] P. Reed, Reference Architecture: The best of best practices, IBM Dev. Work. (2002). http://www.ibm.com/developerworks/rational/library/2774.html.

[24] S. Martínez-Fernández, C.P. Ayala, X. Franch, H.M. Marques, REARM: A reuse-based economic model for software reference architectures, in: ICSR 2013: 97–112.

[25] U. Eklund, A. Eriksson, J. Bosch, A Reference Architecture Template for Software-Intensive Embedded Systems, in: WICSA/ECSA 2012: 104–111.

[26] M. Henning, The Rise and Fall of CORBA, ACM Queue. (2006). http://queue.acm.org/detail.cfm?id=1142044.

[27] M. Galster, Software Reference Architectures: Related Architectural Concepts and Challenges, in: CobRA 2015: 5–8.

[28] L. Bass, P. Clements, R. Kazman, Software Architecture in Practice, Addison-Wesley 2003.

[29] S. Deelstra, M. Sinnema, J. Bosch, Product derivation in software product families: a case study, J.

Syst. Softw. 74 (2005) 173–194.
[30] B. Graaf, H. van Dijk, A. van Deursen, Evaluating an Embedded Software Reference Architecture—Industrial Experience Report—, in: CSMR 2005: 354–363.
[31] L.M. Northrop, P.C. Clements, A Framework for Software Product Line Practice, Version 5.0, (2007). http://www.sei.cmu.edu/productlines/frame_report/benefits.costs.htm
[32] S. Martínez-Fernández, C. Ayala, X. Franch, H.M. Marques, Artifacts of Software Reference Architectures: A Case Study, in: EASE 2014: p. 42:1--42:10.
[33] P. Runeson, M. Höst, Guidelines for conducting and reporting case study research in software engineering, Empir. Softw. Eng. 14 (2009) 131–164.
[34] P. Runeson, M. Host, A. Rainer, B. Regnell, Case Study Research in Software Engineering: Guidelines and Examples, Wiley, 2012.
[35] D.S. Cruzes, T. Dybå, P. Runeson, M. Höst, Case studies synthesis: a thematic, cross-case, and narrative synthesis worked example, Empir. Softw. Eng. 20 (2015) 1634–1665.
[36] W.R. Dillon, M. Goldstein, Multivariate analysis: methods and applications, Wiley, 1984.
[37] C. Ayala, Ø. Hauge, R. Conradi, X. Franch, J. Li, Selection of third party software in Off-The-Shelf-based software development—An interview study with industrial practitioners, J. Syst. Softw. 84 (2011) 620–637.
[38] M. Galster, P. Avgeriou, D. Tofan, Constraints for the design of variability-intensive service-oriented reference architectures–An industrial case study, Inf. Softw. Technol. 55 (2013) 428–441.
[39] S. Martínez-Fernández, H.M. Marques, Practical experiences in designing and conducting empirical studies in industry-academia collaboration, in: CESI@ICSE 2014: 15–20.
[40] A. Sandberg, L. Pareto, T. Arts, Agile collaborative research: Action principles for industry-academia collaboration, IEEE Software (2011) 74–83.
[41] S. Brand, Accenture Has Business-Outcome-Driven EA Capabilities, but Doesn't Automatically Begin With This Approach, (2014). https://www.gartner.com/doc/2651816/accenture-businessoutcomedriven-ea-capabilities-doesnt.
[42] S. Herold, M. Mair, Checking Conformance with Reference Architectures: A Case Study, in: EDOC 2013.
[43] U. Eklund, Ö. Askerdal, J. Granholm, Experience of introducing reference architectures in the development of automotive electronic systems, in: SEAS 2005: 1–6.
[44] S. Murer, C. Hagen, 15 Years of Service Oriented Architecture at Credit Suisse, IEEE Software (2013) 9–15.
[45] P.B. Seddon, R. Scheepers, Towards the improved treatment of generalization of knowledge claims in IS research: drawing general conclusions from samples, Eur. J. Inf. Syst. 21 (2011) 6–21.
[46] P.S. Medeiros Dos Santos, G.H. Travassos, On the representation and aggregation of evidence in software engineering: A theory and belief-based perspective, Electron. Notes Theor. Comput. Sci. 292 (2013) 95–118.
[47] S. Martínez-Fernández, P.S. Medeiros Dos Santos, C.P. Ayala, X. Franch, G.H. Travassos, Aggregating Empirical Evidence about the Benefits and Drawbacks of Software Reference Architectures, in: ESEM 2015: 154–163.