UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Gathering Empirical Evidence and Building a Business Case for Software Reference Architectures in Industry

— PhD Thesis —

SILVERIO MARTÍNEZ-FERNÁNDEZ
Advised by CLAUDIA AYALA and XAVIER FRANCH

A thesis submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computing

Department of Computer Science

Barcelona, Spain
April 2016

A mi padre

Abstract

Background: Software reference architectures are becoming widely adopted by organizations that need to support the design and maintenance of software applications of a shared domain. For organizations that plan to adopt this architecture-centric approach, it becomes fundamental to understand how software reference architectures are engineered, and to know their return on investment. Unfortunately, there is a lack of evidence-based support to help organizations with these challenges.

Goal: The main goal of this PhD thesis is to support organizations making informed decisions about software reference architecture acquisition, design, and use based on empirical evidence.

Methods: To accomplish this goal, we have conducted an action research approach in an industry-academia collaboration between *everis* (a multinational IT consulting firm based in Spain) and our Research Group of Software and Service Engineering (GESSI).

Results: The results from our industry-academia collaboration led to uncover novel evidence on the use of software reference architectures in practice. The procedures and evidence obtained have been packaged to design guidelines that could be used in similar contexts as the one of *everis*.

Conclusions: This PhD thesis supports organizations to acquire and engineer software reference architectures by providing evidence-based support. Such evidence-based support consists of the results of the empirical studies conducted in this PhD thesis, and the presented guidelines for gathering new corporate evidence.

Keywords: software architecture; reference architecture; software reference architecture; empirical software engineering; experimental software engineering; reference architecture acquisition benefits; business case; cost-benefit analysis; industry-academia collaboration.

PhD Thesis:	*Gathering Empirical Evidence and Building a Business Case for Software Reference Architectures in Industry*
Author:	Silverio Martínez-Fernández
Department:	Service and Information System Engineering (ESSI) Group of Software and Service Engineering (GESSI)
Address:	UPC-Campus Nord, Omega building, room S206 c/Jordi Girona 1-3, 08034 Barcelona (Spain)
E-mail:	smartinez@essi.upc.com, martinez.silverio@gmail.com

Co-Authorship Statement

Most of the contents in this PhD thesis are based on published papers authored by the candidate. The content of the papers included in this PhD thesis has been adapted with respect to the published version. In some cases, the papers have been co-authored with other authors. We indicate below the co-authors of such works.

All Chapters

Dr. Claudia Ayala and Dr. Xavier Franch, as supervisors, have contributed to all research described in this PhD thesis. Their guidance, supervision, and active role during these years have been fundamental.

Chapter 3

This chapter is partially based on the work performed during my research stay at the LabES research group at the University of Sao Paulo (Brazil). It has also been supervised by Dr. Elisa Y. Nakagawa, and jointly performed with Dr. Lucas Bueno Ruas de Oliveira and Lina Garcés.

Chapters 4, 5, 9, 10, and 13

These chapters are based on the research performed inside the "*Cátedra everis-UPC*" project. Helena M. Marques, Xavier Terradellas Fernández, and Miguel Ángel González Amate have contributed stating the research goals, with their participation at many meetings, and cooperation to gather evidence for the empirical studies.

Chapters 6 and 11

In the beginning of this PhD thesis, Dr. David Ameller participated in the *"Cátedra everis-UPC"* project, contributing to the initial versions of the guidelines.

Chapter 7

Dr. Elisa Nakagawa presented the survey performed to AUTOSAR practitioners in the First International Workshop on Automotive Software Architecture (WASA).

Chapter 8

This chapter is based on the work performed during my research visit at the Experimental Software Engineering group of the Federal University of Rio de Janeiro (UFRJ). It has also been supervised by Dr. Guilherme Horta Travassos, and jointly performed with Dr. Paulo Sérgio Medeiros dos Santos.

QuPreSS Reference Model

An initial step of this PhD thesis was the creation of QuPreSS, a reference model for predictive services selection. With the help of QuPreSS, we could implement Mercury, a tool for evaluating predictive services customized to the weather forecast domain. This helped me to learn and gain experience on designing and using reference models. This work has also been supervised by Dr. Jesús Bisbal.

Acknowledgments

I would like to express my sincere gratitude to all people who have shared some time with me in the way to my PhD thesis. I am grateful for learning from them as a professional and as a person.

First of all, I want to thank my advisors, Claudia Ayala and Xavier Franch, for their guidance. A PhD thesis introduces you in a no existing road, but I walked on their previous footsteps. They have always been there providing inspiring knowledge and extremely useful assistance. There have been many meetings, many papers, many funny e-mails, many conference trips... but what made them different was their ability to lead this project by engaging me on the research, and improving the quality of the work.

To the whole GESSI research group, I need to say that I have been lucky to work here. The work atmosphere has been excellent, getting their friendly cooperation whenever I needed it, especially from my mates at the office: David Ameller, Óscar Cabrera, Montse Estanyol, Óscar Hernán, Lidia López, Marc Oriol, Xavier Oriol, and Cristina Palomares.

One of the most enriching parts of the PhD has been to do empirical research inside the "Cátedra *everis*-UPC" project. Thanks to all *everis*' employees who have participated in the studies, and to the people from the architecture group (ARCHEX). Special thanks to Xavier Terradellas and Miguel Ángel González, who helped to define the research problem in the very beginning; and to Helena M. Marques, the champion at the *everis* side of the industry-academia collaboration. She has always been willing to cooperate to gather evidence for the empirical studies.

To colleagues and researchers abroad, I am mainly thankful to Elisa Yumi Nakawaga from the University of Sao Paulo (USP) for giving me the opportunity to learn the working philosophy of her group. To all members from USP for their kind hospitality and cooperation during my research stay there, especially to my office mates Lucas Bueno and Lina Garcés. Also to Guilherme

H. Travassos and Paulo Sérgio Medeiros from the Federal University of Rio de Janeiro (UFRJ), for impressing upon me the importance of aggregation and meta-analysis.

I would like to thank to the anonymous reviewers as well as to researchers from many conferences, who provided great ideas and priceless feedback in different stages of this work.

I devote this PhD thesis to my mom and my sister for their love, affection, and support. They are always there to encourage me to follow my goals and believing in me. Especially for instilling me in the values that have made me the person I am today. Also, I devote the PhD thesis to the ones who joined the family in the last years, David and Adán.

Last but not least, I would also like to thank my friends, and all my flatmates in *Can Bruixa* and *Casa da Vó*, who have kept the home in Barcelona and São Carlos always worm. They have given me a hand whenever I needed, especially Elena, Jesús, Jovan, Laia, Pedro, and Rebecca. I particularly thank Petar Jovanovic for all his support since we started our PhDs. He has not only been a real friend, but also contributed to this work with abundantly feedback, proof-readings, nights and weekends in the research hub of our living room, *y fleje cosas más*.

Thanks to all!

This work has been possible thanks to the collaboration "Cátedra everis-UPC", the Spanish FPU grant FPU12/00690, and Becas Santander JPI 2014. Besides, some of its parts were performed in the context of the Spanish MEC projects TIN2010-19130-C02-00 and TIN2013-44641-P.

Curriculum Vitae

Silverio Juan Martínez Fernández

Silverio Juan Martínez Fernández was born on October 29th, 1987, in Almería.

He graduated in Bachelor of Computer Science in June 2008 and in Computer Science Engineering in September 2010, both by University of Almeria (Spain). During his engineering studies, he spent the 2008-2009 academic year at University of Ghent (Belgium), awarded with an Erasmus grant.

In 2010, he was awarded with a "la Caixa" grant to study a master for the academic year 2010-2011. During this period, he studied the Master in Computing (Information Systems' specialization) at UPC-BarcelonaTech (Spain).

Before starting the PhD, he was a developer in two software projects. First, "Chess League Game", a chess online game that simulates team competitions (his final project degree). Second, "Mercury", a tool that measures weather predictive services' quality and automates the context-dependent selection of the most accurate predictive service to satisfy a customer query (his master thesis).

Since the end of 2011, he joined the Research Group of Software and Service Engineering (GESSI) at UPC-BarcelonaTech to carry out his doctoral studies. To this end, he got the first year and a half a FPI-UPC grant, and then a FPU grant from the Spanish Goverment. His research areas in the PhD have mainly been software reference architectures and empirical software engineering. During the three first years, he also earned industrial experience as a part of the research conducted in the "Cátedra *everis*-UPC" project.

During his doctoral studies, he was also the recipient of some competitive grants for researching and teaching. On the one hand, he was awarded with a Santander Grant for Young Lecturers and Researchers (Santander JPI 2014), which enabled him to make a research stay at the University of Sao Paulo (Brazil) in Sao Carlos, Brazil. On the other hand, he got an Erasmus+ grant for teaching mobility, which enabled him to teach a part of a software architecture course in 2015 at Fontys University of Applied Sciences (The Netherlands).

By the time of finishing his PhD, he has participated in several national research projects, and has been the first author of 12 peer-reviewed publications, including 1 JCR indexed Elsevier journal, 1 Springer journal, 3 full-papers at the main track of CORE-A conferences, and 1 best paper award at a workshop. Also, he has been teaching the Software Engineering Project course of the Computer Science Degree at Barcelona School of Informatics in two academic years (2013-2014, 2014-2015).

Contents

List of Figures

List of Tables

Part I

Introduction and State of the Art

Chapter 1

Introduction

This document represents my PhD thesis at the PhD in Computing program at Barcelona School of Informatics (Facultat d'Informàtica de Barcelona, FIB) of BarcelonaTech (Universitat Politècnica de Catalunya, UPC). The research of this PhD thesis had its origin in the *"Cátedra everis-UPC"* project, an industry-academia collaboration among *everis* (a multinational IT consulting firm based in Spain)[1], and our Research Group of Software and Service Engineering (GESSI).

In this chapter, we respectively introduce the topic of the PhD thesis (Section 1.1), the research context (Section 1.2), the research problem (Section 1.3), the research goal (Section 1.4), the research tasks and contributions (Section 1.5), the list of publications (Section 1.6), and the structure of the rest of the document (Section 1.7).

1.1 Software Reference Architectures

Every software system has a software architecture [1]. "The software architecture of a system is the set of structures needed to reason about the system, which comprise software elements, relations among them, and properties of both" [1]. Nowadays, the size and complexity of software systems, together with critical time-to-market needs, demand new Software Engineering (SE) approaches to design their software architectures [2].

Many today's organizations face the development and maintenance of big and complex software families, rather than single software systems. A

[1]There is an introduction about *everis* in Section 1.2

software family is composed by many software systems or software applications. These software applications are developed at multiple locations, by multiple vendors and across multiple organizations [3]. Despite the multiplicity of this scenario, all software applications from the same family share similar architectural needs and belong to the same domain. In this context, organizations may build a central asset called **Software Reference Architecture (SRA)**.

An SRA "encompasses the knowledge about how to design concrete architectures of systems of a given application domain; therefore, it must address the business rules, architectural styles, best practices of software development, and the software elements that support development of systems for that domain. All of this must be supported by a unified, unambiguous, and widely understood domain terminology" [2].

An SRA is used as a foundation for the design of concrete software architectures of a class of software applications in a cost-effective manner [4, 5]. Therefore, SRAs are very attractive when organizations become large and distributed in order to develop new systems or new versions of systems [6].

The idea of using SRAs is not new for the SE community, nor the software industry. Since 1976, the idea of application families was described by Parnas [7], and more than a decade ago, Bass et al. defined SRAs in their seminal book [1] as: "a reference model mapped onto software elements (that cooperatively implement the functionality defined in the reference model) and the data flows between them". Thus, a reference model could be mapped to many SRAs. Likewise, SRAs serve as a reference for the design of the concrete architecture of the software applications of an information system. These three artifacts go from a high level of abstraction to a low level of abstraction. Figure 1.1 shows these relationships among reference model, SRAs and concrete software architecture of applications.

1.1.1 Importance of Software Reference Architectures

Several potential benefits of using SRAs have been claimed, by both industry and academia.

On the one hand, a Gartner's report summarizes the benefits of SRAs as they "reduce the complexity of hardware and software architecture by systematically reducing environmental diversity [...], enable greatly increased speed and reduced operational expenses as well as quality improvements due to lowered complexity, greater investment and greater reuse" [8]. Thus, "IT

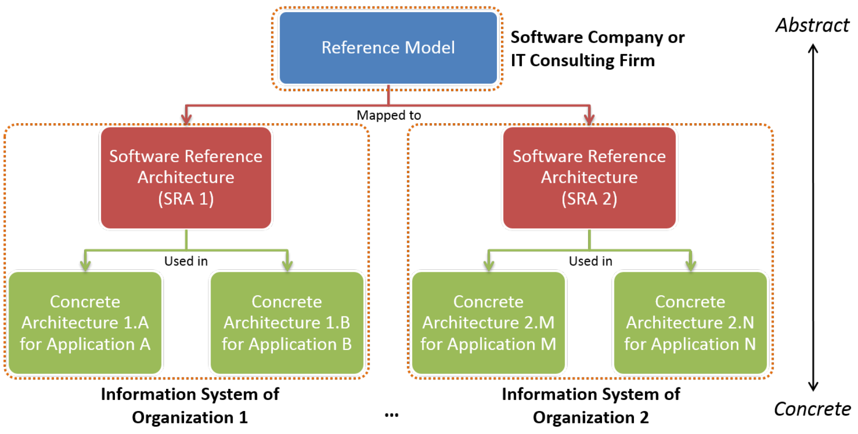
Reference Model
Software Company or IT Consulting Firm
Abstract
Mapped to
Software Reference Architecture (SRA 1)
Software Reference Architecture (SRA 2)
Used in
Used in
Concrete Architecture 1.A for Application A
Concrete Architecture 1.B for Application B
Concrete Architecture 2.M for Application M
Concrete Architecture 2.N for Application N
Information System of Organization 1
...
Information System of Organization 2
Concrete

Figure 1.1: Relationships among reference model, SRA, and concrete architecture.

organizations that lack architecture and configuration standards [...] have higher costs and less agility that those with enforced standards" [8].

On the other hand, much has been written in the scientific literature about the goals of SRAs [9], which aims:

- to ensure standardization and interoperability [4];
- to facilitate reuse, and thereby harvest potential savings through reduced cycle times, cost, risk and increased quality [3];
- to help with the evolution of a set of systems that stem from the same SRA [10];
- to manage design complexity and improve development productivity [11];
- and to ensure that resulting applications' software architecture are consistent with respect to an SRA [12].

According to these expected benefits, SRAs have become a key asset of organizations [3]. Hence, they have become widely studied and used in software architecture research and practice [4][13]. Next, we enumerate a few well-known SRAs classified by their application domain: platform-specific SRAs, industry-specific SRAs, and industry-cross-cutting SRAs [10].

First, there are SRAs that target a technological domain (also called platform-specific SRAs [10]). Examples are The Open Group standard for SOA reference architecture that is a blueprint that provides guidelines to adopt a service-oriented approach to information technology [14], and the IBM big data reference architecture that provides integrated capabilities for the adoption of information governance in the big data landscape [15]. There are also SRAs from academia to solve well-known technological problems (e.g., web browsers [16], and software testing tools [17]).

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 target 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 [18], which focuses on the automotive domain and is being used by many car manufacturers and suppliers in order to standardize the software in modern vehicles. An example of SRAs 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 [19].

Third, the aforementioned SRAs target a single domain (e.g., the automotive or aerospace industry), what makes them hard to be applied to other domains. In this context, the goal of some European industrial research programs [20] is to enable their use across disparate domains. Also, the AUTOSAR consortium plans to adapt its SRA for other commercial sectors, such as such as railway, agriculture and forestry machinery [18]. This last type of SRAs that cover more than one industry are called industry-cross-cutting SRAs.

1.2 Research Context

As it was aforementioned, the research of this PhD thesis had its origin in the "*Cátedra everis-UPC*" collaboration. The collaboration was composed of three partners: the architecture group of *everis*, the Barcelona School of Informatics (FIB) at UPC, and the GESSI research group at UPC. The collaboration, which was funded by *everis*, started in May 2011 and had a duration of three years. Its goal was promoting training in IT by conducting research, innovation, knowledge transfer, and dissemination. The goal of the collaboration was to provide a solution to the current challenges that *everis* faced in SRA projects.

1.2.1 *everis*, a Multinational Consulting Firm

everis is a multinational consulting firm providing business and strategy solutions, application development, maintenance, and outsourcing services. Established in 1996, *everis* has averaged 20% annual growth in revenues and became part of NTT Data in January, 2014. At the time of starting the industry-academia collaboration of this PhD thesis, *everis* had offices in 12 countries. One of *everis* areas of business is Information Technology (IT) services. In IT services, *everis* designs and implements technological solutions (among them SRAs) and manages applications, infrastructures and outsourcing processes[2].

1.2.2 Software Reference Architectures in *everis*

Having seen the general context of SRAs in the industry and their importance, in this subsection we analyze the context of SRA projects from our experience with *everis*. As a consulting company, *everis* offers solutions for big businesses (e.g., banks, insurance companies, public administration, utilities, and industrial organizations) that provide a wide spectrum of services to their clients. Given the complexity of the resulting software applications, which integrate bespoke applications with commercial packages, these organizations need high-quality software architectures, and this is the service that they hire to *everis*. The solution provided by *everis* is based on the adoption of an SRA in the client organization, from which concrete software architectures are derived and used in a wide spectrum of applications.

We focus on the case in which *everis* designed an SRA with the purpose of deriving concrete architectures for each application of a client organization. This usually happens when *everis* is regularly contracted to create or maintain information systems in client organizations. Each information system of a client organization is built upon the SRA and includes many software applications. SRAs enable reuse of architectural knowledge and software components (normally associated to particular technologies) for the design of concrete architectures in client organizations. Therefore, SRAs provide a baseline that facilitates standardization and interoperability as well as the attainment of business goals during applications' development and maintenance.

Besides, a special characteristic of *everis* is its previous experience in multiple SRA projects. This experience allows *everis* to build and use a more abstract industry reference model. This reference model includes best practices from

[2]*everis'* site: `http://www.everis.com/usa/en-US/about-everis/the-company/Paginas/the-company.aspx`

previous successful experiences, which serve as a reference for new SRAs that inherit a certain level of quality. Details of the type of projects and stakeholders at *everis* can be found in Section 2.4, and Figure 2.8.

The context of *everis* is very similar to other IT consulting firms. As a recent Gartner report shows, IT consulting firms "leverages industry-specific or industry reference models to accelerate client delivery and ensure quality and consistency across client engagements" [21]. However, "clients must ensure that generic industry or reference models [...] are sufficiently customized and tailored to enable their unique business capabilities and environments" [21], so that the reference model does not stifle competitive advantage of the SRA.

1.3 Research Problem

As we have seen in Section 1.1.1, the adoption of an SRA might lead to plenty of theoretical benefits (e.g., standardization in a software family). However, it also implies several challenges, such as the ability to get real evidence for driving its design and use [22], and the need for an initial investment [5]. Currently, organizations have little support for dealing with these two challenges. This problem originates from the specific features of SRAs with respect to software architectures [23], such as the need of an initial investment, their generic nature, the wide group of stakeholders that they involve, their high level of abstraction, or their instantiation in the organization's portfolio of software systems. Therefore, as Angelov et al. point out, **practitioners face difficulties in working with SRAs** [22]. This PhD thesis aims to cope with the following two problems to help practitioners in their daily work with SRAs.

First, **there is a shortage of experience reports about the context of SRAs in industry and how they are currently being designed and used**. For instance, a recent literature review about evidence in software architecture, in which only two papers were about SRAs, shows that there is limited knowledge about SRAs [24]. As a result, academics' perspective of SRA is not always in line with the industry's practice, and practitioners usually find the current literature about SRA scarce and abstract [4], limiting the industrial uptake of research results in the field. This situation triggers the following questions:

- How can an organization get corporate evidence to support SRA engineering[1]?

[1]Throughout this document, we use the term "SRA engineering" to refer to common practices in SRA projects, such as defining the goals of an SRA, SRA design, SRA evaluation, and SRA use.

- How different stakeholders perceive the potential benefits and drawbacks of SRAs?
- Which are the artifacts that compose an SRA in the industrial practice and what is their potential reuse across domains?

In this scenario, we argue that in order to enable practitioners to fully exploit the benefits of SRA adoption and usage, the research community must clarify the diverse contexts of SRA in practice, as well as the characteristics (e.g., benefits, drawbacks, and challenges) of such contexts. This situation could be addressed by conducting empirical studies to accumulate real evidence and understand the context of SRAs from essential types of stakeholders. Such evidence might help practitioners to better understand SRA engineering and, then, to identify the current challenges to improve these engineering practices in their organization.

Second, **there is a shortage of economic models to precisely evaluate the benefit of SRAs in order to make informed decisions about their adoption in an organization** [5]. Organizations with a wide portfolio of applications, which may consider adopting an existing or new SRA to create and maintain such applications, lack an approach to know whether it is worth for them to invest on the adoption of an SRA. This situation triggers specific questions that have not been addressed yet:

- Is it worth to invest on the adoption of an SRA?
- How is it possible to calculate the Return-on-Investment (ROI) of the adoption of an SRA in an organization?
- Which commonly available data do organizations have to quantitatively calculate the costs and benefits of adopting an SRA in an organization?
- Which are the cost and benefit factors of acquiring an SRA in an organization?

This situation could be addressed by making a business case with the help of an economic model that perform cost-benefit analysis about the adoption of an SRA [25].

1.3.1 The Problem at the "Cátedra everis-UPC" Project

everis commissioned our research group two main tasks (respectively aligned with the two research problems defined above):

1. Gathering Evidence of SRAs (technical): systematically gathering evidence of SRAs projects that they conducted at their client organizations. The objective is identifying strengths and weaknesses of SRA engineering in SRA projects in order to disseminate and improve them.

2. Building the Business Case for SRAs (strategic/organizational): building the business case and calculating the ROI that their client organizations get after adopting an SRA. It aims to provide quantitative evidence to its clients about the potential economic benefits of applying an SRA.

Next, we describe the rationale of these two tasks, and their importance for *everis*.

Gathering Evidence of SRAs

The architecture group of *everis* wanted to capture empirical evidence and the architectural knowledge of years of work in a congruent vision, so that it can help *everis'* employees in the inception, design, and application of both current and prospective SRAs. To gather such empirical evidence, it is necessary to contact SRA stakeholders and *everis'* employees [26].

In this context, it becomes necessary to support practitioners in their daily work when working with SRAs.

Building the Business Case for SRAs

The architecture group of *everis* experienced the inability to calculate the ROI derived from SRAs that they create (or plan to create) for client organizations. Reifer defines a business case as the "materials prepared for decisions makers to show that the idea being considered is a good one and that the numbers that surround it make financial sense" [25]. That is, business cases enable to justify investments in technology. Spending in the adoption of an SRA without a previous and trustworthy analysis seems to be reckless and can lead to a disaster.

In the SRA context, an economic model is needed to help making business cases. An economic model should take into account costs, benefits, risks, and schedule implications. An economic model to perform cost-benefit analysis on the adoption of software reference architecture is a key asset for optimizing architectural decision-making.

To sum up, software reference vendors (e.g., software companies and information technology consulting firms such as *everis*) and acquisition organizations lack of support for dealing with these two problems: gathering empirical evidence and building a business case for SRA engineering. Next, we show the goal of this PhD thesis, which attempts to ameliorate these two problems present in *everis* and other companies with a similar context.

1.4 Research Goal

Having seen the importance of SRAs in the industry and the research problems described at Section 1.3, this section presents the objectives of the PhD thesis. The main goal of this PhD thesis is to package the knowledge and evidence gathered during our industry-academia collaboration in order:

To support organizations making informed decisions about SRAs acquisition, design, and use based on empirical evidence.

In this context, this PhD thesis supports organizations to deal with the following Research Questions (RQ):

RQ 1: How can an organization get corporate evidence that is useful for the SRA engineering? The objective of the RQ 1 consists of gathering, increasing and disseminating empirical evidence about relevant aspects of SRAs to improve SRA engineering practices in an organization. Among the relevant aspects of SRAs, we have focused on how different stakeholders perceive the potential benefits and drawbacks of SRAs, and which are the artifacts that compose an SRA in the industrial practice as well as their potential reuse across domains.

RQ 2: Is it worth for an organization to invest on the adoption of an SRA? The objective of the RQ 2 is to provide guidelines to support organizations to quantitatively analyze if it is worth to adopt an SRA. Such an objective consists of constructing an economic model for SRAs that enables to make a business case for financial analysis. This analysis optimizes the decision-making process when studying whether to make the strategic move to adopt an SRA in an organization.

1.5 Research Methodology

In order to answer our RQs, we have performed an empirical research. Empirical research is a way of gaining knowledge by means of direct and indirect observation or experience[3]. One of the objectives of Empirical Software Engineering (ESE) is to gather and utilize evidence to advance software engineering methods, processes, techniques, and tools [27]. Since empirical studies help to solve the industrial problems, **this PhD thesis fosters the conduction of empirical studies as a way to increase the empirical evidence about SRAs**.

We followed an action research approach. Action research is "learning by doing": a group of people identify a problem, do something to resolve it, see how successful their efforts are, and if not satisfied, try again [28]. The action research cycle consists of five steps (see Figure 1.2):

Step 1: diagnosis of a problem,

Step 2: examination of options to solve the problem,

Step 3: selection of options and execution,

Step 4: analysis of the results, and,

Step 5: repetition for improvement.

Due to the practical nature of this PhD, which is highly bound to the software industry, and to the data that we needed to gather and analyze from *everis* and other organizations, ESE studies are ideal to solve the research problem stated in Section 1.3. The empirical studies of our action research process drove the establishment of the guidelines to gather empirical evidence and to build a business case for SRAs. We distinguish among two stages in our research: formative and summative.

First, the formative stage involves the evolution of the research work. The central role of the formative stage is shaping the proposed empirical studies and guidelines. It is called formative because it serves as the origin and evolution of the ideas and concepts presented in this PhD thesis. The last step of our formative stage is packaging guidelines with the aim of being applied in prospective SRA projects and also in similar organizations.

Second, once the guidelines were adequately shaped and improved, the summative stage aims to evaluate them. The central role of the summative

[3]https://en.wikipedia.org/wiki/Empirical_research

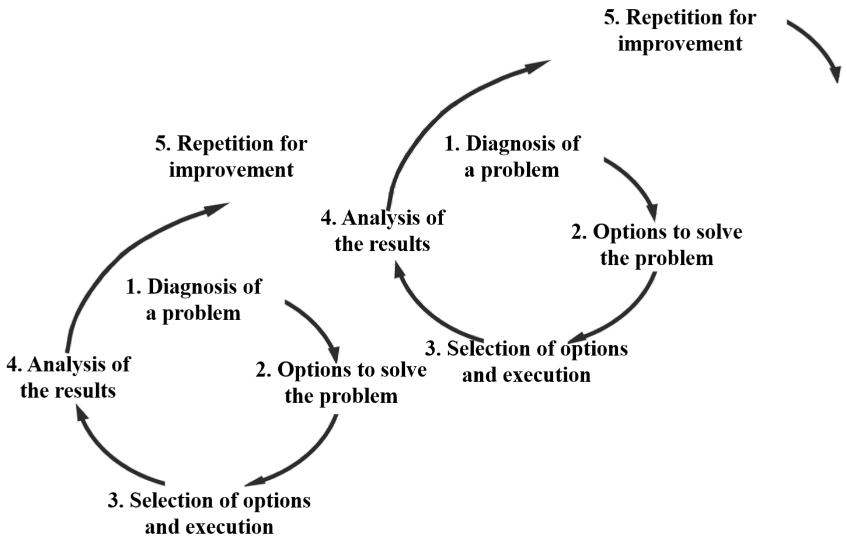
5. Repetition for improvement
5. Repetition for improvement
1. Diagnosis of a problem
4. Analysis of the results
2. Options to solve the problem
1. Diagnosis of a problem
3. Selection of options and execution
4. Analysis of the results
2. Options to solve the problem
3. Selection of options and execution

Figure 1.2: The five steps of an action research approach.

stage is integrating successful results from formative stages, and to validate them with practitioners (which may come from *everis* or even other organizations). Such validation consists of using the guidelines to design and conduct empirical studies. Organizations facing the design and use of SRAs, and analyzing whether to make the strategic move to SRA adoption based on evidence, benefit from these guidelines.

The next subsection describes the tasks of the action research initiative of this PhD thesis. All empirical studies designed and conducted are a cycle inside our action research initiative, and belong to either the formative or summative stages.

1.5.1 Tasks and Results

Having established the RQs, the tasks and results of this PhD thesis can be grouped in four parts:

- Part I: Incubation of the PhD thesis.

- Part II: Gathering empirical evidence of SRAs (RQ1).
- Part III: Building the business case for SRAs (RQ2).
- Part IV: Discussions and conclusions.

Part I: Incubation of the PhD thesis

The starting point of this PhD thesis was its incubation, which ended up with the PhD proposal [29, 30]. This incubation had four main tasks (see Table 1.1).

First, this PhD started by analyzing the problems identified in our action research collaboration with *everis* (see Section 1.3.1). During this task, we identified the research gaps about SRAs and propose the research goals of this PhD thesis.

Second, we studied the basic background on SRAs (see Chapter 2). In this stage, we stated the borders of SRA with regard to similar architectural concepts, such as enterprise reference architecture and product line architecture.

Third, to have an unbiased view of the state-of-the-art and current research on SRA engineering, we designed and conducted a Systematic Literature Re-

Table 1.1: Initial tasks of the PhD thesis

Stage	Task	Main Results of the Task	Ref.	Ch.
Formative stage	Establishing the research gap and the research goals	Identification of the problem at *everis*	[31]	1
Formative stage	Studying the background on SRAs	Identification of concepts related to SRA, and SRA theory	[29]	2
Formative stage	Performing a state-of-the-art on SRA engineering	A holistic overview of the existing techniques and approaches oriented to support SRA engineering	[32]	3
Formative stage	Definition and use of a reference model for predictive service selection	QuPreSS, a reference model which measures predictive service quality and guides the selection of the most accurate predictive service, and a tool based on QuPreSS (which is called Mercury)	[33, 34, 35, 36]	-

view (SLR), which is presented in Chapter 3. This work was jointly done with the LabES research group from the University of Sao Paulo (Brazil).

Fourth, we needed to conduct empirical studies to observe how practitioners from *everis* design and use reference models and SRAs. However, we had no experience on designing and using reference models nor SRAs. For this reason, we decided to design QuPreSS, a reference model for predictive services selection. Also, we used QuPreSS to create Mercury, a tool for evaluating predictive services customized to the weather forecast domain. This work has been jointly performed with a researcher from the Universitat Pompeu Fabra.

Part II: Gathering empirical evidence of SRAs (RQ1).

This part details our action research initiative with regard to the RQ 1 (defined in Section 1.4). To answer RQ 1, we conducted five tasks (see Table 1.2). These tasks are cycles of the action research conducted in order to answer RQ 1. Such cycles are depicted in Figure 1.3, which explains the formative stage to create the guidelines to gather relevant evidence from SRA projects (above), and how such guidelines were validated in the summative stage (below).

In the first cycle of the RQ 1, we diagnosed the need of knowing about the state of past and current SRA projects in *everis* in order to reuse architectural knowledge and improve SRA engineering. As a consequence, we planned to identify a set of criteria about SRAs that are relevant for practitioners. As a result, we identified five aspects that indicate what evidence to gather in order to support SRA engineering, which are presented in Chapter 4.

In the second cycle of RQ 1, we planned to gather evidence about the aspects identified in the first cycle. We designed interview guides and on-line questionnaires to gather mostly qualitative evidence about such aspects. Then, we executed them in a case study with several SRA projects in *everis*. As a consequence, we obtained results about why *everis'* clients adopted SRAs, the artifacts of SRAs, and the benefits and drawbacks of SRAs. These results are reported in Chapter 5.

With the above two tasks/cycles, which belong to the formative stage, we shaped the template surveys to gather empirical evidence. The guidelines to gather empirical evidence of SRAs are presented in Chapter 6.

Then, at the third cycle of the RQ 1, we needed to validate these guidelines and their template surveys. We decided to execute the same survey out of our industry-academia collaboration with *everis*. Then, as part of the validation and summative stage, we conducted a survey to gather empirical evidence about the benefits and drawbacks of AUTOSAR, a mature and accepted SRA

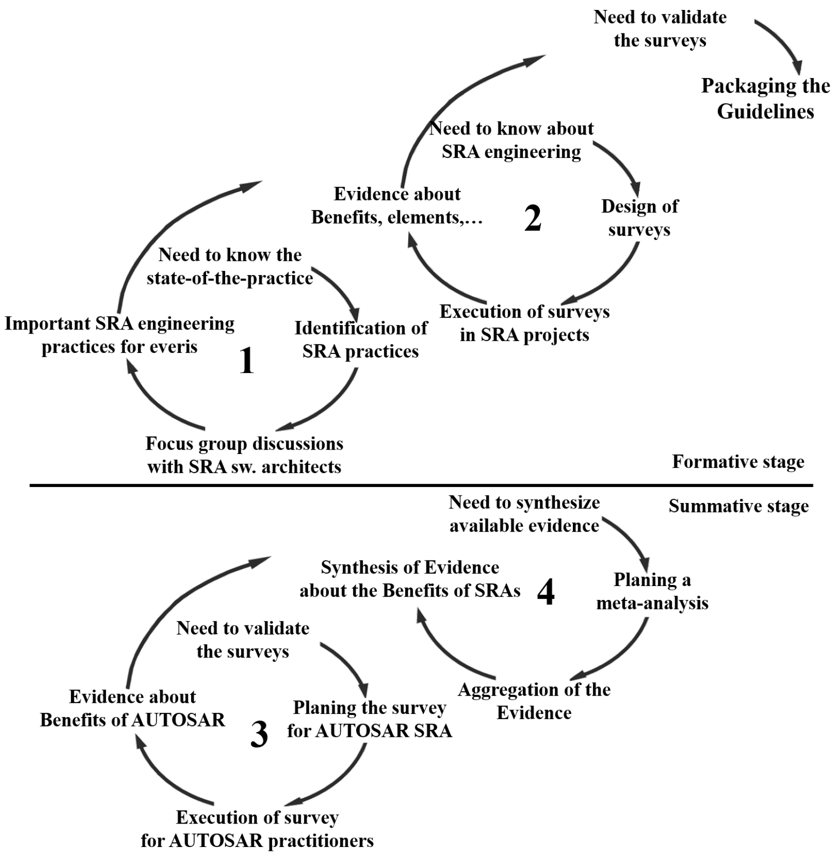

Figure 1.3: Action-research cycles of RQ 1.

Table 1.2: Tasks related to gathering empirical evidence of SRAs (RQ1).

RQ (Stage)	Task	Main Results of the Task	Ref.	Ch.
RQ1 (formative stage)	Meetings to study the relevant criteria of SRAs for an organization	A list of aspects that may be important for organizations to support SRA engineering	[26, 31]	4
RQ1 (formative stage)	Design and execution of case studies to gather evidence to improve SRA engineering in an organization	Template questionnaires to gather evidence and understand the impact of using SRAs for designing the concrete architectures of software applications in an organization	[37, 38]	5
RQ1 (formative stage: packaging)	Packaging the guidelines for many organizations	Guidelines for gathering empirical evidence of SRAs in industry	[39, 26, 29, 30, 31]	6
RQ1 (summative stage)	A survey on the benefits and drawbacks of a mature and accepted SRA used worldwide	Evidence on the benefits and drawbacks of AUTOSAR, an SRA used by more than 180 organizations, and directions to handle its major drawbacks	[40]	7
RQ1 (summative stage)	A meta-analysis of SRA benefits and drawbacks	Aggregation of the available empirical evidence of SRA benefits and drawbacks	[41]	8

for automotive applications used worldwide by more than 180 organizations. This work has been jointly done with the University of Sao Paulo (Brazil). The results are depicted in Chapter 7.

Despite our work on the benefits and drawbacks of SRAs at *everis* and AUTOSAR, such benefits and drawbacks have also been studied in other contexts by other researchers. Therefore, to strengthen current evidence on SRAs, it became necessary to aggregate the empirical evidence from *everis* with the empirical evidence from other works. We perform a research synthesis with the results from all these works. This task has been jointly performed with the Federal University of Rio de Janeiro (Brazil), and is presented in Chapter 8.

Part III: The business case for SRAs (RQ2).

This part details our action research initiative with regard to RQ 2, which is intended to quantify the benefits and costs of SRAs through a business case. To answer RQ 2, we conducted four tasks (see Table 1.3). These tasks are cycles of the action research conducted in order to answer RQ 2. Such cycles are depicted in Figure 1.4, which explains the formative stage to create the guidelines to build the business case for SRAs (above), and how such guidelines were validated in the summative stage (below).

In the first cycle of the RQ 2, we diagnosed the problem of the lack of approaches to justify the investment on SRAs to *everis'* clients in monetary terms. As a consequence, we designed online questionnaires to ask stakeholders the metrics available in SRA projects and conducted them in several SRA projects from *everis*. As a result, we observed that effort metrics could be derived from time tracking practices and that cost-benefit factors could be computed

Table 1.3: Tasks related to the business case for SRAs (RQ2)

RQ (Stage)	Task	Main Results of the Task	Ref.	Ch.
RQ2 (formative stage)	A survey to check existing value-driven data in SRA projects	Identification of quantitative information that can commonly be retrieved in SRA projects in order to quantitatively calculate the costs and benefits of adopting an SRA in an organization.	[31]	9
RQ2 (formative stage)	Design of an economic model to calculate the ROI of adopting an SRA	REARM, an economic model to perform cost-benefit analysis on the adoption of SRAs as a key asset for optimizing architectural decision-making	[42, 5]	10
RQ2 (formative stage: packaging)	Packaging the guidelines for many organizations	Guidelines for building a business case for SRAs in industry	[39, 26, 29, 30, 31]	11
RQ2 (summative stage)	Workshops to get feedback to evaluate REARM	Validation of REARM and identification of future work	-	12

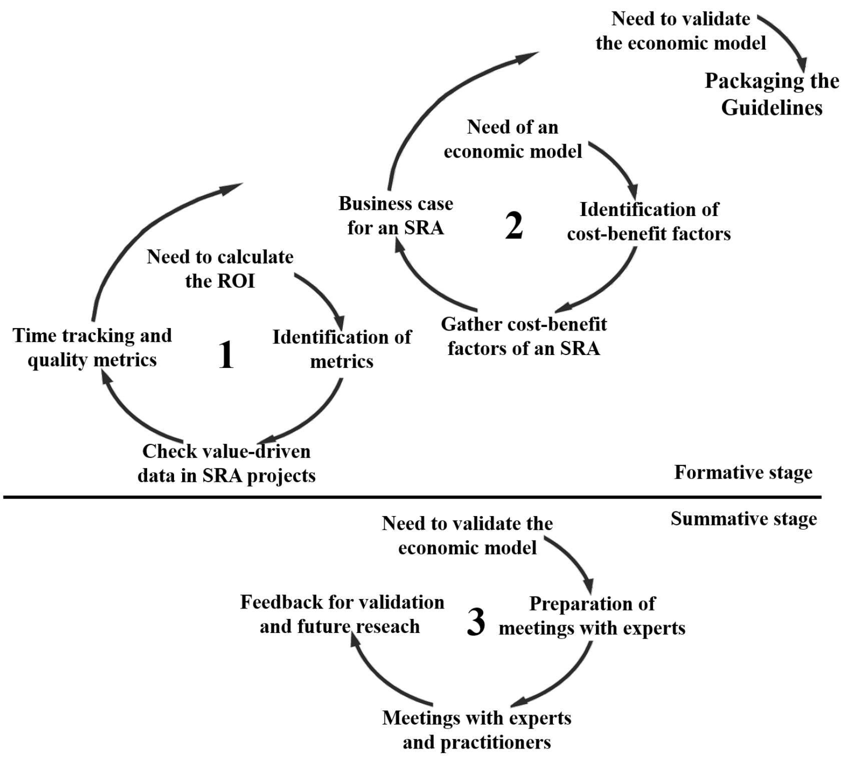

Figure 1.4: Action-research cycles of RQ 2.

by using reuse-based metrics from the source code of the SRA projects. The results from the online questionnaires to study the metrics available in SRA projects are reported in Chapter 9.

In the beginning of the second cycle of RQ 2, we diagnosed the need of having an economic model that would use the available data. Then, we identified economic functions meaningful to *everis* to justify SRA investments. We also identified cost-benefits factors from the literature that can be calculated from the available metrics. Then, we computed these cost-benefit factors for a real SRA project. At the end of this second cycle, we could build the business case for that SRA project. In Chapter 10, we report an application of REARM, whose acronym comes from REference ARchitecture Model, an economic model to quantitatively analyze the adoption of SRAs in organizations.

The above two first tasks/cycles are in the formative stage, because their feedback contributed to incrementally design the guidelines to build a business case. For this reason, these two tasks were best characterized as formative studies due to their central role in shaping the guidelines. The guidelines to build the business case for SRAs are presented in Chapter 11.

Once the guidelines were packaged, it was necessary to validate the results got and to analyze lessons learned. The third cycle of RQ 2 is best characterized as summative, since it focus on validating the economic model and identifying further areas of improvement (see Chapter 12).

Part IV: Discussions, conclusions, and future work.

In this part, we present the lessons learnt in our industry-academia collaboration, and the conclusions of this PhD thesis. The two tasks of this part are shown in Table 1.4.

Table 1.4: Tasks related to discussions and conclusions.

Stage	Task	Main Results of the Task	Ref.	Ch.
Summative stage	Reporting on practical experiences of our industry-academia collaboration	Evaluation of the success of the industry-academia collaboration of this PhD, and reporting the experience with conducting empirical studies in the industry and lessons learnt	[43]	13
Summative stage	Wrapping up with conclusions and future work	Summarizing the contributions of this PhD thesis, and identifying gaps for future work	-	14

First, we evaluated our collaboration with an existing model for technology transfer, and performed a focus group discussion to identify challenges that we faced. The results are presented in Chapter 13.

Second, we wrapped up this document by stating the main contributions of this PhD thesis, and identifying the research gaps for future work. The conclusions are presented in Chapter 14.

1.6 List of Publications

This section enumerates and summarizes the work that has been published. For a better representation of the work done, in what follows, we show these publications grouped by the type of publications: journals, conferences, doctoral symposiums, workshops, technical reports, other publications, and ongoing journals (which have either been submitted or will be submitted soon).

Among the 12 peer-reviewed publications, the most notable ones are: **1 JCR indexed Elsevier journal, 1 Springer journal, 3 full-papers at the main track of CORE-A conferences, and 1 best paper award** (at the ESELAW workshop).

Journals

1. S. Martínez-Fernández, X. Franch, and J. Bisbal, "Mercury: Using the QuPreSS Reference Model to Evaluate Predictive Services," *Journal of Science of Computer Programming (SCP)*, To appear [JCR 2014, Q3, IF: 0.715]. [36]
2. S. Martínez-Fernández, C. P. Ayala, X. Franch, H. Martins Marques, and D. Ameller, "Towards Guidelines for Building a Business Case and Gathering Evidence of Software Reference Architectures in Industry," *Journal of Software Engineering Research and Development (JSERD)*, vol. 2, iss. 7, 2014. A SpringerOpen Journal. [31]

Conferences

1. S. Martínez-Fernández, P. S. Medeiros Dos Santos, C. Ayala, X. Franch, and G. H. Travassos, "Aggregating Empirical Evidence about the Benefits and Drawbacks of Software Reference Architectures," in *Proceedings of the ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2015, pp. 154-163 [CORE2013: A]. [38]
2. S. Martínez-Fernández, C. Ayala, X. Franch, and H. Martins Marques, "Artifacts of Software Reference Architectures: A Case Study," in *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering (EASE)*, 2014, p. 42:1-42:10. [CORE2013: A]. [38]
3. S. Martínez-Fernández, C. Ayala, X. Franch, and H. Martins Marques, "REARM: A Reuse-Based Economic Model for Software Reference Architectures," in *13th International Conference on Software Reuse (ICSR)*, 2013, pp. 97-112. [CORE2013: A]. [5]

4. S. Martínez-Fernández, C. Ayala, X. Franch, and H. Martins Marques, "Benefits and Drawbacks of Reference Architectures," in *7th European Conference on Software Architecture (ECSA)*, 2013, pp. 307-310 (research in progress short paper). [CORE2013: A]. [37]

5. S. Martínez-Fernández, J. Bisbal, and X. Franch, "QuPreSS: A Service-Oriented Framework for Predictive Services Quality Assessment," in *7th International Conference on Knowledge Management in Organizations: Service and Cloud Computing (KMO)*, 2012, pp. 525-536. [34]

Doctoral Symposium

1. S. Martínez-Fernández, "Towards Supporting the Adoption of Software Reference Architectures: An Empirically-Grounded Framework," in *11th International Doctoral Symposium on Empirical Software Engineering (IDoESE) hosted at the International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2013. [CORE2013: A]. [30]

Workshops

1. S. Martínez-Fernández, C. P. Ayala, X. Franch, and E. Y. Nakagawa, "A Survey on the Benefits and Drawbacks of AUTOSAR," in *Proceedings of the 1st International Workshop on Automotive Software Architecture (WASA)*, 2015, pp. 19-26. [40]

2. S. Martínez-Fernández, C. Ayala, X. Franch, and H. Martins Marques, "Practical Experiences in Designing and Conducting Empirical Studies in Industry-Academia Collaboration," in *Proceedings of the 2nd International Workshop on Conducting Empirical Studies in Industry (CESI)*, 2014, pp. 15-20. [43]

3. S. Martínez-Fernández, X. Franch, and J. Bisbal, "Verifying predictive services' quality with Mercury," in *4th International Workshop on Academic Software Development Tools and Techniques (WASDeTT)*, 2013. [35]

4. S. Martínez-Fernández, C. Ayala, X. Franch, H. Martins Marques, and D. Ameller, "A framework for software reference architecture analysis and review," in *Memorias del X Workshop Latinoamericano de Ingeniería en Software Experimental (ESELAW)* - ISBN 978-9974-8379-3-5, 2013, pp. 89-102. **Best paper award!!** [26]

Technical Reports

1. S. Martínez-Fernández, C. Ayala, and X. Franch, "A Reuse-Based Economic Model for Software Reference Architectures," Departament ESSI. Universitat Politècnica de Catalunya (UPC). BarcelonaTech, ESSI-TR-12-6, 2012. [42]

2. S. Martínez-Fernández, D. Ameller, C. Ayala, X. Franch, and X. Terradellas, "Conducting Empirical Studies on Reference Architectures in IT Consulting Firms," Departament ESSI. Universitat Politècnica de Catalunya (UPC). BarcelonaTech, ESSI-TR-12-2, 2012. [39]

Other publications

1. S. Martínez-Fernández, "A Framework for Software Reference Architecture Analysis and Review," *PhD Proposal*. Universitat Politècnica de Catalunya, 2013. Advisors: Xavier Franch and Claudia Ayala. [29]

2. S. Martínez-Fernández, L. B. Ruas de Oliveira, C. P. Ayala, X. Franch, and E. Y. Nakagawa, "Planning a Systematic Review on Business Case for Reference Architectures," *Poster Session from Component-Based Software Engineering and Software Architecture federated conference (CompArch)*, 2014. [CORE2013: A]. [32]

3. S. Martínez-Fernández, J. Bisbal, and X. Franch, "Accuracy Assessment of Forecasting Services (poster)," *1st European Business Intelligence Summer School (eBISS)*, 2011. [33]

Ongoing Journals

1. S. Martínez-Fernández, C. P. Ayala, X. Franch, and H. Martins Marques, "Benefits and Drawbacks of Software Reference Architectures: A Case Study," *ACM Transactions on Software Engineering and Methodology Journal (TOSEM)*, **(submitted, currently under review)**.

2. S. Martínez-Fernández, L. Bueno, L. Garcés, C. P. Ayala, X. Franch, and E. Y. Nakagawa, "Reference Architecture Engineering: A Mapping Study," *Journal to be decided*, **(to be submitted)**.

3. S. Martínez-Fernández, P. S. Medeiros Dos Santos, C. Ayala, X. Franch, and G. H. Travassos, Extension of the conference paper "Aggregating

Empirical Evidence about the Benefits and Drawbacks of Software Reference Architectures,"[38] *Journal to be decided*, **(to be submitted)**.

Besides the publication of research papers, I have used my personal page of UPC as an open science tool: `http://www.essi.upc.edu/~smartinez/`. Throughout the course of this PhD thesis and as it progressed, I have uploaded pre-prints and publications to disseminate the work. Therefore, author versions of all these publications can be downloaded from there.

1.7 Structure of this Document

The main contributions of this PhD thesis are divided in four parts (see Section 1.5.1): introduction, RQ1, RQ2, and conclusions:

1. Part I shows this introduction, basic concepts related to SRA, and an SLR with the current state-of-the-art in SRA engineering (see Table 1.1).
2. Part II presents the five tasks about gathering empirical evidence to support the design and use of SRAs in an organization (see Table 1.2).
3. Part III discusses the four tasks related to build the business case for SRAs (see Table 1.3).
4. Part IV presents discussions about our industry-academic collaboration, and ends up with conclusions and future work (see Table 1.4).

Finally, this document includes bibliographic information, and several appendices: a glossary (see Appendix A), the included studies in the SLR (see Appendix B), and materials to support the guidelines presented (see Appendix C and Appendix D).

Chapter 2

Background

This chapter presents basic concepts related to SRA. The first section introduces the different disciplines around the concept "architecture". The second section includes the rudiments of SRAs (i.e., definitions, types, and elements). The third section defines the boundaries of concrete architectures and product line architectures with regard to SRAs. Finally, the fourth section provides the *everis* industrial view about the types of projects and stakeholders.

2.1 Architecture Disciplines Basic Concepts

The term "architecture" has been used extensively, but not always together with software. Two architecture disciplines related to software architecture are system architecture and enterprise architecture [1]. "A *system's architecture* is a representation of a system in which there is a mapping of functionality onto hardware and software components, a mapping of the software architecture onto the hardware architecture, and a concern for the human interaction with these components" [1]. "*Enterprise architecture* is a description of the structure and behavior of an organization's processes, information flow, personnel, and organizational subunits, aligned with the organization's core goals and strategic direction" [1].

Next, we discuss the relationships and boundaries between these three architecture disciplines, and where SRA belongs to.

2.1.1 Relationships Between Architecture Disciplines

Software architecture is different from other architecture disciplines (e.g., system architecture and enterprise architecture). The main objects of study of a software architecture are: the abstraction of a software system that consists of three components: elements, form, and rationale [44]; and the set of significant decisions about the organization of such software system [45].

However, system architecture and enterprise architecture "have broader concerns than software and affect software architecture through the establishment of constraints within a software system must live" [1]:

- a system architecture is concerned with a total system, including hardware, software and humans [1].
- an enterprise architecture is concerned with how an enterprise's software systems support the business processes and goals of the enterprise [1].

To sum up, software is only one concern of system architecture and enterprise architecture.

In spite of being different architecture disciplines, software architecture and system architecture share their support to software systems. Inside the context of enterprise architecture, they are indistinguishably referred to as *solution architectures*. The Open Group Architecture Framework (TOGAF) defines a solution architecture as "a description of a discrete and focused business operation or activity and how IS/IT supports that operation. A Solution Architecture typically applies to a single project or project release, assisting in the translation of requirements into a solution vision, high-level business and/or IT system specifications, and a portfolio of implementation tasks"[1]. Poort et al. [46] also use the term solution architecture to group various architecture "genres" with the common denominator of finding a solution to a particular set of stakeholders' needs. Such common denominator is shared by software architecture and system architecture, so we can consider that both of them are solution architectures.

[1]Definitions TOGAF: `http://pubs.opengroup.org/architecture/togaf9-doc/arch/chap03.html`

2.1.2 Where Software Reference Architecture Belongs To

A reference architecture provides a prescriptive way (a template solution) for an architecture for a particular domain [47, 48, 49]. Reference architectures can be found in the aforementioned architecture disciplines, leading to:

- SRA, such as RACE [50], which addresses the engine software for document processing systems.
- System reference architecture, such as a distributed system reference architecture for adaptive QoS and resource management [51].
- Enterprise reference architecture, such as PERA [52], which is a complete enterprise reference architecture as defined by the IFAC/IFIP Task Force on Enterprise Integration.

Figure 2.1 shows the relationship between software architecture, system architecture and enterprise architecture. There are three achitecture disciplines: software architecture, system architecture and enterprise architecture. The solution architecture includes two architecture disciplines (software architecture and system architecture) as it is seen in the enterprise architecture context. SRA is a sub field inside the software architecture discipline (reference architectures are also studied in the system and enterprise disciplines).

Although all of them are different architecture disciplines, they are interconnected, e.g., "the software architect for a system should be on the team that provides input into the decisions made about the system or the enterprise" [1]. For example, an enterprise could adopt an enterprise architecture as an strategic activity, but needs also to have a solution architecture (i.e., system architecture and/or software architecture) that deepens in the structure of a each system. This scenario is depicted in Figure 2.2. Another example is that the software architecture of a system needs to be in compliance with the system architecture (e.g., software systems' technologies need to be compatible with the hardware architecture).

This PhD thesis focuses on SRAs, which are inside the software architecture field of research.

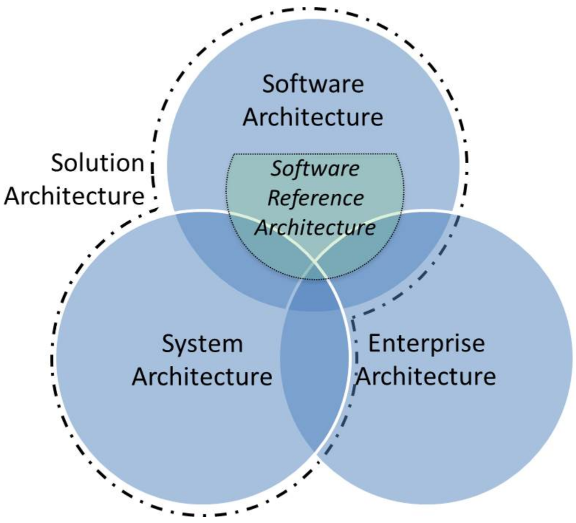

Figure 2.1: Relationship between architecture disciplines.

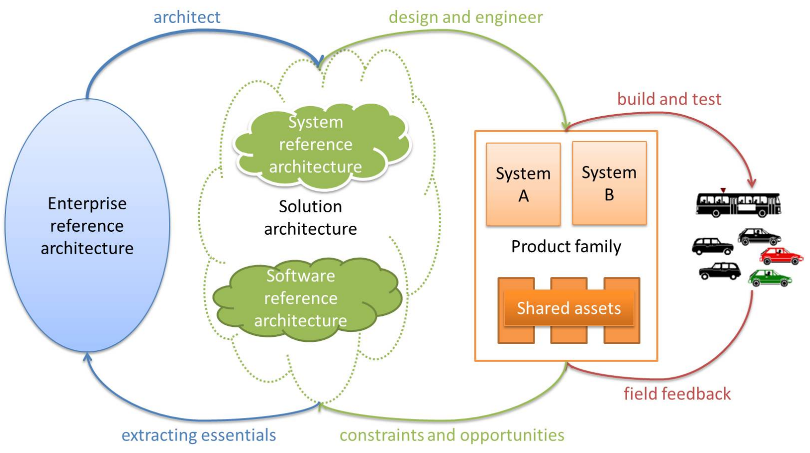

Figure 2.2: Co-existence of reference architectures from different architecture disciplines: enterprise, system and software (adapted from [3]).

2.2 Software Reference Architecture Essentials

This section focuses on SRA and studies:

- the definition of SRA;
- the types of SRAs; and,
- the elements of SRAs.

2.2.1 Definition of Software Reference Architecture

As we have already stated in Chapter 1.1, Nakagawa et al. [2] define an SRA as "an architecture that encompasses the knowledge about how to design concrete architectures of systems of a given application [or technological] domain; therefore, it must address the business rules, architectural styles (sometimes also defined as architectural patterns that address quality attributes in the reference architecture), best practices of software development (for instance, architectural decisions, domain constraints, legislation, and standards), and the software elements that support development of systems for that domain. All of this must be supported by a unified, unambiguous, and widely understood domain terminology".

SRAs are attractive when enterprises have many software systems that have very similar structure and share a technological or business domain. Then, a common SRA to such software systems can be designed. The SRA defines a standard structure of systems.

2.2.2 Types of Software Reference Architecture

Angelov et al. [4] distinguish between five types of SRAs. They define a multi-dimensional space to classify these types of SRAs, which is composed of 3 main dimensions: context (C1), goal (G1), and design (C3). Next, all dimensions and their possible values are summarized:

- Context dimension (C)
  - C1: Where will the SRA be used? Values: single organization, multiple organizations.
  - C2: Who defines the SRA? Values: software groups, user groups, and independent groups.

  - C3: When is the SRA defined? Values: preliminary, classical.

- Goal dimension (G)
  - G1: Why is the SRA defined? Values: standardization, facilitation.
- Design sub-dimensions (D)
  - D1: What is described in the SRA? Values: components and connectors, interfaces, protocols, algorithms, policies and guidelines.
  - D2: How detailed is it described? Values: detailed, semi-detailed, and aggregated.
  - D3: How concrete is it described? Values: abstract, semi-concrete, and concrete.
  - D4: How is it represented? Values: informal, semi-formal, formal.

The values of an SRA are mutually exclusive for the G1, C1, and C3 sub-dimension (i.e., an SRA can be attributed only one value from these sub-dimensions). It leads to five congruent types of SRAs [4]:

- Type 1) Classical, standardization SRAs to be implemented in multiple organizations;
- Type 2) Classical, standardization SRAs to be implemented in a single organization;
- Type 3) Classical, facilitation SRAs for multiple organizations designed by a software organization in cooperation with user organizations;
- Type 4) Classical, facilitation SRAs designed to be implemented in a single organization;
- Type 5) Preliminary, facilitation SRAs designed to be implemented in multiple organizations.

2.2.3 Elements that Compose an SRA

RAModel is a reference model for SRAs that shows possibly all elements, organized by types and relationships, which could be contained in an SRA. RAModel is depicted in Figure 2.3. As Figure 2.3 shows, these elements are inside one of the following four groups:

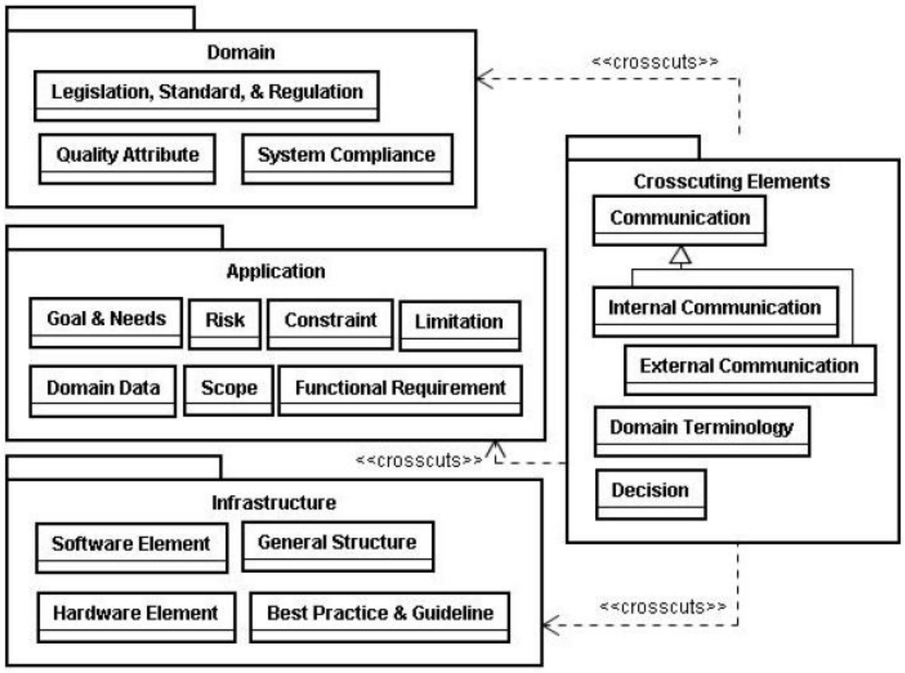

Figure 2.3: RAModel: Reference model for SRAs [13].

- Domain: It contains elements related to self-contained, specific information of the space of human action in the real world, such as domain legislations, standards, and certification processes, which impact systems and related reference architectures of that domain;

- Application: It contains elements that provide a good understanding about the reference architecture, its capabilities and limitations. It also contains elements related to the business rules (or functionalities) that could be present in software systems built from the reference architecture;

- Infrastructure: It refers to elements that could be used to build the software systems based on the reference architecture. These elements are responsible to enable these systems to automate, for instance, processes, activities, and tasks of a given domain; and

- Crosscutting Elements: It aggregates a set of elements that are usually spread across and/or tangled with elements of other three groups (domain, application, and infrastructure). For instance, the communication (internal and external) in the software systems built from the SRA, the domain terminology, and decisions.

2.3 The Boundaries of SRAs with Respect to Related Terms

This section focuses on the boundaries of SRAs with respect to: reference models; concrete software architectures; and, product line architectures.

2.3.1 Reference Model and SRA

As defined by Bass et al. [1], reference models and SRAs are different concepts.

On the one hand, "a reference model is a division of functionality together with data flow between the pieces. A reference model is a standard decomposition of a known problem into parts that cooperatively solve the problem" [1]. They arise in mature domains in which experience has lead to a standard solution for the problem, e.g., the standards parts of a compiler or a database management system and how such parts work together to accomplish their collective purpose.

On the other hand, an SRA is "a reference model mapped onto software elements (that cooperatively implement the functionality defined in the reference model) and the data flows between them" [1]. Whereas a reference model divides the functionality, an SRA is the mapping of that functionality onto a system decomposition.

Figure 2.4 shows the relationship among reference models, SRAs, and (concrete) software architectures. The arrows indicate that subsequent concepts contain more design elements. Summarizing, "an SRA *is a set of domain concepts mapped onto a standard set of software components and relationships*" [53].

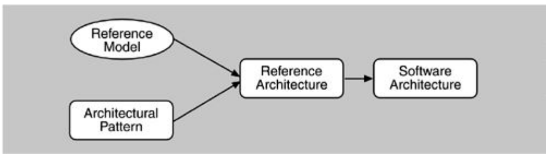

Figure 2.4: Reference models, architectural patterns, and architectures ([1]).

2.3.2 Concrete Software Architecture and SRA

There are many definitions of (concrete) software architecture. We show below three of the most cited ones. The Software Engineering Institute keeps an up-to-date list of software architecture's definitions[2].

"The software architecture of a system is the set of structures needed to reason about the system, which comprise software elements, relations among them, and properties of both" [1].

"The fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution" [54].

"An architecture is the set of significant decisions about the organization of a software system, the selection of the structural elements and their interfaces by which the system is composed, together with their behavior as specified in the collaborations among those elements, the composition of these structural and behavioral elements into progressively larger subsystems, and the architectural style that guides this organization—these elements and their interfaces, their collaborations, and their composition" [45].

The main difference between an SRA and a (concrete) software architecture is that the former one is for many software systems of a domain. The above definitions highlight the architecture of *a single system* [1, 54, 45] whereas an SRA "encompasses the knowledge about how to design concrete architectures of *systems*" [2]. Angelov et al. point out the generic nature of SRAs as a main feature distinguishing them from concrete software architectures [4]. Their generic nature implies their applicability in multiple, different contexts, reflecting the requirements of the stakeholders in these contexts. The generic nature of SRAs is achieved by designing them at higher levels of abstraction (abstracting from the differences introduced by the contexts). Thus, we can label an architecture as reference, only if it is defined to abstract from certain contextual specifics allowing its usage in differing contexts. Also, Galster et al. show that SRAs "capture the essence of the architecture of similar systems in an application or technology domain" and "can be instantiated for different contexts and at the same time support a high degree of variability in instantiated architectures" [55].

[2]Published software architecture definitions: `http://www.sei.cmu.edu/architecture/start/glossary/published.cfm`

2.3.3 Product Line Architecture and SRA

The terms software product line architecture and SRA are sometimes used indistinctly. Inside the software product line engineering context, the term SRA refers to "a core architecture that captures the high-level design for the applications of the software product line" [56, p. 124] or "just one asset, albeit an important one, in the software product line's asset base" [57, p. 12].

However, out of the software product line context, SRA and product line architecture are considered different types of artifacts [4, 58, 55, 2]. In Fig. 2.5 we show the main similarities and differences:

- Product line architectures are SRAs whereas not all SRAs are product line architectures [4], i.e., product line architectures are one type of SRAs [55]. Product line architectures are just one asset of software product lines [57, p. 12].

- SRAs are more generic and abstract than product line architectures, which are more complete architectures [4, 55]. Hence, "SRAs can be designed with an intended scope of a single organization or multiple organizations that share a certain property" [4] whereas product line architectures are produced for a single organization [55].

- SRAs provide standardized solutions for a broader domain (i.e., "spectrum of systems in a technology or application domain" [55]) whereas product line architectures provide standardized solutions for a smaller subset of the software systems of a domain [2] (i.e., "group of systems that are part of a product line" [55]). Therefore, product line architectures give a coherent and more congruent view of the products in a project (i.e., possible to track the status of) [58] whereas by means of SRAs it is more difficult to obtain congruence [4], since they can only provide guidelines for applications' development.

- Product line architectures specifically address points of variability and more formal specification in order to ensure clear and precise behavior specifications at well-specified extension points [4]. In contrast, SRAs have less focus on capturing variation points [4, 58, 2]. Although variability is not typically addressed by SRAs in a systematic manner, it is also a key fact for SRAs [10], and it can be treated as a quality attribute, rather than explicitly as 'features' and 'decisions' [10].

- SRAs include “the reuse of knowledge about software development in a given domain, in particular with regard to architectural design” [2] and dictate the patterns and principles to implement, i.e. “what the design should be” [58]. Conversely, product line architectures specifically indicate deviations, i.e. “what the design is” [58].

- SRAs include architectural knowledge and the instantiation of this architectural knowledge (i.e., reference model) into software elements [1]. In this sense, both SRAs and product line architectures are “a superset, a tool box, with every possible architecture element described, which can be used in the design of a product architecture” [58].

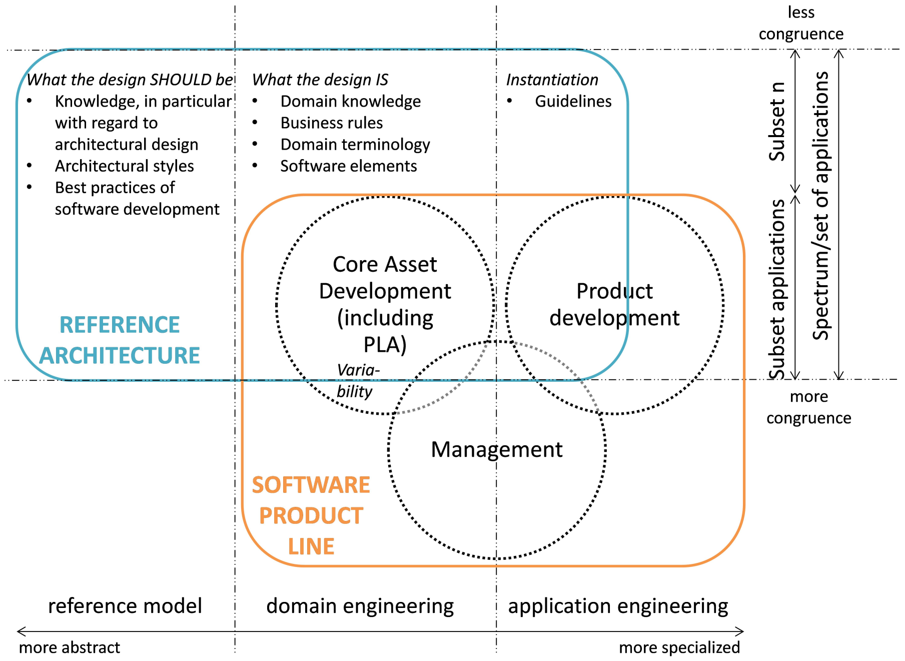

Figure 2.5: Similarities and differences between SRA, product line architectures, and software product lines [5].

2.4 The Industrial Context of SRAs in *everis*

In the previous chapter at Section 1.2, we reported a summary of the industrial context of SRAs in *everis*. In this section, we give further details and show the types of projects related to SRAs and their stakeholders.

everis supports its client organizations to design and develop their own SRAs, and to build applications on top of such SRAs. To support these tasks, *everis* uses a corporate reference model that gathers and centralizes the architectural knowledge and practices of the company. Such a corporate model was built and is continuously shaped to the usual business values and services that *everis*' clients share. Thus, the *everis* reference model supports architectural knowledge reuse in different client organizations.

Fig. 2.6 shows two different views of the *everis* reference model: execution and development. The execution view considers possible functionalities to be mapped to SRAs; whereas the development view describes software artifacts that SRAs could provide to support the development of applications. Other views, such as the physical view, are not a responsibility of SRA designers.

With the help of the *everis* reference model, *everis*' software architects share the vision of the typical elements that compose modern information systems from their clients. Then, they can analyze architecturally-significant requirements to decide which functionalities should be mapped to the SRA of their clients. As an example, Fig. 2.7 shows an SRA of a public administration in Spain, available at `http://canigo.ctti.gencat.cat/canigo/framework/`. We can see that some functionalities of the *everis* reference model have been 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 specific architecturally-significant requirements of the client organization.

In the *everis* context, we can find three types of projects with different targets, which are defined in the next subsections (see Figure 2.8): *reference model projects*; *SRA projects*; and *concrete architecture projects*.

Different stakeholders participate in each type of project. Stakeholders need to be clearly defined for SRA engineering practices (e.g., assessment) [23]. In the three types of projects defined above performed by *everis*, we consider the following five stakeholders essential for studying SRA engineering

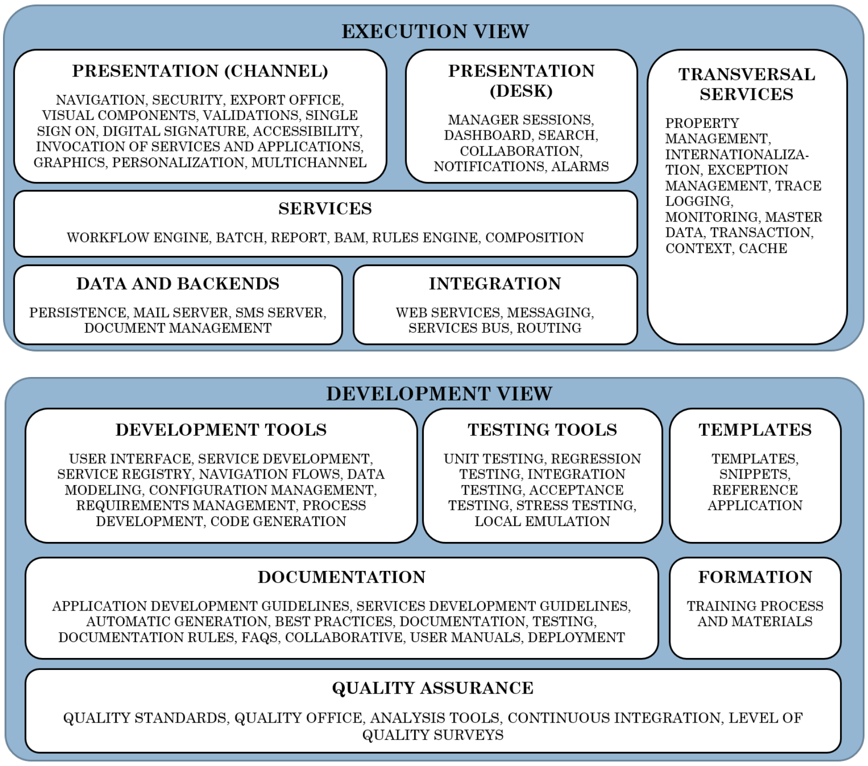
EXECUTION VIEW
PRESENTATION (CHANNEL)
NAVIGATION, SECURITY, EXPORT OFFICE, VISUAL COMPONENTS, VALIDATIONS, SINGLE SIGN ON, DIGITAL SIGNATURE, ACCESSIBILITY, INVOCATION OF SERVICES AND APPLICATIONS, GRAPHICS, PERSONALIZATION, MULTICHANNEL
PRESENTATION (DESK)
MANAGER SESSIONS, DASHBOARD, SEARCH, COLLABORATION, NOTIFICATIONS, ALARMS
TRANSVERSAL SERVICES
PROPERTY MANAGEMENT, INTERNATIONALIZATION, EXCEPTION MANAGEMENT, TRACE LOGGING, MONITORING, MASTER DATA, TRANSACTION, CONTEXT, CACHE
SERVICES
WORKFLOW ENGINE, BATCH, REPORT, BAM, RULES ENGINE, COMPOSITION
DATA AND BACKENDS
PERSISTENCE, MAIL SERVER, SMS SERVER, DOCUMENT MANAGEMENT
INTEGRATION
WEB SERVICES, MESSAGING, SERVICES BUS, ROUTING
DEVELOPMENT VIEW
DEVELOPMENT TOOLS
USER INTERFACE, SERVICE DEVELOPMENT, SERVICE REGISTRY, NAVIGATION FLOWS, DATA MODELING, CONFIGURATION MANAGEMENT, REQUIREMENTS MANAGEMENT, PROCESS DEVELOPMENT, CODE GENERATION
TESTING TOOLS
UNIT TESTING, REGRESSION TESTING, INTEGRATION TESTING, ACCEPTANCE TESTING, STRESS TESTING, LOCAL EMULATION
TEMPLATES
TEMPLATES, SNIPPETS, REFERENCE APPLICATION
DOCUMENTATION
APPLICATION DEVELOPMENT GUIDELINES, SERVICES DEVELOPMENT GUIDELINES, AUTOMATIC GENERATION, BEST PRACTICES, DOCUMENTATION, TESTING, DOCUMENTATION RULES, FAQS, COLLABORATIVE, USER MANUALS, DEPLOYMENT
FORMATION
TRAINING PROCESS AND MATERIALS
QUALITY ASSURANCE
QUALITY STANDARDS, QUALITY OFFICE, ANALYSIS TOOLS, CONTINUOUS INTEGRATION, LEVEL OF QUALITY SURVEYS

Figure 2.6: An excerpt of the *everis* reference model for modern information systems.

practices: project business manager, project technological manager, software architect, developer, and application builder. These five stakeholders are defined below, next to the type of project in which they participate. Each of these stakeholders has a vested interest in different architectural aspects, which are important to analyze and reason about the appropriateness and the quality of the three types of projects [59]. However, there could be more people involved in an SRA project, as Clements et al. indicate in [53]. As a consequence, although this context may coincide in other IT consulting firms besides *everis*, projects' stakeholders may vary between different firms. Below, we describe to which type of project essential stakeholders belong and their interests.

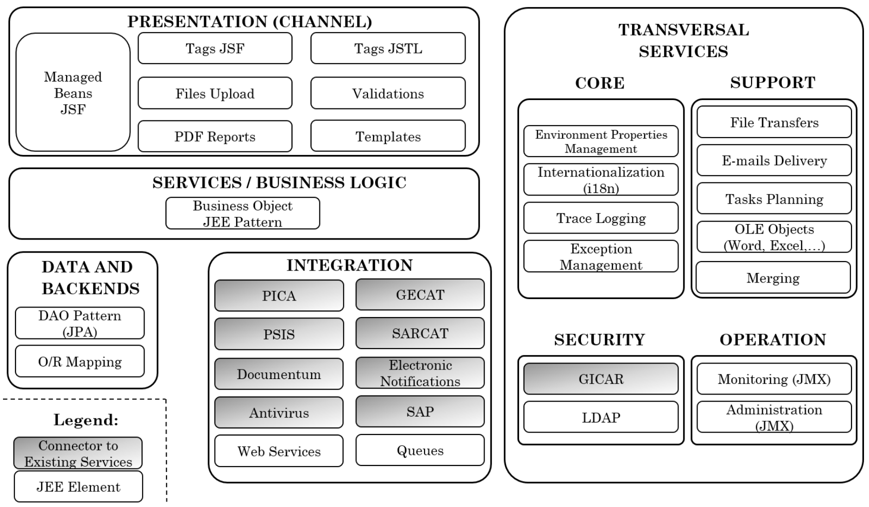

Figure 2.7: An excerpt of the execution view of an SRA designed with the help of the *everis* reference model.

2.4.1 Reference Model Projects

A reference model project is composed of *software architects* from *everis* that worked in previous successful SRA projects. They are specialized in architectural knowledge management. Their goal is to gather the best practices from previous SRA projects' experiences in order to design and/or improve the corporate reference model.

2.4.2 Software Reference Architecture Projects

SRA projects involve people from *everis* and likely from the client organization. Their members (project technological managers, software architects and architecture developers) are specialized in architectural design and have a medium knowledge of the organization business domain.

Project technological managers from *everis* are responsible for meeting schedule and interface with the project business managers from the client organization.

Software architects (also called as SRA managers) usually come from *everis*, although it may happen that the client organization has software architects in which organization's managers rely on. In the latter case, software architects from both sides cooperatively work to figure out a solution to accomplish the desired quality attributes and architecturally-significant requirements.

Architecture developers come from *everis* and are responsible for coding, maintaining, integrating, testing and documenting SRA software components.

2.4.3 Concrete Architecture Projects

Enterprise application projects can involve people from the client organization and/or subcontracted IT consulting firms (which may even be different than the reference model owner, i.e., *everis*) whose members are usually very familiar with the specific organization domain. The participation of the client organization in SRA and concrete architecture projects is one possible strategy for ensuring the continuity of their information systems without having much dependency on subcontracted IT consulting firms.

Project business managers (i.e., customer) come from client organizations. They have the power to speak authoritatively for the project, and to manage resources. Their aim is to provide their organization with useful applications that meet the market expectations on time.

Application builders take the SRA reusable components and instantiate them to build an application.

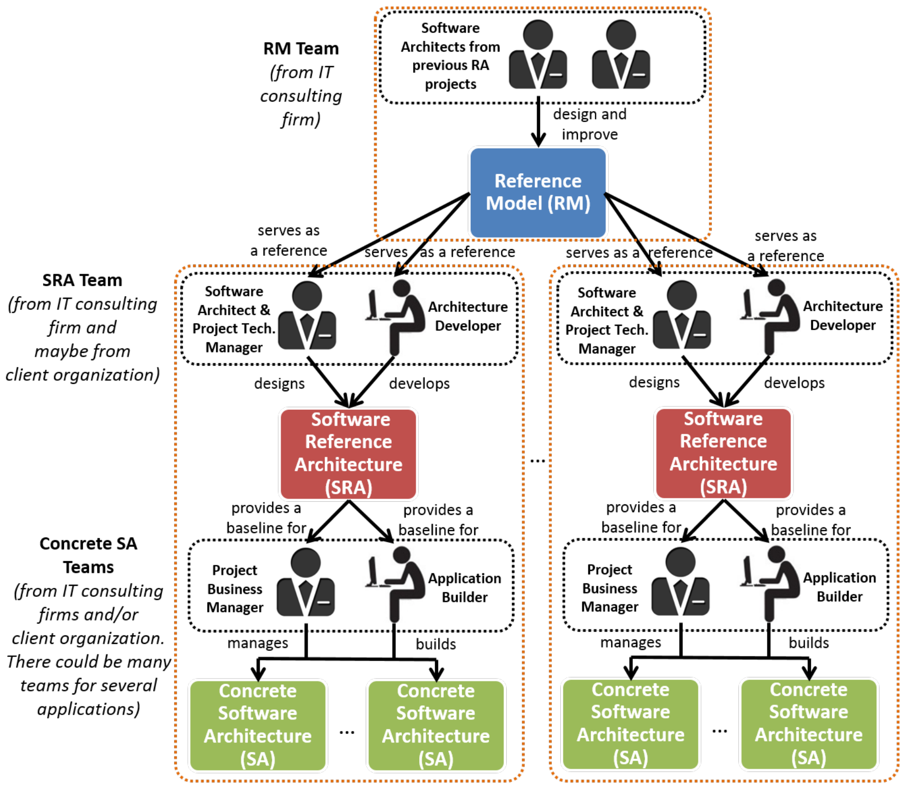
RM Team
(from IT consulting firm)
Software Architects from previous RA projects
design and improve
Reference Model (RM)
serves as a reference
serves as a reference
serves as a reference
serves as a reference
SRA Team
(from IT consulting firm and maybe from client organization)
Software Architect & Project Tech. Manager
Architecture Developer
designs
develops
Software Reference Architecture (SRA)
provides a baseline for
provides a baseline for
Concrete SA Teams
(from IT consulting firms and/or client organization. There could be many teams for several applications)
Project Business Manager
Application Builder
manages
builds
Concrete Software Architecture (SA)
...

Figure 2.8: Stakeholders and their roles in each type of project (reference model, SRA and concrete architecture projects).

Chapter 3

State-of-the-Art

This chapter summarizes the current state-of-the-art in SRAs. We conducted an SLR to present a holistic overview of the existing techniques and approaches oriented to support SRA engineering. Also, we give further details about the topics of this PhD thesis: empirical evidence on SRAs, and SRA adoption.

Currently, it is possible to find research studies focused on supporting different activities of SRA engineering (e.g., SRA analysis [4], business case [5], design, representation and evaluation [60], and conformance checking [61]) using a particular approach. These works can be considered important initiatives, since they have contributed to define architectural assets (e.g., architecturally significant requirements, domain rationale, architectural decisions documentation, and architectural guidelines) influencing software architectures in an application domain and their respective software systems.

We checked if there was any existing state-of-the-art, review (systematic or not) or survey that present a holistic overview of the existing techniques and approaches that support SRA engineering. We found four previous reviews about SRAs [4, 62, 63, 64]. In [4], Angelov et al. conducted a survey of 24 existing SRAs. However, this survey did not focus on techniques for SRA engineering. Moreover, there exist reviews about specific research themes belonging to the SRA field, namely, representation of SRAs [62], SRAs in the context of agile methodologies [63], and service-oriented SRAs [64]. These reviews have focused on specific topics. However, the objective of identifying all the studies proposing approaches or techniques for SRA engineering was not covered.

In this perspective, the objective of this chapter is twofold. First, to systematically select and review published literature, and present a holistic overview of the existing techniques and approaches to engineer SRAs. Second, to provide a detailed analysis about the current achievements and challenges in the specific SRA topics of this PhD thesis. To accomplish this goal, the following question was defined:

What are the main topics covered in the scientific literature regarding SRA engineering?

This question aims to analyze the topics that have been pursued by the research community in SRA, and to identify gaps that could be covered by this PhD thesis. Hence, we decided to conduct an SLR to answer this RQ.

This chapter is structured as follows. Section 3.1 presents the research method used: an SLR. Section 3.2 answers our RQ. Section 3.3 provides details about the topics of this PhD thesis.

3.1 Research Method

The review protocol of the SLR was developed following the guidelines proposed by Kitchenham and Charters [65]. The following subsections give a brief overview of the SLR process (search, inclusion/exclusion criteria, data extraction, and synthesis, respectively).

3.1.1 Search Process

The following electronic databases were used:

- Scopus (`http://scopus.com`),
- Web of Science (`http://isiknowledge.com`),
- IEEE Xplore (`http://ieeexplore.ieee.org/Xplore/home.jsp`),
- ACM Digital Library (`http://dl.acm.org`),
- ScienceDirect (`http://sciencedirect.com`), and
- Springer (`http://link.springer.com`).

Scopus and Web of Science are general indexing systems. These two general indexing systems are efficient, covering a huge amount of high-quality studies from all research topics [66, 67]. The use of other general indexing systems such as Google Scholar would increase the number of studies to evaluate, but not add new included ones [67]. For SE journals and conferences, IEEE Computer Science Digital library and ACM are good choices in addition to general indexing systems. Important venues in the software architecture area, such as the Working IEEE/IFIP Conference on Software Architecture (WICSA) and the federated conference series from Component-Based Software Engineering and Software Architecture (CompArch), are included in these digital libraries. We also include Science Direct and Springer because they cover a subset of journals and conferences important for our topic, such as Journal of Systems and Software (JSS), Information and Software Technology journal (IST), Empirical Software Engineering Journal (ESEJ), and European Conference on Software Architecture (ECSA).

During the selection of the search terms, the most suitable words, synonyms, acronyms or alternative spelling within the research field were identified according to the five viewpoints (population, intervention, comparison, outcomes, and context) recommended in [65]. In the context of our SLR, *Population* denotes the application area; in our case, software systems from a domain. *Interventions* refer to software technologies that address specific issues; in this SLR, SRA engineering. *Comparison* is the software engineering methodology, tool, technology, or procedure with which the intervention is being compared; however, we do not compare SRAs with other approaches. *Outcomes* are defined as factors of importance to practitioners; in the case of SRAs, reduced development costs for instance. Finally, *Context* refers to the context in which the tasks are performed; in our case, SRA engineering to architect software systems.

We tested several search strings, and we decided not to put any restriction on the population, comparison nor outcomes, since we want a broad overview of the research area. For instance, if we only consider certain outcomes (e.g., increased reuse and productivity), the overview could be biased and the review incomplete. Focusing on SRAs (intervention) for the software architecture of software systems (context), the greatest number of relevant papers was retrieved.

It is very important to include the context restriction to avoid getting SRAs focusing on other architecture disciplines rather than software architecture, such as reference architectures for hardware infrastructures, enterprises, or software product lines. We considered the following similar terms of software architecture as listed by Qureshi et al. [24]: software structure, software design, system architecture, system structure, and system design. Specifically, our final search string was:

(*"reference architecture?"*)
AND
(*"software architecture?"* OR *"software structure?"* OR *"software design?"* OR *"system architecture?"* OR *"system structure?"* OR *"system design?"*)

Two strategies were selected to perform the search: automatic and manual (see Figure 3.1). First, we queried our search string to perform the search into the aforementioned databases. The search was conducted using filters on the titles, abstracts, and keywords of the studies. We also adapted the search string notation for each digital library, since each one uses a different syntax. We did not use refinement of the resulting list considering only works related to computer science, since this refinement was only available in certain data sources, such as Web of Science. The number of studies that we got from each database was: Scopus (332 studies), Web of Science (129), IEEE Xplore (175), ACM Digital Library (102), ScienceDirect (29), Springer (59). In total, 826 studies.

Second, to ensure that we found a representative set of studies, we compared the results obtained in the automatic search with a collection of primary studies that had previously been identified as studies expected to appear in the results [4, 61, 3, 10]. This comparison was quite satisfactory, since all of these studies were obtained by the automatic search. However, since it might miss useful citations, bibliographies of every selected citation were checked iteratively during the data extraction phase for useful studies that could be missed in the initial search (i.e., snowballing). Using snowballing, two new studies were included.

3.1.2 Inclusion and Exclusion Criteria

Once we had the potentially relevant primary studies, we started the selection process. A complete list of selected and excluded studies was maintained identifying the reason of inclusion/exclusion. We managed the primary studies with JabRef[1] (during the process of screening the papers). Next, we present the inclusion and exclusion criteria.

Inclusion criteria:

- IC1: The work is mainly focused on an approach to support SRA engineering (e.g., analysis, design, representation, evaluation, and use).

[1]`http://jabref.sourceforge.net/`

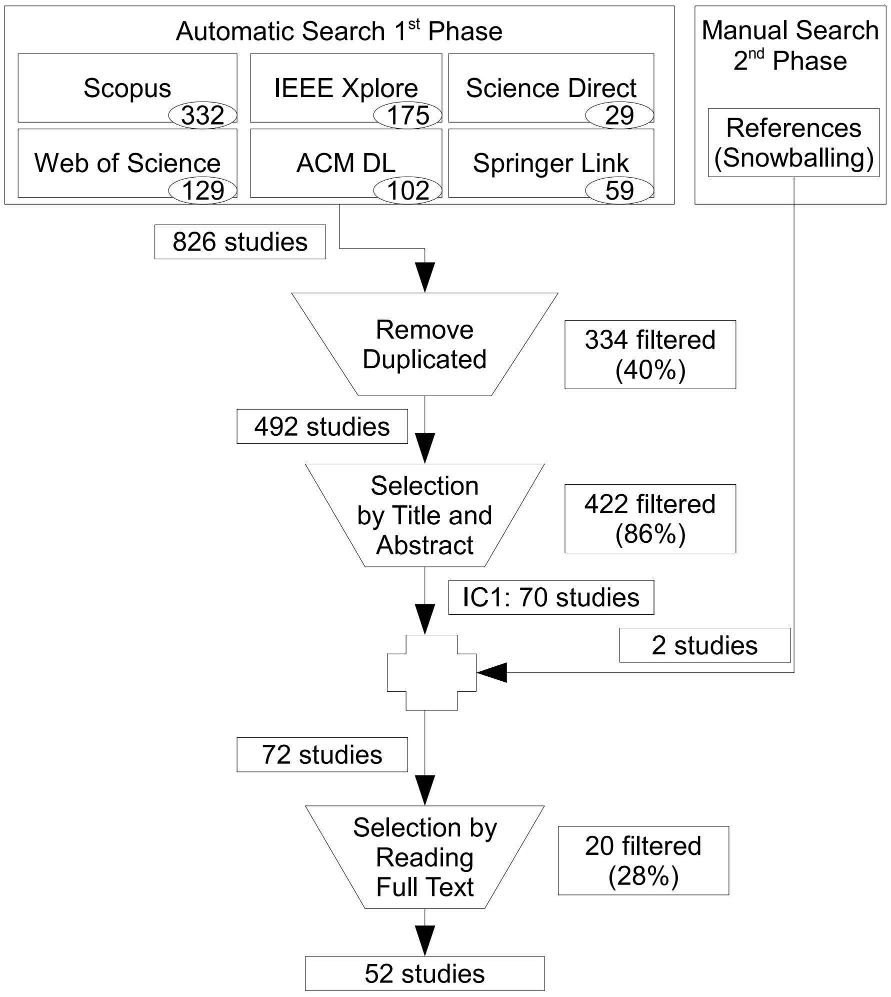

Figure 3.1: Search process (automatic and manual phases), and study selection (duplicates removal, selection by title and abstract, selection by full text).

Exclusion criteria:

- EC1: The topic of the paper is not focused on SRAs for software systems, but in other area (e.g., enterprise architecture, and software product lines). Although the work may be about an approach to support SRA engineering, it is not mainly used for software systems.

- EC2: The work only claims to present an approach to support SRA engineering, but the work is not mainly focused on that. The term SRA just appear incidentally and no real work is done in that direction.

- EC3: The work is not a research paper published in books, journals, conferences or workshops (e.g., an editorial for a special issue, a table of contents, short course description, tutorial, summary of a conference, PhD thesis, and master's thesis). Moreover, short papers of less than three pages were excluded.

- EC4: The work is not written in English.

It is important to note that in the case that the same authors reported similar work in different papers or venues, the older work was not excluded. We aggregated these works considering only last improvements and results.

Studies were included/excluded according to the following steps: removal of duplicates, selection/exclusion by title and abstract, and selection/exclusion by reading full text. The study selection process is depicted in Figure 3.1.

To ensure the quality of the SLR, the selection process was carried out by three researchers. This improves the reliability of our study [68]. Every paper was reviewed exactly by two researchers. Agreement between them was measured using the Cohen's kappa coefficient [69]. Table 3.1 shows that for the inclusion criteria IC1, both researchers agreed on including 68 studies, and excluding 388 studies. However, they initially disagreed in the remaining 36 studies (i.e., 25 studies that Reviewer A included but Reviewer B excluded, and 11 studies that Reviewer A excluded and Reviewer B included). Each of the 36 studies was discussed and the disagreements were resolved during two working sessions involving the three researchers. Finally, 2 studies were also included out of the 36 disagreements. The value of the Cohen's kappa coefficient was κ=74.67. Therefore, we can consider that there was "substancial agreement".

Table 3.1: Selection by title and abstract by two reviewers.

		Reviewer B	
		Inclusion (IC1)	*Exclusion*
Reviewer A	*Inclusion (IC1)*	68	25
	Exclusion	11	388

3.1.3 Data Extraction and Synthesis

To analyze the selected studies, we defined a classification criteria, categorized into several values. For instance, one classification criteria was the "topic" of the study, which could have any value because it was an emergent criteria. The data extraction was executed with a personalized Google Form, which we designed based on the classification criteria, and was used to facilitate the process and keep track of the data.

3.2 Results

In this section, we present a very brief version of the results of the SLR. The list of included studies in this SLR is available on the Appendix B. In the remaining of this chapter, the included studies are cited as [*Si*], where *S* indicates that it is a primary study of the SLR (see Appendix B), and *i* indicates its number.

To structure and model the topics of the 52 included primary studies, we performed the following analysis. During the data extraction, reviewers assigned a topic to each primary study. These topics were not predefined, and were emerging as the data extraction progressed. Once the data extraction finished, we gained a first impression of the topical content of the primary studies. We created an initial classification with six topics: conception, requirements, design, evaluation, usage, and evolution of SRAs. We mapped these six topics to the phases of the life cycle of SRAs. For this, we used ProSA-RA, a consolidated process that systematizes the design, representation and evaluation of SRAs [60]. ProSA-RA consists of four main steps: information source investigation, architectural analysis, architectural synthesis, and architectural evaluation. As the mapping, data analysis, and synthesis progressed, we split our original "conception" topic to three topics: basic concepts, adoption, and information source. Therefore, we ended up with eight topics in SRA engineering (see Figure 3.2), which are ordered to the SRA life cycle:

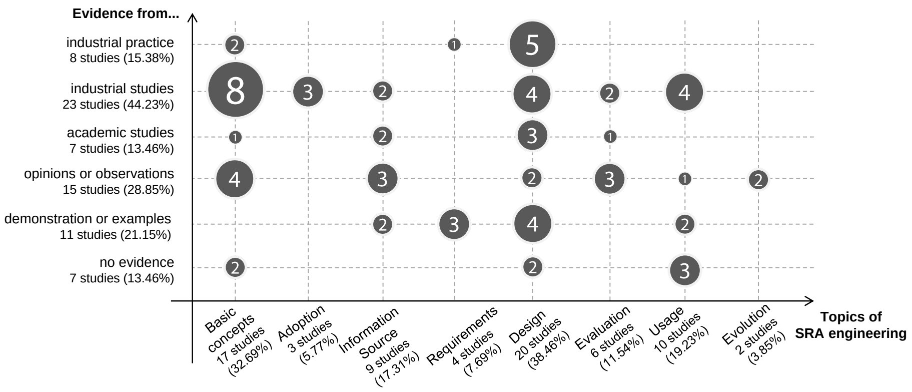

Figure 3.2: Number of studies divided by topic and the maturity of each study (from where does the evidence come).

1. understanding theoretical concepts of SRAs;
2. deciding on the adoption of SRAs;
3. looking for information to build SRAs, which matches to the information source investigation of ProSA-RA;
4. eliciting requirements of SRAs, which corresponds to the architectural analysis of ProSA-RA;
5. taking decisions about the design of SRAs, which corresponds to the architectural synthesis of ProSA-RA;
6. evaluating the design of SRAs, which corresponds to the architectural evaluation of ProSA-RA;
7. using SRAs to design concrete architectures of software systems;
8. and, evolving SRAs.

All these topics are not mandatory for SRA engineering (e.g., an SRA that is not applied in industry can skip the adoption step). Besides, they may not be sequential and be executed in parallel. Thus, this order should not been seen as a restriction, but only as a classification.

Once we determined these eight topics, we reconsidered the mappings of the included studies to such topics. In this step, we designated at least one topic to each study depending on its contributions. Note that a study could be mapped to more than one topic, for instance, [S37] gives guidelines for four topics: information source, requirements, design, and evaluation.

Next, we report the *topics* addressed by researchers regarding SRA engineering.

3.2.1 SRA Basic Concepts

Studies in this topic aim to establish the basic concepts of SRAs. The main contributions are definitions, characteristics of SRAs, and relation with other architectural assets. Although we have introduced these basic concepts in Chapter 2, next we provide a holistic overview of all the research conducted.

- **Definitions:** We have found eight definitions for SRA. The main ones are: "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" [S1]; "SRAs capture the essence of existing architectures, and the vision of future needs and evolution to provide guidance to assist in developing new system architectures" [S9]; and the definition of SRA used in Chapter 2.2.1 [S34]. Analyzing all of them, we can say the most complete definition of SRA is provided by [S34], because it considers and analyzes previous definitions.

- **Characteristics of SRAs:** Many studies have gathered from practice: characteristics of SRAs (e.g., minimal, complete, disjoint) [S9][S18], motivations and goals (e.g., to follow best practices) [S3][S9], benefits (e.g., reuse) and drawbacks or problems (e.g., learning curve) [S3][S27][S43], experiences and context in the software industry (e.g., design and use of SRAs at multiple locations, by multiple vendors and across multiple organizations) [S15][S25][S29][S31][S48][S52]. Additionally, guidelines to gather these characteristics for each organization have been proposed [S25][S29]. We conclude that [S1] presents a framework that consider most of these characteristics, which is fundamental for the understading and systematic inception of SRAs.

- **Relation with other architectural assets:** We have found several works pointing out the difference between several architectural assets and

SRAs: concrete architecture [S1][S29] (SRAs are used for designing concrete architectures), system architecture [S33] (SRAs are more abstract), architectural patterns (SRAs are not domain-independent) [S1], reference models (SRAs are mapped onto software elements) [S8][S29], standard architectures (SRAs are more flexible) [S18], product line architectures (SRAs focus on a broader domain by indicating what the design should be, not what the design is) [S1][S26][S34], architecture frameworks (SRAs target a domain for the creation of software architectures) [S33], domain specific architectures (SRAs are only a element of domain specific architectures) [S1], non-structured SRAs (SRAs address software elements and their data flows) [S1].

3.2.2 SRA Adoption

Studies in this topic help to determine if it is worth for an organization to invest on the adoption of an SRA. We have found two main contributions:

- **Checklists of value-driven data:** Studies [S25][S29] show the type of data that an organization needs to make a cost-benefit analysis to decide the adoption of an SRA (e.g., effort and reuse metrics).
- **Economic models for SRAs:** They help to make the business case of an SRA and to calculate its return-on-investment for an organization (e.g., how many instantiations are necessary before savings pay off the up-front SRA investment) [S26][S29].

3.2.3 SRA Information Source Investigation

Studies related to this topic aim at selecting and investigating information sources that could be used to build SRAs. Three main contributions were identified:

- **Sources of domain knowledge:** There is evidence on the sources of domain knowledge. Study [S37] lists several information sources: people, software systems, publications, reference models, related SRAs, and domain ontologies. Moreover, [S33] addresses five complementary ways to search in these sources: static analysis, run-time analysis, reading documentation, interviewing, and workshops.

- **Elements:** Five studies provide information on elements that could be contained in SRAs. In a high level of abstraction, [S9][S32] differ in customer context, technical architecture, and business architecture in SRAs. Moreover, study [S41] presents a reference model (called RAModel) that defines the possible elements that an SRA may have. It is also possible to find studies that aim at identifying elements of SRAs in specific application domains [S8]. Finally, we found studies that focus on a subset of these elements: the artifacts that are given as deliverables to the SRA users [S28].

- **Stakeholder identification:** [S5] identifies the stakeholders of SRAs: mainly stakeholders for the SRA design process, and stakeholders for whole life-cycle of the SRA.

3.2.4 SRA Requirements Elicitation

This topic aims at performing domain engineering to elicit the common requirements in a family of software systems:

- **Identification of commonalities in a family of systems:** There exist several approaches to identify and manage the commonalities in a family of software systems: clone detection aims at identifying the repetitive parts of several software systems (e.g., by means of CCFinder tool support) [S6]; the use of SLRs (e.g., the SyRRA process) [S40], and ontologies [S35] can systematize the elicitation of SRAs requeriments; and, finally, feature models can be used to represent the common parts of a set of software systems [S23].

3.2.5 SRA Design

Contributions of this topic aim to design an SRA and related activities, such as the representation of the SRA design. These contributions mainly focus in architectural synthesis, although they may also cover partially other activities (e.g., SRA evaluation) to support the SRA creation. We found five main contributions:

- **General processes for the design of SRAs:** We can find an empirically-grounded process to systematically design SRAs either from scratch or based on existing architecture artifacts [S14], and a mature process that systematizes the design of SRAs (called ProSA-RA) [S37].

- **Processes for the design of SRAs in specific domains:** Besides the aforementioned processes, there are others that were tailor-made and applied to design SRAs in specific domains. These processes have been used for web servers [S19], electrical/electronical systems [S24], agent-systems [S44], smart energy systems [S22], complex terminal systems [S45][S46], and the space domain [S42]. Although these processes were initially envisaged for a specific domain, they may be also useful for other domains.
- **Guidelines for SRA documentation:** Different approaches to document SRAs have been proposed. Study [S13] prescribes the specification of two documents: one to describe the SRA principles for all stakeholders; and another document to detail the SRA for architects and developers. Study [S32] discusses on different types of documentation: compact (few diagrams only) and extensive (many documents). An approach for describing SRAs that uses the principles of open source software to improve dissemination and evolution of SRAs is described in [S39]. Study [S12] discusses on SRA documentation specific for agent systems.
- **Managing Architectural Knowledge:** We can find different approaches: the use of the unified method architecture to harvest knowledge and assets from industry projects and create an SRA [S52], and guidelines to gather knowledge [S33].
- **Representation:** An adequate architectural representation improves the reuse of domain knowledge contained in an SRA. Several approaches to document such knowledge were found: through a set of architectural views [S37][S38]; considering an architectural viewpoint for variability [S17]; using Architectural Description Languages (e.g., Rapide) to capture and represent elements of an SRA [S30]; and using the 4+1 model specifically through its scenario, process, implementation, and logical views [S12][S42][S44].

3.2.6 SRA Evaluation

In this topic, evaluation refers to the task of checking the architectural description of an SRA and detecting its defects together with diverse stakeholders. We have found two main contributions:

- **SRA evaluation approaches:** We have found diverse approaches: the adaptation and extension of traditional evaluation methods of software architectures, specifically ATAM [S4] and SAAM [S15] to the context of SRAs; a checklist-based inspection approach with a list of questions that guides reviewers in detecting defects in the documentation of SRAs [S37]; and a process to evaluate rule-centric industry SRAs [S51].

- **Analysis of SRA congruence:** An SRA is congruent if it has the same type (i.e., characteristics) of previous succesful SRAs [S1]. A congruent SRA has more chances to be successful [S1]. Study [S2] presents five types of congruent SRAs that were established considering the context, goal, and architectural design characteristics.

3.2.7 SRA Usage

This topic aims at guiding the use of SRAs as well as to check how SRAs are being used. The main contributions that were found are:

- **Easing the use of SRAs and software development:** Five works supported SRA usage. Among them, we can find: a toolset to derive software architectures from feature models [S47], a knowledge base and a tool that support architects to make decisions about the design of software architectures [S31], and experiences on how certain SRAs have been used [S43][S49]. Moreover, software architectures also have been automatically instantiated using MDD approaches, that in turn, use SRAs as a PIM (Platform Independent Model) [S46].

- **Conformance checking:** Verifying if a concrete architecture is compliant to an SRA and its restrictions is important to avoid architectural erosion. We have found five studies about conformance checking: an automatic SRA conformance checking approach for different systems within a SOA landscape [S7][S48]; a rule-based approach based upon logic programming concepts towards a formalism for architectural compliance checking [S10][S21]; and an architecture-centric assessment approach for model evaluation over SRA to quantitatively estimate architecture quality [S50].

3.2.8 SRA Evolution

This reseach theme covers activities to maintain and evolve an SRA.

- **Evolution approaches:** We can find a guiding tool for the transformation/evolution of an SRA type [S1], and an evolutionary agile approach to maintain an SRA [S32].

These are the eight topics found on the literature regarding SRA engineering. In the next subsection, we focus on the specific topics of this PhD thesis.

3.3 Focusing on the Topics of this PhD Thesis

This section describes details about the three topics related to this PhD thesis, which are the three first topics of the classification from previous subsection.

First, we provide details about existing empirical evidence on *elements of SRAs* (inside the "information source investigation" topic), and *characteristics of SRAs* (inside the "basic concepts" topic). To this end, the next two former subsections deep on the literature about artifacts and benefits/drawbacks of SRAs.

Second, we provide details about *economic models for SRAs* (inside the "SRA adoption" topic). To this end, the next two latter subsections present general literature on business case analysis, and give further details on economic models for SRA adoption.

3.3.1 Literature on Artifacts of SRAs

This subsection respectively provides a brief background on the artifacts that constitute an SRA, and the design, the reuse and the usage of SRAs. In Chapter 5.3, this background is compared with results from a case study at *everis*.

SRA infrastructure: artifacts or constituent parts

The artifacts or constituent parts of SRAs have received little attention [4]. However, a few works in the SRA literature describe the artifacts that could be used to build software systems based on an SRA (see "elements" contributions in Section 3.2.3). These artifacts are also known as infrastructure [13]. Next, we show how diverse authors state significantly different views about the artifacts or deliverables of an SRA.

First, Angelov et al. distinguish *components and connectors*, *interfaces*, *protocols*, *algorithms*, and *policies and guidelines* [4]. They identified these artifacts after analyzing 24 SRAs.

Second, Galster et al. indicate that the basic structure of an SRA consists of its *common building blocks* (i.e., common stakeholders, views, model kinds) according to ISO/IEC 42010 [10]. Besides, they note the importance of the *documentation* of these SRA building blocks. The authors are based on their own experience.

Third, as we shown in Chapter 2.2.3, Nakagawa et al. [13] indicate that an SRA infrastructure provides: *software elements*, used to develop software systems; *general structure*, normally represented by architectural styles; *hardware elements*, which host software systems based on the SRA; and *guidelines*, which indicate how to apply best practices. They studied the literature of SRAs, concrete software architectures, and generic models of software systems (e.g., Zachman).

Fourth, Cloutier et al. point out that *architectural knowledge* is the key asset of SRAs. They indicate as common elements of SRAs: *business purpose*, *standards*, *guidance for implementing*, and *roadmap* [3]. The two latter ones are artifacts of an SRA infrastructure. Their vision comes from the system architecture discipline.

Fifth, Herold et al. identify the following artifacts in the SRA of a German public administration: *reusable components of software*, *operation platform*, *methodology*, *tools*, *blue-line prints* [70].

By analyzing the similarities of these views, an SRA may include the following artifacts:

- **Software elements** [13] (i.e., implementation of components and connectors [4, 70]).
- **Best practices and guidelines** [13] (i.e., policies [4], guidance for implementing [3], methodology, tools and blue-line prints [70]).
- **General structure** [13] (represented by documentation of common building blocks [10], architectural knowledge and roadmaps [3]).
- **Hardware elements** [13] (i.e., operating platform [70]).
- **Others**: interfaces, protocols, algorithms [4].

Reuse of artifacts in SRA design

One of the definitions of SRA found on our SLR was "an SRA is, in essence, a predefined architectural pattern, or set of patterns, possibly partially or completely instantiated, designed and proven for use in particular business and technical contexts, together with supporting artifacts to enable their use. Often, these artifacts are harvested from previous projects”[45]. Therefore, SRAs artifacts are often harvested from previous projects . Hence, the design of an SRA usually involves reusing the essential of existing software architectures [10]. Reusable software assets are not limited to code [71], they may include algorithms and models, design patterns, scripts, technical documentation, test results, use metrics as well as other artifacts. Reuse in the design of an SRA is a cost-effective approach to create common building blocks [72], and implies the application of proven components and architectural styles that induce specific quality attributes [71].

However, there is little evidence on what is reused in order to design SRAs. As a consequence, several problems arise when reusing assets in the design of SRAs:

- SRAs are usually not designed in a systematic manner with repeatable steps [10].
- Sharing architectural assets is not an explicit part of software architects’ job description. Thus, they need to be motivated by assisting them during architecting activities, instead of only offering repositories or templates to store their expertise and experiences [73].
- It requires a high degree of communication between people, especially when the knowledge is shared implicitly [74].

Reuse of SRA artifacts across business domains

SRAs are designed to be used in a given domain. SRAs may be designed for three types of domains: platform-specific, industry-specific, and industry-cross-cutting (see Chapter 1.1.1).

This clash of multiple disciplines, different sectors, numerous enterprises and organizations with own goals and visions, complicate the possibility of reusing SRAs across different domains. In spite of these difficulties, several efforts have been conducted to facilitate reuse across disparate domains in large European industrial research programs [20] and private partnerships [75]. Still, there is no evidence when an SRA can be reused across domains.

Usage of SRA artifacts

Checking the conformance of software systems with respect to SRA is vital to evaluate whether a software system satisfies the quality attributes enforced by the SRA or there is architectural erosion [76][70][77]. An SRA, then, is an approach for quality control during software systems development. SRAs can take up to three different roles in software development: an *instructive* role for designing new application architectures, an *informative* role for sharing architectural knowledge, and a *regulative* role for restricting the design space of systems in development [76].

Although automatic rule-based conformance checking has been explored for SRAs [76][70], there are not empirical studies that investigate how the aforementioned roles are adopted for SRAs.

3.3.2 Literature on Benefits and Drawbacks of SRAs

We reviewed the benefits and drawbacks of SRAs as asserted in the literature (see "characteristics of SRAs" contributions in Section 3.2.1). In Chapter 5.4, we will compare these theoretical benefits and drawbacks of SRAs with the results from a case study at *everis*.

On the one hand, benefits from the literature show the value of SRAs, which are a justification for their use in industry. We identified the following benefits:

- *(B1)* **Standardization** of concrete software architectures by using the SRA as a template to design a software family whose applications fulfill such standardized design [4, 59, 10, 6, 2, 22].
- *(B2)* **Facilitation** of the design of concrete software architectures by providing guidelines and inspiration to applications builders [4, 10, 22, 78, 59, 2].
- *(B3)* **Systematic reuse** of common functionalities and configurations throughout applications generation [22, 3, 59, 10, 78].
- *(B4)* **Risk reduction** through the use of proven and partly prequalified architectural elements included in the SRA [3, 59].
- *(B5)* **Enhanced quality** by facilitating the achievement of software quality aspects already addressed by the SRA [78, 2].
- *(B6)* **Interoperability** among different applications and their software components by establishing common mechanisms for information exchange [22, 3, 59, 10].

- *(B7)* **Creation of a knowledge repository** as the SRA inherently acts as a repository of applied knowledge such as architectural and design principles [3, 6].
- *(B8)* **Improvement of the communication** in the organization and multiple suppliers because stakeholders share the architectural mindset established in the SRA [22, 3].
- *(B9)* **Elaboration of the organization mission, vision and strategy**, as the design of the SRA might imply to reason about the organizational goals to be fostered by the SRA [3].
- *(B10)* **Following of best practices** as an SRA provides good practices for the organization, such as prior project artifacts, company standards, design patterns, and commercial frameworks [22, 79].
- *(B11)* **Use of the most novel design solutions**. Preliminary SRAs are usually designed to provide innovate design solutions with respect to the existing state of the art [22].

On the other hand, although the benefits of SRAs have been widely considered, their drawbacks have been scarcely documented. Below we describe the drawbacks reported in the literature:

- *(D1)* **The need for an initial investment** to create the reusable assets that compose the SRA [5].
- *(D2)* **Inefficient support for adaptation and instantiation** from the SRA to applications, as some SRAs usually lack of annotations with attributes and rules [10].
- *(D3)* **Too much abstraction**. The SRA might end up providing an inadequate level of abstraction, leaving the specific choice for specific elements fully open [22].
- *(D4)* **Lack of common interpretation** of SRA, coming from a lack of terms conventions among different types of stakeholders [22, 3].
- *(D5)* **Bad documentation** of SRA, which greatly hampers its whole understanding and use by its stakeholders. [22, 58].

- *(D6)* **Poor quality** of SRA, mainly in terms of correctness and coverage of the needs of the organization that hamper its use. SRA quality depends on whether it can be transformed into a meaningful organization-specific architecture [22, 10].

- *(D7)* **SRA too specific or limiting**. The SRA specifies the choice from the class of options for each element, what can limit innovation and novel ideas [22, 80].

3.3.3 Business Case Analysis and Return-On-Investment

This subsection and the next one respectively analyze several techniques to build a business case for SRAs and a set of specific factors and metrics (see the "SRA adoption" topic in Section 3.2.2). We consider many of the below works to create an economic model for SRAs. Details are in Chapter 10.

"A business case is a tool that helps you make business decisions by predicting how they will affect your organization. Initially, the decision will be a go/no-go for pursuing a new business opportunity or approach" [81]. Reifer has identified the following types of analysis techniques for business case [25]:

- **Breakeven Analysis**. Analysis performed to compute the value at which the solution will recover expenditures when comparing alternative use of resources.

- **Cause-and-Effect Analysis**. Analysis to explore solutions to problems.

- **Cost/Benefit Analysis**. Analysis performed to compute the net benefits (can be plus or minus) resulting from an investment decision.

- **Value Chain Analysis**. Analysis to evaluate alternatives and assess the impact of each option using a form of decision tree.

- **Investment Opportunity Analysis**. Analysis to assess the attractiveness of a range of alternatives. Example of financial measures are return on capital, after-tax rate of return.

- **Pareto Analysis**. Analysis based on the premise that most effects are generated from relatively few causes (sometimes called the 80-20 rule).

- **Payback Analysis**. Analysis to calculate the amount of time required to recover the costs of the initial investment.

- **Sensitivity Analysis**. Analysis conducted to determine to which of the input parameters the solution is sensitive.
- **Trend Analysis**. Statistical procedure used for estimating the mathematical relationship between the dependent variable and time.

The most common form of business case analysis is *cost-benefit analysis*. It involves determining the relative financial costs and benefits across a system's life-cycle as [82]. A useful economic function for business cases is the ROI. The ROI is a "measure of how much profit an investment earns computed by dividing net income by the assets used to generate it" [25]. Although the ROI is the most popular function in business cases, there are many others [83]. For instance, Net Preset Value (NPV) estimation with discounted cash flow is mostly used to address the time value of money, and the Internal Rate of Return (IRR) compares the profitability of investments. Sarmad Ali et al. summarize the economic functions used for software product lines [83]. The ROI is calculated as:

$$ROI = \frac{Benefits - Costs}{Costs}$$

Three additional factors may be important in business case analysis: the time value of money, unquantifiable benefits, and uncertainties and risk.

First, money has value that increases over time due to inflation, i.e., today's money is less worth tomorrow. To calculate such value increase, present value is used to enable decision makers to view future investments in terms of what money is worth today [25].

Second, there are benefits that may be difficult to quantify. These benefits should also been taken into account when making a business case. To cope with them, there are two possibilities: determining an estimated quantitative value for them, or listing such qualitative benefits to consider them.

Third, to adjust cost and benefits to risk, they can be multiplied by percentages that generally increase the costs and reduce the benefits (assuming the worst case). For instance, the Total Economic Impact (TEI) model by Forrester [84] proposes to multiple costs by values that range from 98% to 150% and benefits by values between 50% and 110%.

3.3.4 Economic Models for SRAs

Current research on SRA evaluation consists of analysis methods [23, 59, 50] that involve the analysis of risks, non-risks, benefits and trade-offs. Although they facilitate the analysis of those aspects based on the most important and critical scenarios, they have little support to analyze the cost and benefits of SRAs based on economics.

Introducing an SRA into an organization involves making a decision of a greater degree than only considering the aforementioned aspects, since it should not only include quality, but it should also include productivity issues. Whereas architectural quality is usually estimated in relation to eliciting implicit and explicit requirements of the different stakeholders affected by the development of the system, productivity is actually measured in terms of effort, cost, and economic benefits. Nevertheless, both views are necessary to achieve a comprehensive analysis.

Up to our knowledge, there is no specific economic model for estimating whether it is worth or not to invest in an SRA for an organization. Due to the lack of research in this specific area, we have aimed at adopting and adapting existing results in related areas:

- economic models for software product lines,
- cost-benefit analysis methods for software architectures, and
- more generic metrics about cost savings.

Economic models for software product lines and software reuse

Although we consider that SRAs and product line architectures are different (see Section 2.3.3), some perceived benefits of SRA (e.g., cost avoidance from reusing software elements) and cost-benefit factors (e.g., common software costs, unique development costs) are applicable to both, since both have reuse as their core strategy. For this reason, we studied the applicability of some economic models originally conceived for software product lines to SRAs.

Below, we summarize our results with respect to cost and benefit factors from SIMPLE [85], one of the most widespread economic models for software product lines. SIMPLE comprises a set of seven cost factors:

- C_{org}, upfront investments to establish a software product line infrastructure.
- C_{cab}, the cost to build reusable assets of the software product line.
- C_{unique}, the cost to develop unique parts of products in a software product line.

- C_{reuse}, the cost of reusing reusable assets in a product inside the software product line.
- C_{cabu}, the cost to evolve the core asset in a software product line.
- C_{prod}, the cost to build a product in a stand-alone fashion.
- C_{evo}, the cost to evolve a product in a stand-alone fashion.

These cost factors and benefit functions can be used to construct equations that can answer a number of questions such as whether the software product line approach is the best option for development and what is the ROI for this approach. Ganesan et al. extended SIMPLE by considering infrastructure degeneration over time [86].

Other important economic models for software product lines are Poulin's model [87] and COPLIMO [88]. They base their reuse-based models in two parameters: RCR and RCWR.

- RCR (Relative Cost of Reuse). Assuming that the cost to develop a reusable asset equals one unit of effort, RCR is the portion of this effort that it takes to reuse a reusable asset without modification (black-box reuse).
- RCWR (Relative Cost of Writing for Reuse). Assuming that the cost to develop a new asset for one-time use equals one unit of effort, RCWR is the portion of this effort that it takes to write a similar "reusable" asset.

For those cases in which there are difficulties to obtain historical data of building and evolving products in a stand-alone fashion (C_{prod}, C_{evo}), we consider more adequate the use of RCR and RCWR.

Finally, we must note two models (Schmid [89], InCoME [90]) that integrate cost and investment models in different layers, which make them more comprehensive. To see more models, the reader is referred to [83], in which Ali et al. surveyed twelve economic models for software product lines, and to [91] [92] in which the authors surveyed economic models for software reuse.

Value of software architecture design decisions

There exist a few economics-based software architecture analysis methods that drive the decision-making process during software architecture review and design. In this direction, CBAM [93] is a useful method for prioritizing architectural decisions that bring higher value. In addition, Ozkaya et al. proposed an economic valuation of architectural patterns [94].

These approaches help to find the optimal set of decisions that maximizes the ROI [95]. They pursue to solve the same problem of the RQ 2 of this PhD thesis, but their scope is broader and general for any kind of software architecture decision and do not reflect fundamental characteristics of adopting an SRA (e.g., cost-benefit factors for architecture-centric reuse are not considered).

Generic software metrics

There exist several approaches that propose metrics for estimating cost savings in software development and maintenance. Metrics as dependency structure matrices (DSM) have been applied to assist architecture-level analysis, such as value of modular designs, and they have proven to be particularly insightful for validating the future value of architecting modular systems [96]. MacCormack et al. extracted coupling metrics from an architecture DSM view for inferring the likelihood of change propagation and, hence, future maintenance costs [97]. Baldwin et al. presented a generic expression for evaluating the option to redesign a module also based on DSMs [98].

In addition, the concept of technical debt (either architecture-focused [99] or code-based [100]) is a way to measure unexpected rework costs due to expediting the delivery of stakeholder value in short.

Summary of economic models for SRAs

Although there is a lack of research in evaluating the economic viability of SRA adoption, there is a strong base of research in related areas. The most important related area is economic models that identify cost and benefit factors for product line architecture adoption. Although there is a significant amount of research is this direction, it falls short in:

- Validation in industry. "Very few [economic models for software product lines] actually have been used as a basis for further development or adopted in industry" [101]. Thus, "there is a clear need for many more empirical studies to validate existing models" [83].

- Easy adoption of models in industry by identifying realistic metrics to collect and report. "It is difficult for the practitioners to evaluate the usability and usefulness of a proposed solution [economic model for software product lines] for application in industry" [101]. No guidelines exist to fully operationalize the models in practice [89].

Economics-driven software architecture analysis methods do not specifically aim at making an investment analysis of the adoption of an architecture-centric program. SRA adoption is a sub area inside their generic decision-making context.

At a lower level, more simple metrics like DSM, could also be adequate for calculation the cost and benefit factors of SRA adoption and make more complete models.

Part II

Empirical Evidence of Software Reference Architectures

Chapter 4

Identifying Practical Criteria for SRA Engineering

As we mentioned in the Chapter 1, this Part II details our action research initiative with regard to the RQ 1 of the PhD thesis:

How can an organization get corporate evidence that is useful for the SRA engineering?

To answer RQ 1, we conducted five tasks (see Table 1.2, and Figure 1.3). Each of these five tasks is a chapter in this Part II.

In the current chapter, we start with the first cycle of RQ 1. In order to gather empirical evidence of SRAs, it becomes necessary to previously identify the aspects that are relevant for practitioners about SRAs. In this chapter we identify an initial set of criteria, which might be further refined after gathering evidence from SRAs. To do so, we analyzed the key criteria mentioned by *everis*' software architects during the meetings and discussions of our action research initiative, and we also studied the literature. We prioritized the *everis*' vision to make a more practitioner-oriented set of criteria.

To identify practical criteria for SRA engineering, it is useful to analyze how stakeholders perform the SRA evaluation. Although a commonly accepted set of criteria to evaluate SRAs does not exist [23, 59, 50], it has been claimed that SRAs have to be evaluated for the same aspects as concrete software architectures [23]. For this reason, we started by analyzing the identified works on concrete software architecture and SRA evaluation in Chapter 3.2.6, e.g., [53, 102, 27]. However, existing evaluation methods for concrete software architectures are not directly applicable to SRAs or other architecture-centric approaches such as product lines architectures because they do not cover their generic nature [23]. The development of a family of software systems has some characteristics that distinguish it from the development of single software systems [103]. Therefore, existing evidence for product line architectures can be also used to evaluate SRAs, namely: generic characteristics such as "variability, reusability, commonality, and compositionality" [103]; the propagation of architectural decisions while reusing common assets [103]; and lower development costs with respect to developing systems individually [83, 104, 103].

Due to these reasons, three researchers from GESSI and two software architects from *everis* elaborated further this analysis considering the specific characteristics of SRAs as described in [4, 23, 10, 50, 2], commonalities with other architecture-centric approaches such as product line architectures [105, 83, 104, 103], and experiences from *everis*. The resulting relevant aspects for understanding and evaluating SRAs are detailed below and summarized in Table 4.1.

Aspect 1 refers to the need of determining the context and classifying an SRA. As we have seen in Chapter 2.2.2, there are five types of SRAs depending on their characteristics. Among the most important characteristics are [4]:

- the organization(s) that will use the SRA (e.g., a *single organization* or *multiple organizations* that share a domain),
- who defines the SRA (e.g., *software companies* as IT consulting firms, *software groups* from the organization that use the SRA, and so on),
- the origin and motivation of the SRA (e.g., *preliminary* when the SRA solves a new problem or *classical* when it is based on previous experiences),
- the goal of the SRA and the domain of the SRA-based applications (e.g., *standardization* of concrete software architectures or *facilitation* of the design of concrete software architectures),
- and the SRA elements it may include (e.g., *components and connectors*, *policies and guidelines*, and so on).

The classification of an SRA is vital to better understand its limits, to ensure its congruency, and to facilitate its evaluation.

Aspect 2 consists of the quality attributes targeted by an SRA. The achievement of quality attributes is in fact the most compelling reason for the existence of concrete software architectures [1]. However, concrete software architectures and SRAs do not strictly determine all of an application's qualities. One example is usability: "whether the user sees red or blue backgrounds, a radio button or a dialog box" [53] is not determined by a concrete software architecture or an SRA. A list of quality attributes that lie squarely in the realm of concrete software architectures is defined by the architecture trade-off analysis method [53]: performance, reliability, availability, security, modifiability, portability, functionality, variability, subsetability and conceptual integrity. For instance, variability shows how well an SRA could be expanded or modified to produce new concrete software architectures of applications. Besides, an SRA could address more architectural qualities than a concrete architecture (e.g., reusability, commonality, compositionality, and applicability) [23, 103]. Quality attributes analysis should be wider for SRAs in this sense.

Aspect 3 comprises architectural decisions. Many prominent researchers [23, 27] highlight the importance of architectural decisions for the concrete software architecture design process and the architectural evaluation. For SRAs, architectural decisions are even more important than in a single software system since, owing to systematic reuse, an inadequate design decision could be propagated to several software systems [103].

Aspect 4 consists of the supportive technologies such as methods, techniques and tools [2, 27] that aim to improve the SRA design process and support the use of the SRA during the development of applications. Moreover, this aspect is very important for practitioners, since they are interested in knowing the latest versions of technologies and tools used in the SRA projects, and providing application builders with tools that improve their productivity.

Aspect 5 refers to business qualities of an SRA. Concrete architectures also address business qualities [23] (e.g., cost, time-to-market) that are business goals, i.e. the objectives of an organization that affect their competence [102]. These business qualities are even more important in the context of families of applications, such as SRAs: "the main arguments for introducing software product family engineering are to increase productivity, improve predictability, decrease the time to market, and increase quality (dependability)" [104].

We recommend gathering evidence about these five aspects, which are summarized in Table 4.1, in order to improve SRA engineering. Next, Chapter 5 explains how we are gathering evidence about these five practical review criteria for SRA engineering. Currently, there are no guidelines to support the gathering of these criteria. This motivated our work.

Table 4.1: Summary of relevant aspects for SRA engineering as seen by *everis* practitioners and the SRA literature.

Aspect	**Description of the SRA Aspect**
1	Overview and classification of an SRA
2	Requirements and quality attributes analysis
3	Architectural knowledge and decisions
4	Supportive technologies
5	Business qualities and architecture competence

4.1 Summary of the First Cycle of RQ 1

With the goal of supporting organizations to analyze which aspects of SRA projects are important to improve SRA engineering, we have analyzed the vision of *everis* architects and previous literature. We have identified five relevant aspects for SRA engineering:

1. Overview and classification of an SRA.
2. Requirements and quality attributes analysis.
3. Architectural knowledge and decisions.
4. Supportive technologies.
5. Business qualities and architecture competence.

Chapter 5

Gathering Evidence of SRA Engineering

As we have seen in Chapter 3, many research studies has focused on supporting different architecting activities of SRA engineering in the literature. However, their support by empirical evidence is limited, since these SRA engineering approaches are rarely based on industrial practice [22]. In order to improve the SRA discipline and to envisage realistic and effective solutions, more evidence-based research is needed to understand the current SRA engineering in practice [6]. Therefore, under this scenario, we work further to answer the RQ 1 of this PhD thesis: *How can an organization get corporate evidence that is useful for the SRA engineering?*

With this goal in mind, we have designed and conducted **an exploratory case study that analyzes nine SRA projects from *everis***. This case study was conducted in two stages. Firstly, the data related to SRA engineering in *everis* and practical review criteria for SRAs was collected and analyzed (see Chapter 4). Secondly, based on the former results, a case study involving nine of SRA projects run in *everis*' client organizations was designed to gather evidence in SRA projects. In this chapter, we report the results from the second stage.

Our research goal is to investigate the current industrial practice of SRAs from different stakeholders' perspectives. The results of this case study aims to make SRAs industrial uptake easier by helping practitioners to understand, evaluate and improve SRA engineering.

The chapter is structured as follows. Section 5.1 presents the objectives, methodology, and details of this case study.

Among the five practical criteria for SRA engineering, we focus on the most important SRA criteria for the RQs of this PhD thesis, which are:

- Aspect 1: Overview and classification of an SRA. We have focused on the motives (see Section 5.2) and artifacts (see Section 5.3) of SRAs created by *everis* for their clients organizations. This aspect helped *everis* to capture the architectural knowledge of years of work in a congruent vision; so that non experienced employees can analyze when a client needs an SRA, and understand which elements compose an SRA.
- Aspect 5: Business qualities and architecture competence. We have focused on the benefits and drawbacks from using SRAs in *everis*' clients organizations (see Section 5.4), and from designing many SRAs from a reference model at *everis* (see Section 5.5). This aspect also helped *everis* to qualitatively answer the RQ 2 of this PhD thesis (i.e., reasoning on SRA adoption).

Finally, Section 5.6 wrappers up this chapter stating its contributions.

5.1 Research Methodology

Next subsections respectively present details of the methodology of the case study: research setting, RQs, research design, data collection instruments, data analysis, and limitations.

5.1.1 Research Setting

In this case study, we focus on three of the five key stakeholders[1] related to the design and use of SRAs in *everis*:

1. **software architects** that cooperatively work 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** that are responsible for coding, maintaining, integrating, testing and documenting the software components and other artifacts of the SRA for the client organization; and

[1]To see a complete vision of the stakeholders at *everis*, the reader is referred to Section 2.4 of Chapter 2.

3. **application builders** that instantiate SRA reusable components to build concrete software architectures for the client organization's applications.

Fig. 5.1 shows these stakeholders. It shows how software architects and architecture developers are **SRA designers** whereas application builders are **SRA users**. In our context, these stakeholders are mainly professionals from *everis*, but sometimes may also come from the client organization.

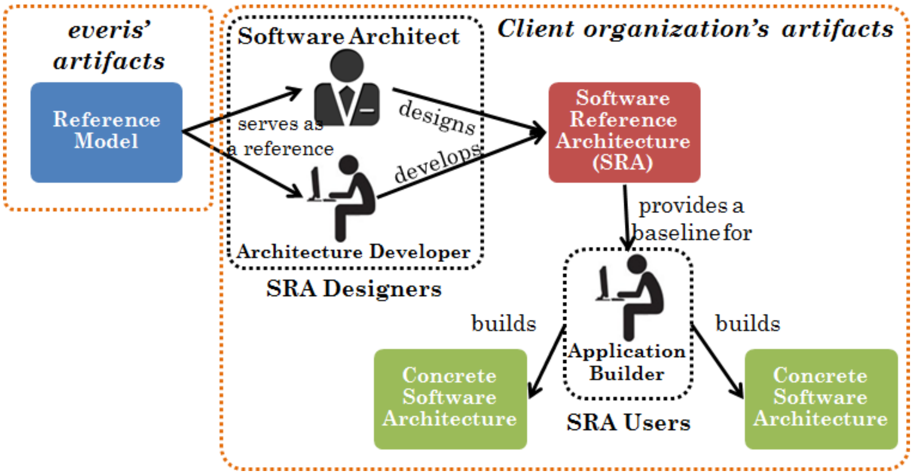

Figure 5.1: Key stakeholders in *everis*' SRA projects.

5.1.2 Research Questions

Once we got a substantial understanding of the context of SRAs in *everis*, we stated the RQs leading this case study (see Table 5.1)[2]. Below, we discuss the motives for such RQs.

[2]These RQs are from the case study of this chapter. They are not the RQs of this PhD thesis presented in Chapter 1. For this reason, we have labeled as RQ A, RQ B, RQ C, and RQ D; instead of RQ 1, RQ 2, and so on.

Table 5.1: Research questions of the case study at the second action research cycle of the RQ 1 of this PhD thesis.

Aspect	# RQ	RQ Description	Section
1	A	Which are the main motives to use an SRA to design systems' concrete architecture in an organization?	5.2
	A.1	What are the motives to use an SRA in *everis'* client organizations?	
	B	Which are the artifacts that compose an SRA and how such artifacts are designed, reused, and used?	5.3
	B.1	Which artifacts constitute an SRA designed by *everis* for a client organization?	
	B.2	What is reused by *everis* in order to create SRAs?	
	B.3	Could SRA's artifacts of a specific client organization be reused in other organizations with a different business domain?	
	B.4	What is the perception of application builders about the role that an SRA plays in the development of applications?	
5	C	What are the benefits and drawbacks of adopting SRAs in the industrial practice from the perspective of different stakeholders involved in its usage?	5.4
	C.1	What are the main benefits of SRAs in the context of *everis'* client organizations?	
	C.1.1	What are the main similarities and differences among stakeholders' perception of such benefits?	
	C.2	What are the main drawbacks of SRAs in the context of *everis'* client organizations?	
	C.2.1	What are the main similarities and differences among stakeholders' perception of such drawbacks?	
	C.2.2	What are the potential improvements that stakeholders would be willing to perform to overcome the drawbacks of the SRA?	
	D	What are the benefits and drawbacks of adopting SRAs in the industrial practice from the perspective of different stakeholders involved in its design?	5.5
	D.1	What are the main benefits for *everis* from designing many SRAs from their corporate reference model?	

RQ A: Which are the main motives to use an SRA to design systems' concrete architecture in an organization?

RQ A.1 focuses on the problems and needs of *everis'* clients organizations that led to adopt an SRA (*What are the motives to use an SRA in everis' client* *organizations?*). These results can help organizations to analyze if they need an SRA.

RQ B: Which are the artifacts that compose an SRA and how such artifacts are designed, reused, and used?

Although SRA teams are suggested to use *everis'* reference model that provide them a unified view of what an SRA should provide, we wanted to investigate which of those conceptual elements proposed by the reference model were actually used in practice or even to identify elements that are not currently part of this reference model. This would lead to pragmatic improvements on the reference model. Therefore, we stated RQ B.1 (*Which artifacts constitute an SRA designed by everis for a client organization?*).

Regarding reuse, two aspects resulted of interest. On the one hand, understanding and identifying the reuse practices of SRA teams for creating SRAs would lead *everis* to potentiate reuse initiatives inside the organization. Therefore, we stated RQ B.2 (*What is reused by everis in order to create SRAs?*). On the other hand, SRAs are naturally used to foster and improve reuse throughout the client organizations' software applications. However, there was a lack of understanding about the scope of reuse that the artifacts created for a specific client organization could have. This might help to better realize reuse strategies in *everis*; thus we stated RQ B.3 (*Could SRA's artifacts of a specific client organization be reused in other organizations with a different business domain?*).

Finally, in order to understand how the client organizations use the SRA's artifacts designed by *everis*, we stated RQ B.4 (*What is the perception of application builders about the role that an SRA plays in the development of applications?*). We especially focused on the role that an SRA plays in the development (informative, instructive or regulative). This will help us to understand the importance of the SRA's artifacts in practice.

RQ C: What are the benefits and drawbacks of adopting SRAs in the industrial practice from the perspective of different stakeholders involved in its usage?

RQ C.1 focuses on the main benefits of SRAs for *everis'* client organizations. To have an unbiased view from all stakeholders, RQ C.1.1 analyzes the perception of different stakeholders (*What are the main similarities and differences among stakeholders' perception of such benefits?*).

RQ C.2 focuses on the main drawbacks of SRAs for *everis'* client organizations. RQ C.2.1 deeps on the perception from different stakeholders (*What are the main similarities and differences among stakeholders' perception of such drawbacks?*). RQ C.2.2 seeks for potential improvements to overcome the drawbacks of SRAs (*What are the potential improvements that stakeholders would be willing to perform to overcome the drawbacks of the SRA?*).

The results from RQ C aim to provide practitioners with evidence about the benefits and drawbacks of SRAs in order to improve the current practice in SRA engineering.

RQ D: What are the benefits and drawbacks of adopting SRAs in the industrial practice from the perspective of different stakeholders involved in its design?

RQ D.1 addresses the benefits of having a corporate reference model at *everis* (*What are the main benefits for everis from designing many SRAs from their corporate reference model?*). The results from this RQ might show other software companies when it is useful to have a corporate reference model.

5.1.3 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 engineering in the *everis'* client organizations context.

We devised a flexible case study protocol since the very beginning, as suggested by [106], to register and update our procedures, instruments, decisions and deviations. This protocol was devised and agreed among the researchers from GESSI and two *everis* managers that participated in the study. These two *everis* managers had extensive experience on SRAs as they had previously participated in several *everis'* SRA projects covering all potential roles. Such experience was crucial to tackle this research.

The target population of the study was *everis'* SRA projects. We decided to approach several projects in different client organizations as it allowed us a better interpretation and assessment of the design and usage context of each SRA. It would otherwise had been very difficult to interpret certain decisions or influential factors related to the contexts of the projects. Furthermore, we targeted different stakeholders involved in the SRA projects as each of them might have different concerns about certain architectural aspects, and this might influence the perceived SRA engineering practices [59].

The two *everis* managers selected nine SRA projects from different client organizations on the basis of their suitability and feasibility to contact at least with one person playing each of the targeted stakeholder roles. Then, *everis* managers contacted potential participants for agreeing on their participation. We finally ended up with 28 people that participated in the selected projects.

Table 5.2 summarizes the projects and stakeholders that participated in the study. It can be observed that most of the studied projects are from the public sector domain while banking, insurance and industry are also represented.

In all the studied organizations (except one), we had the opportunity to approach stakeholders that covered all the related roles, namely: Software Architect (SA), Architecture Developer (AD) and Application Builder (AB); see Section 5.1 for details about these stakeholders. In two organizations (F and H), we had access to more than one application builder, and in the organization G, we could not contact any application builder as some time passed since the end of the project and any of the Application Builders that participated in the project was still at the company.

The main goals of the studied SRAs and the applications based on them are also stated. We can observe that most of their stated goals range from improving productivity by reusing components, homogenizing applications, easing the development of applications based on the SRA, ensuring the fulfillment of certain functionalities and requirements, to enabling the adoption of new technologies by the organization.

5.1.4 Data Collection and Instruments

Once the projects were selected, *everis* managers provided us the so-called SRA project card containing a summarized description, documentation, and metrics about the invested effort of each project. Whenever we needed clarification, they contacted with the corresponding project technical manager or suitable people to handle our questions. Thus, 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 about SRA engineering that the targeted roles had, we designed and piloted different data collection instruments following the guidelines stated in [106][107] and the corresponding literature background. All the instruments were designed mainly considering the literature discussed in Chapter 3.3, the practical experience of the two *everis* managers that participated in the study, and the background of our research team.

Table 5.2: Overview of the selected *everis'* SRA projects.

Id org.	Participants			Main domain	SRA project goal	Applications based on the SRA	Approx. effort (in hours)
	SA	AD	AB				
A	1	1	1	Industry	To create a minimal SRA to develop homogeneous applications.	Web-based applications to allow vendors updating information about clients in a department store.	≈5,000
B	1	1	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.	≈97,000
C	1	1	1	Banking	To provide an SRA and its guidelines to application builders so that they are more productive and can develop application easier.	Multi-platform applications of a bank with improved usability.	≈37,000
D	1	1	1	Insurance	To create an SRA that improves productivity and supports new functionalities to migrate applications to new technologies.	Applications that satisfy internal request for proposals.	≈29,000
E	1	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.	≈6,500
F	1	1	2	Public sector	To evolve the existing SRA with new technologies and functionalities.	Web-based applications for the different departments of a public administration.	≈20,000
G	1	1	0	Public sector	To evolve the existing SRA to standardize the development of applications.	Applications with enhanced reusability and reduced development costs.	≈4,500
H	1	1	2	Insurance	To create a component-based SRA with latest technologies that allows reuse in the development of applications.	Applications integrated with services of an insurance company.	≈4,000
I	1	1	1	Public sector	To create an SRA with latest technologies that support business processes.	Applications that include the business processes of a utility organization.	≈6,500

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

For software architects, we designed semi-structured interviews based on an interview guide. We choose semi-structured interviews mainly because software architects pose 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 inquiry about these details. Semi-structured interviews provided us with the ability of eliciting details for each of the analyzed projects, whilst inquiring their particularities with follow-up questions. Prior to the interview, we requested to each software architect their personal information (to shorten the meetings) and documentation of the SRA (to prepare the meetings). Interviews were conducted face-to-face in Spanish by two researchers. Each interview took about one hour and was audio-taped. An external company performed the manual transcription of the audio records into text documents.

To gather information about the SRAs vision from architecture developers and application builders, we used online questionnaires. This was based on the fact that it was almost impossible to contact them personally, as they used to work on different locations based on the client organizations, sometimes located in other cities or even countries. Furthermore, given the differences in the SRAs related responsibilities associated to them, 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 (designed by the software architects) in the client organizations. For application builders, their questionnaire focused on the use of the SRA (produced by the software architects and architecture developers) 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, and the literature studied in Chapter 3.3. To partially mitigate the rigidness of closed questions in the online questionnaires, we provided room by including also open questions to add any comment.

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 previously agreed that they could spend some time in this activity. We gave them a period of 2 weeks to complete their answers. After such invitation e-mail, we got all responses on time. Data gathered by online questionnaires was automatically prepared for its subsequent analysis using LimeSurvey's functionalities[3].

[3]`https://www.limesurvey.org/en/`

5.1.5 Data Analysis

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

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

For processing the data gathered from the interviews with software architects, we used an Excel-based template to organize each participant's answer to each question. For doing this, we used the interview transcripts and individual notes taken by the researchers during the interviews. We processed and analyzed the data as follows. First, we processed the answers to each question in order find categories that suitably described the answers. The template used to gather and process the information of the devised categories included the following columns: the name 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. Second, the categories were then further discussed and analyzed by the research team to better interpret and describe the evidence. 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 realize the contextual influence. Afterwards, we analyzed the evidence with respect to the role of the participants on the SRA projects. This led us to a better interpretation of contextual factors of our results and thus improve our understanding for devising the categories. Consequently, some initial categories were split, modified, discarded or added to ensure that all answers and their contexts were well-represented. Processing the answers of each question for envisaging its categories had its own peculiarities that will be summarized in the contexts of the description of the results.

In the case of online questionnaires, closed questions were easy to process as we took each option given in the questionnaire as a category. These categories were automatically reported by Lime Survey. 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 coming from the options given in the questionnaire). We faced three typical situations: 1) the open answer supported an existing category by providing further detail about it; 2) the open answer contradicted some of the categories; and 3) the open answer provided additional information not included in any category. In the first case, we registered the further information supporting the category but no modification of categories was done. In the second case, we carefully assessed the affected categories and had to do some modifications, such as splitting, modifying or adding another category to ensure that all answers and their contexts were well-represented. In the third case, we assessed the convenience of widening existing categories or adding a new one. In all cases, the open questions related information helped us to enrich our understanding of the situations.

To provide a global understanding of SRA engineering (as shown in sections 5.2, 5.3, 5.4, and 5.5), we assessed all categories gathered from each role's instrument, and then proceed 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 aimed to provide us a way to describe our findings instead of providing a quantitative vision of the *everis* context.

5.1.6 Limitations of the Study

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

Construct validity

It refers to issues that affect our ability to reflect the constructs under study using adequate instruments. To strengthen this aspect we performed a planning of the study and established a protocol as suggested by [106]. We paid special attention to the design of our data collection instruments (i.e., the interview guide and the questionnaires) in such a way that they were fully understood by the respondents. We made sure of polishing 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 for referring to the same thing. Thus, all instruments were revised by *everis* managers, piloted, and enhanced to ensure their effectiveness. Furthermore, we included specific mitigation actions for evaluation apprehension by ensuring the confidentiality and aggregation of the answers.

We are aware that the online questionnaires (used for architecture developers and application builders) limited our ability to further inquiry about their perceptions compared to the software architects' perceptions that were gathered through face-to-face interviews. To mitigate this threat we carefully chose the options provided by the questionnaires together with *everis* managers and the input from the software architects interviewed. In addition, we added open questions to the questionnaires to gather the participants' opinions that do not match with the options given. Hence, the respondents could freely share their real perceptions, either on the interviews or in open questions for the case of the online questionnaires.

Internal validity

It refers to factors that might affect our conclusions. For instance, when the researcher is investigating whether one factor affects an investigated factor there is a risk that the investigated factor is also affected by a third factor [106]. We are aware *everis'* managers could have chosen the most successful projects as sampling. To minimize this issue, we explained them the importance of having a representative sampling of the SRA projects in order to obtain reliable data. In addition, the fact that diverse roles from these projects were chosen as unit of analysis allows us better interpretation and assessment of contextual information.

It is important to emphasize that our results are based on the stakeholders' perceptions from the specific project that they participated in. Therefore, even if an aspect of SRA engineering was not explicitly mentioned by these individuals, there could be several factors affecting this, for instance that our instruments did not explicitly requested some potentially influential information, or cultural issues. In addition, regarding individuals that participated in the study, there is always the possibility that they forget something or do not explicitly state when they are asked about it. To reduce this risk: 1) in the case of the interviews, we discussed some potential topics that might be omitted by the respondents, and paid particular attention to ask for clarifications if necessary; 2) in the case of the online questionnaires, we designed them in such a way that the respondent must answer all the corresponding questions while s/he could complete the questionnaire at any time, so it gives them the possibility of consulting registries and documentation in case s/he needs to 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. We also made sure to design our data collection instruments in such a way that tricky questions have related questions that help to confirm the correctness of the answers. For instance, we had two questions asking about the benefits of SRAs, and differentiate between the benefits from an SRA and the corporate reference model.

Other mitigation strategies were recording and transcribing all interviews to contribute to a better understanding and assessment of the data gathered. Also, to reduce the potential researcher bias, several meetings were held among the researchers and *everis*' managers in order to discuss the course of the case study and the preliminary results.

External validity

It is concerned with to what extent 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 *everis*' client organizations context and therefore should be interpreted as such. The following results sections provide representative categories obtained through analytical generalization of the gathered evidence. We exhaustively analyzed the data together with *everis* managers, applying why questions to fully understand and explain results. To strengthen the correct understanding of this analytical generalization, in this chapter we aim to provide as much information as possible of the context and participants' sentences.

We recognize that our results cannot be generalized to other organizations without further work. However, we remark that there exist organizations with similar contexts to *everis* that could benefit from the results of this study (see Chapter 6.1). Other IT consulting firms that could be considered to some extent similar to *everis* are, for instance, Accenture [21] and Capgemini [70] 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 Figure 5.1). Besides IT consulting firms, other companies have reported a similar use of SRAs without using a corporate reference model, such as Volvo [108], Océ [50], Credit Suisse [109], and the Dutch e-government [55]. It is also important to note that all aforementioned SRAs are based on practical experience in the industry.

As Seddon et al. suggest: "if the forces within an organization that drove observed behavior are likely to exist in other organizations, it is likely that those other organizations, too, will exhibit similar behavior" [110]. Thus, we made available our instruments to foster other researchers and practitioners to use them and compare results (see Appendix C). We expect that our results strengthen the evidence regarding SRAs and encourage others to provide similar evidences that help to mature SRAs research and practice.

Next sections present the results of this case study, divided in four aspects of SRAs:

- motives to use SRAs (see Section 5.2),
- artifacts of SRAs (see Section 5.3),
- benefits and drawbacks from using SRAs (see Section 5.4), and
- benefits and drawbacks from designing many SRAs (see Section 5.5).

5.2 Analysis of Motives to Use SRAs (RQ A)

This subsection presents the results for the RQ A: *"Which are the main motives to use an SRA to design systems' concrete architecture in an organization?"* (see RQs in Section 5.1.2).

5.2.1 Results

The below results answer why SRAs were adapted for creating concrete architectures of *everis* client organizations' applications. Next, we report the motives that triggered the origin behind each SRA project. Table 5.2 shows the characteristics and objectives of each SRA project.

The motives why *everis*' client organizations adopted an SRA are shown below. We report between squared brackets the identifier of the client organization that indicated each motive (see Table 5.2).

- *(Mot-A)* 5 out of 9 projects [B, C, D, F, H] reported the update of technologies to develop applications since they were obsolete or application maintenance was costly.
- *(Mot-B)* 4 out of 9 projects [A, B, G, H] mentioned the need to homogenize the development of similar applications and identify their common elements to foster reuse.
- *(Mot-C)* 4 out of 9 projects [C, D, F, G] aimed to simplify application development (e.g., use of widely-known technologies) and improve productivity of application builders in order to hire profiles less specialized and reduce development time.

- *(Mot-D)* 2 out of 9 projects [C, I] needed to improve business processes of the organization because of organizational changes or applications misalignment with business needs.

- *(Mot-E)* 2 out of 9 projects [C, E] were started because the client organization had difficulties in developing applications without the help of a software vendor.

- *(Mot-F)* 2 out of 9 projects [A, B] mentioned the need to support and enable application development in any platform (e.g., web, smartphone, POS terminal, ATM).

- *(Mot-G)* 2 out of 9 projects [B, D] stated the need to migrate functionality from legacy systems to new systems (also known as "downsizing") to reduce maintenance costs.

- *(Mot-H)* 1 out of 9 projects [A] reported the lack of products on the market adapted to its needs and business processes.

5.3 Analysis of Artifacts of SRAs (RQ B)

This subsection presents the results and discussions for the RQ B: "*Which are the artifacts that compose an SRA and how such artifacts are designed, reused, and used?*" (see RQs in Section 5.1.2).

5.3.1 Results

The results are grouped in four subsections according to the RQ B.1, RQ B.2, RQ B.3, and RQ B.4. Results are described in terms of the categories or codes generated from the data analysis. Given the qualitative nature of our study, these categories are complemented with narrative descriptions and some representative quotes[4]. After representative quotes, it is indicated between squared brackets the identifier of the organization of the respondent (from A to I) and the role of stakeholder (nothing for software architects, AD for architecture developers, and AB for application builders). Figures are used when necessary to show the frequency of answers belonging to each category.

[4]Besides, these categories are further explained in the annex available at `http://www.essi.upc.edu/~smartinez/files/ease14annex.pdf`.

RQ B.1: Which artifacts constitute an SRA designed by *everis* for a client organization?

Software architects were specifically inquired: *"Which deliverables were produced during the SRA project and what was the aim of these deliverables?"* Based on all their answers, we found that in general, the SRA created by *everis* may provide three main types of artifacts:

- Common software elements (i.e., software components) aimed to be reused for all the applications, e.g., "the SRA is the set of software elements that support the development of applications" [E].
- Guidelines for the homogeneous development of applications. These guidelines facilitate the development of applications with the software elements. As one interviewee said, the SRA provides "a methodology, procedures and methods that have to be applied to be able to develop with the provided software elements" [C], "guidelines have to be applied to be able to develop with the provided software elements" [C].
- Documentation that describe the logical solution to create a set of applications. One participant noted: "the SRA includes the design of a logic solution to create a set of applications, and the set of software components that are developed to give support to such logic design" [C].

Regarding the produced deliverables, we respectively present below the results in terms of the three main types of artifacts (i.e., software elements, guidelines, and documentation).

Software elements Software elements are provided by means of code. This code can be provided in two complementary forms: as source code (in the 9 projects), and as ready-to-use libraries (3 projects). One software architect noted: "The SRA source code is given to the client in case that they need it in the future, and the libraries of the SRA are normally uploaded to a repository, so that application builders can get them and start to develop applications with them" [F].

Guidelines From the answers of the interviewees, we identified up to three different types of artifacts related with guidelines:

- User manuals and guidelines for development (in 7 projects). They show the procedures to be followed to develop applications. There are development guides for SRAs software elements. They show how to use them as required by the applications needs, and how to set them up. One participant summarized the issues in these words: "we make development guides for the presentation module and the rest of SRA components. Also, there are guides for the installation process of the integrated development environment and its plugins, for development methodologies, and for indicating when the SRA will give functionalities for coordination between teams (i.e., roadmap) and when an application can be released" [C].

- Tools prescription or plugins to facilitate software development (in 4 projects). Some development tools are usually prescribed (e.g., a specific integrated development environment as Eclipse, IBM RAD could be used). In addition, bespoke plugins for the IDE that automatize the development of some development tasks, and tools that support the development of applications (e.g., for continuous integration) may also be provided. Among the examples mentioned are: "a plugin that allows to visually develop workflows" [D], "a plugin that facilitates the generation of services and the invocations to services" [G].

- Templates and sample instantiations (in the 9 projects). The best way to see how something works is through examples. The SRA is always delivered with an application based on the SRA. The application could be demanded by the client organization (i.e., real), a demo or a ready-to-use template for new applications. One software architect noted these exemplar deliverables: "an application that serves as a reference and a demo application. The former is a template for any new application based on the SRA. The latter is a sample implementation developed from the aforementioned template" [G].

Documentation As the SRA grows, documentation is generated:

- Technical documentation and architectural knowledge (in 8 projects). Technical documentation includes SRA functions agreed with the client during the analysis, technical design of all the SRA components, the test plan, etc. It is useful for future architecture developers, so that they will know where everything is. However, its level of detail varies depending on the client demands. Some client organizations want detailed documentation whereas others prefer it in digestible proportions. One software architect noted the advantages of the latter approach: "This documentation does not have a lot of details; it would be impossible and non-maintainable because there are changes day by day. Therefore, we describe the main functionalities that the SRA offers and a technical description about how problems have been solved and implemented with UML diagrams of the main modules" [B]. Part of this documentation is also architectural knowledge. As a software architect pointed out: "when we have investigated how to communicate with an environment and considered it interesting, this knowledge was added to our knowledge management tools (wiki, confluence)" [D].

- Management documentation (in 2 projects). Clients also ask for management documentation such as presentations explaining the status of the SRA project. As one software architect noted: "we made .ppt presentations explaining the status of the SRA project, excel files with the tracked time, deviations from initial planning, and prediction about the end date. These deliverables are for management" [D].

RQ B.2: What is it reused by *everis* to create SRAs?

We asked software architects: *"For SRA design, was any existing component or knowledge reused, either from everis or the client organization?"* We obtained the following categories:

- Architectural knowledge from *everis* was reused in order to design SRAs (8 projects). As one participant noted: "we used *everis'* reference model to establish the gap with respect to our to-be model" [G]. It must be noted that they also stated that their experience in the current project acted as a source of new knowledge that feeds the architectural knowledge available in *everis*. As a consequence, explicit feedback was applicable to the *everis'* reference model and architectural knowledge in 3 projects, e.g., "there is another SRA that thrives in much of what we did on this SRA project. Not only in technology, but in terms of the design approach, evolutions that we have done, and so on" [A], "other SRAs are based on best practices or lessons learned from the SRA of our project" [B].

- Architectural knowledge from the market (1 project). Just one of the SRA projects was not based on the *everis'* reference model: "the SRA was based on an Oracle solution for SOA" [I].

- Architectural knowledge from own experience (9 projects). All interviewed software architects had personal experience in at least one SRA project before. Hence, they reused: "designs or solutions that we have previously applied in other SRA projects" [C]; "knowledge and technologies applied in previous SRAs" [D]; "architectural knowledge and experience from other project, concretely the use of an ESB" [E]. Another participant summarized: "at the end, components' designs are very similar (e.g., authentication and authorization). Although software elements from client organizations cannot be reused, obviously you gain architectural knowledge in previous projects and it is what you then reuse." [H].

- Architectural knowledge from colleagues (8 projects). It consists of transfer of tacit knowledge, e.g., "the transfer of knowledge and experience has been done by people, that is, the people who were in SRA projects has moved to other SRA projects and his/her knowledge and way of working has been expanded. Also, news and important things are discussed among us in meetings once a month. Finally, when anyone wants more detail of SRAs, it is discussed in front of the coffee machine" [B].

- Software elements from *everis* (1 project). Since *everis*' employees realized that architectural knowledge is reused in most of new projects, they are building a corporate platform-specific SRA that includes the most popular cross-cutting software elements common in diverse business domains. This corporate platform-specific SRA (called *j-everis*) is an implementation of the *everis*' reference model. In the newest SRA project [A], they were reusing some of these software elements.

- Software elements from the client (6 projects). When client organizations have some functionality already implemented, some software components can be reused. In 6 projects, participants noted that they reused: "existing functionalities of the financial terminal" [B], "legacy systems in Cobol through Tuxedo" [D], "a service broker of a previous version of the SRA" [E], "existing backends of the public administration" [F], "an existing database system" [H], "services implemented in Siebel (e.g., search of city halls)" [I].

- Software elements from the market (2 projects). Both open source and commercial components are sometimes reused, e.g., "an open source internationalization component" [G], "Oracle products: Portal, BPM Studio and Service Bus" [I].

- No reuse of software elements (1 project), e.g., “we reused ideas and designs from other SRA projects, of course, but we did not get any software elements and reuse it” [C].

RQ B.3: Could SRAs be reused in other organizations with different business domain?

We asked to all the stakeholders: *“Is the SRA specific to the business domain (e.g., banking, insurance, industry, utilities) of the project or generic? Could it be reused in a different domain?”*. We coded their vision of situations in which an SRA can be used as a reuse artifact in three categories (see Figure 5.2):

- Platform-specific SRA. Among these SRAs, we found two situations. On the one hand, platform-specific SRAs that can be fully applicable to other business domains. These SRAs are not tied to the business logic, i.e., SRA-based applications implement the business logic. Therefore, the SRA can be transferred to a great extent to different business domains, e.g., “I think the key of a good SRA is being completely modular, scalable, and agnostic to the application that is developed above. It enables the SRA to be adaptable to the specific needs of each project, allowing its use and application in various business domains” [D-AB]. On the other hand, mostly platform-specific SRA that also have some artifacts tied

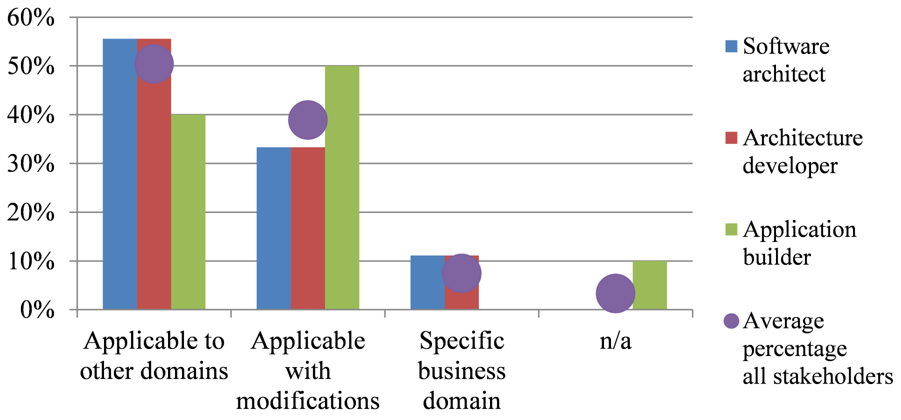

Figure 5.2: What stakeholders think about SRA reuse in other domains.

to the organization business domain that cannot be reused. Hence, the SRA could be partially reused and some modifications are needed. Some representative quotes: "An SRA must have a common part between domain/sectors and another part that should be adapted to each sector, as they have different requirements" [F-AD]. "There would be some modules that do not apply to other business domains but most parts of the SRA can be reused" [F-AB1].

- SRA is designed for a specific business domain, in which only concepts and design of a few generic functionalities can be reused, e.g., "there are software elements that could be generic, but the SRA is mostly for the banking domain" [B].

- N/A. In this category, we put the stakeholders that did not reply to the question. As one of them said: "With my experience and knowledge during the use of the SRA, I cannot give an answer" [A-AB].

Additionally, since most of the participants replied that the SRA of their projects were platform-specific and diverse software elements could be reused in several business domains, we asked to all stakeholders: *"Does the SRA offer reusable modules for cross-cutting services?"* Several options were given in the online questionnaires including an open-answer option. The answers are shown in Figure 5.3. The most popular ones are (in more than 50% of projects): persistence, security, logging, error management and configuration. These elements provide cross-cutting functionalities with a technological scope that are generic and applicable in many business domains.

RQ B.4: What is the perception of application builders of the role that an SRA plays?

To see how application builders follow the guidelines from the SRA, we asked them: *"To what extent did you follow the development guidelines provided by the SRA?"* The feedback was categorized into three possible responses:

- 2 out of 10 application builders indicated that the SRA played a regulative role as its use was mandatory, leaving them a limited degree of freedom and its use was subsequently validated. The compliance with the restrictions set by the SRA was verified "at all times" [C-AB].

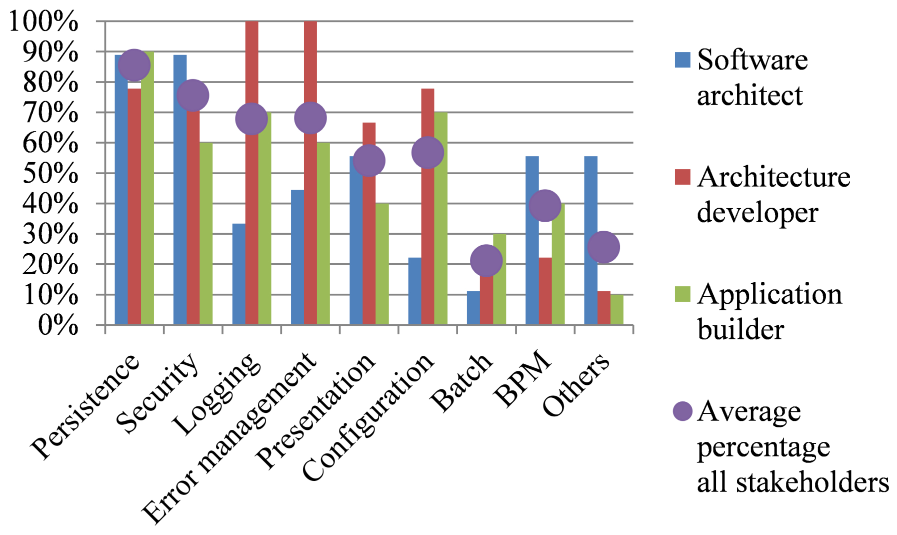

Figure 5.3: Popularity of SRAs' cross-cutting software elements.

- 7 out of 10 application builders mentioned that the SRA played an instructive role, providing them a medium degree of freedom. The guidelines established by the SRA were followed without verifying compliance (e.g., although SRA libraries are used, its usage is not controlled or verified). As one participant noted: "there was not a constant verification but we always tried to use the artifacts provided by the SRA" [H-AB2].
- 1 out of 10 application builders stated that the SRA played an informative role as its use was optional, leaving them a high degree of freedom. There was neither control about the compliance of SRA nor its use.

5.3.2 Discussions

This subsection discusses the most relevant observations from the previous results. Each subsection corresponds to one sub RQ B (i.e., RQ B.1, RQ B.2, RQ B.3, and RQ B.4). Some of the results that may be related to the context of the organizations (e.g., type of SRA, hours invested) are explained when necessary.

Discussions on RQ B.1: software elements, guidelines and documentation are the main artifacts of SRAs

In Chapter 3.3.1 we discussed the types of artifacts of SRAs as reported in the literature. The results from this study support the three first types of artifacts (see Section 5.3.1). First, an SRA provides software elements (provided as source code or libraries) to be reused for all the applications. Second, an SRA provides guidelines (user manuals and guidelines for development, development tools, templates of applications and sample instantiations) to homogenize and facilitate the development of applications. Third, an SRA includes documentation with the design of a solution (explicitly stated in technical documentation with architectural knowledge and management documentation) to create a set of applications.

Next, we analyze the artifacts that were not highlighted by the participants of this study or that surprisingly differs from our results.

- **Hardware infrastructure is concern of other stakeholders in the consulting firms context.** The hardware infrastructure and its proper working is also necessary in SRA projects. However, it is not direct responsibility of the participants of the study. Indeed, it is managed by other stakeholders, even from other providers which were not *everis*. In any event, SRA stakeholders stated that the SRA needs to be compliant with the hardware infrastructure and this issue is dealt in different ways: a) firstly design of software elements, e.g., "we started working in the conceptual approach of the SRA, and later went down and thought about the hardware infrastructure that supports it" [A]; b) firstly design of hardware infrastructure, e.g., "when we analyzed their environment, we found a pre-defined hardware infrastructure, which was a Unix with WebSphere and Java" [D]; c) in parallel, e.g., "we designed SRA modules and they fulfilled very strong non-functional requirements. At the same time, we needed to be sure that some hardware infrastructure would support it" [B].

- **SRAs mostly perceived at a technological level.** The elaboration of mission, vision and strategy of the organization was not mentioned by the participants [3]. This element is the backbone of enterprise architectures [111]. Contrary, SRAs are focused on the IT solution, instead of business processes or organizational changes. However, when the organization needs them, an SRA should provide the IT solution for these tasks, e.g., 5 software architects stated that the SRA provides software elements to support business process management (BPM), e.g., see BPM cross-cutting module in Figure 5.3). In addition, SRAs do not aim to make organizational changes. Software architects pointed out that the adoption of the SRA did not imply any organizational change, and there were changes only in the way to develop applications. However, an SRA can support organizational changes as it did in one client organization: "The SRA allowed going from a centralized software system in Barcelona to split it into 6 regions. This allowed both organizational and technological changes" [I]. It has led to the creation of two different "cultures", clearly explained by one software architect: "An enterprise architecture defines the different areas of the organization at a much higher level than SRAs, and also how it can be translated into systems, without dealing in depth in the implementation of software components at the low or technology level. It has led to two definitions of architecture: enterprise architecture, and solution SRA that is like the enterprise architecture already landed on a specific technical implementation" [H]. To sum up, SRAs result in an extension or sub part of enterprise architectures at a lower level.

- **An SRA could also include other artifacts.** Some artifacts reported in the literature, such as algorithms [4], were not mentioned in this study. A reason could be that the study does not cover all the possible business domains, and the presence of specific artifacts depends on the domain. For example, it has been reported by other researches the presence of computational models and algorithms in SRAs for the space domain [71][112]. We can conclude that an SRA does not have to include all the artifacts uncovered by this study (indeed, there are SRAs from our study that do not include all of them). Also, SRAs can provide other artifacts that were not uncovered. The unique artifacts that are mandatory are software elements because without them, no SRA can exist (it would be a reference model instead [1]). However, the more artifacts an SRA has, the more control it has over the applications, and the more benefits from reuse it triggers.

To sum up, the views of Nakagawa et al. [13] and Herold et al. [70] are very close to the results of our case study. The former only differs in the importance given to the hardware. The latter do not mention the technical and management documentation, but it has a very similar view with regard to the artifacts that serve as guidelines.

Discussions on RQ B.2: SRAs are created from existing assets

When an SRA project starts, software architects could bump into two possible situations: "when we have the chance to create a completely new SRA", but it also could happen that some architecture exists and "the client organization asks you: 'do whatever you want, but improve it' " [A]. As we have seen in the results, even in the former case, the nine SRAs of the study are defined based on accumulated practical experience from previous software systems developments (either from *everis* or the client organization). The most important artifact being reused is *everis*' reference model, in 8 out of 9 projects. This reflects its usefulness and its reuse for different client organizations. Under this scenario, *everis* has recently created a platform-specific SRA, called *j-everis*, to benefit as much as possible from reuse in future SRA projects.

This coincides with the literature that states that the SRA design is based on reusing existing assets when possible [10, 71, 72]. In industry, it seems difficult to find futuristic/preliminary SRAs (those only based on theoretical architectural patterns instead of experience), being common practice/classical SRAs [4]. Among the assets reused it is surprising that no architect mentioned reusing architectural knowledge from clients. It may be because they are consulting projects that need external support. Also, in 66% of the projects some software element from the client has been reused, which indicate the popularity of incremental evolution in SRA projects.

Finally, when assets from the market have been reused, the decision of going open source or commercial depended on the non-technical requirement of availability of budget. For instance, in projects A, E and G, open source were used to reduce costs and there was no possibility to acquire commercial packages whereas in project H the organization previously acquired Oracle products and wanted to take as much benefit from them as possible.

Discussions on RQ B.3: platform-specific SRAs are potentially reusable in many business domains

SRAs can be designed to capture the essence of software systems that belong to either a technological or a business domain. The respondents of our survey support these two types of domains. It is important to note, though, that the SRAs of our survey were designed for a single organization (an *everis*' client organization).

As a main result, we saw that platform-specific SRAs (those with the scope of a technological domain) can be interesting for many organizations with different business domains but similar technological problems. Therefore, they are potentially reusable: "you cannot reinvent the wheel; if you create an SRA is because you think it is good. If you then go to another SRA project and do not use any of the previous, it would be a little suspicious" [C]. However, since they are created for a single organization confidentiality and property issues come up: "from previous SRAs we reuse gained knowledge, but the code of the SRA is property of the client organization" [H]. On the other hand, the reuse of SRAs with the scope of a business domain would require the cooperation between competitor organizations, what seem difficult if there are not special interests by all parts. An example of a industry-specific SRA for many organizations is AUTOSAR, which standardizes software development of automotive competing firms (see Chapter 7.1).

Although potentially reusable, platform-specific SRAs are difficult to design in the beginning: "designing reuse SRAs complicates the design phase because you have to identify the pieces that are really reusable whereas the business logic is responsibility of the application developer" [F]. Platform-specific SRAs are possible because "there are not that many architectural styles, e.g., Microsoft made a compendium of architectures [113]" [A].

We think that a confounding factor to this result is the effort invested in an SRA. A high effort invested in an SRA project could lead to create many specific artifacts to the business domain. The SRA of the project B, which has been evolved since 2006, was the unique highlighted as not reusable in other domains. The five SRAs categorized as platform-specific [A, D, E, G, I] and that can be applicable to other business domains are among the ones in which less effort has been invested. Another factor is that three of these SRA projects were only in the design phase.

- **Divergences among stakeholders.** With regard to SRA reuse in other domains, software architects and architecture developers of the same projects share a similar vision whereas application builders differ from them. It may be because architecture developers have a global vision similar to software architects. However, application builders lack experience. Indeed, this role could be performed by people that have just started to work in the company without experience (as the two application builders that do not share the same vision).

  The discrepancy about the most popular cross-cutting elements shows the importance of such elements for the different types of stakeholders. Architecture developers and application builders give importance to the modules that help then through the development (e.g., logging, error management, and configuration) whereas software architects highlight the modules that help to fulfill significant requirements (e.g., BPM).

Although most of the organizations share popular cross-cutting elements (e.g., persistence and security), the inclusion of others elements in a SRA depend on the business domain and on the organization needs. For instance, the organizations of the public sector domain present different needs: project E needed batch tasks; project I did not need presentation; and the projects E, F and G do not include business processes (i.e., BPM).

Discussions on RQ B.4: conformance analysis is unusual

Software architects give high importance to an adequate adoption of guidelines, e.g., "no matter how good the SRA is, if application builders do not follow guidelines and procedures properly for the good SRA usage, they will not get much profit" [C]. However, in most of the cases, although application builders follow SRA guidelines, the compliance of the resulting software systems with the SRA is not verified. A reason for this situation is whether the goal of the SRA may involve the need of conformance analysis or not. The nine SRAs of this study aim to either standardize or facilitate the design and implementation of a set of software systems (the two goals that an SRA could have as stated in [4]). The former type of SRA requires to all applications of the organization to be based on the SRA (plays a regulative role) whereas the latter type just recommends and facilitates the development of applications based on the SRA (instructive/regulative role). As one software architect noted: "there are usually two types of approaches to SRA: the extremist, in which the SRA indicates how to do everything and application builders focus on what the SRA gives and cannot do anything else; and other less strict, in which the SRA provides some tools, some modules and components, and then application builders use and extend them as they wish" [B]. When conformance analysis is done (e.g., through rules for analyzing code and dependencies in Sonar) applications are not uploaded to the production environment if they do not conform the SRA rules. For those SRAs that aim at facilitating the design, conformance analysis is not a must. For instance, the SRA of the project F is for a public administration with many IT departments, therefore they cannot force developers to forget previous technologies, but only suggest them to use the new SRA. Other reasons could be that companies demands short time-to-market and conformance checking requires time and resources, or the unavailability of tools.

5.4 Analysis of Benefits and Drawbacks of SRAs (RQ C)

This subsection presents the results and discussions for the RQ C: *"What are the benefits and drawbacks of adopting SRAs in the industrial practice from the perspective of different stakeholders involved in its usage?"* (see RQs in Section 5.1.2).

5.4.1 Results

The results are grouped in two subsections according to the RQ C.1 and RQ C.2. The following elements are used to report the main findings:

1. non-mutually exclusive categories created from the analysis of all stakeholders' responses as indicated in Section 5.1.5;
2. representative quotes of these categories, indicating among square brackets their project.
3. bar and bubble charts that respectively show the frequency in which stakeholders mentioned each category. They indicate the most popular SRAs benefits and drawbacks, and the different perception among SRA designers and users.

Moreover, we provide further details about categories, stakeholders' representative quotes, and cluster analysis to understand different contextual aspects (such as respondent experience and different application domains) in an additional document[5]. Furthermore, the raw data is available on a csv file[6].

RQ C.1: SRA benefits from using SRAs in everis' client organizations

We report the resulting categories of SRA benefits, enclosing the perception of all stakeholders. Table 5.3 shows representative quotes of these categories and Fig. 5.4 presents the detail of how many respondents of each type considered each of the benefits. The categories of the mentioned benefits were:

- (*Ben-A*) **Reduced development costs**. 23 out of 28 participants emphasized the perception that the SRA reduces the development effort and the costs by enabling the reuse of common assets, facilitating functionality and speeding up the development of applications. Regarding to this,

[5]Available at `http://www.essi.upc.edu/~smartinez/files/tosem15-attachment.pdf`
[6]Available at `http://www.essi.upc.edu/~smartinez/files/tosem15-data.xlsx`

some respondents also commented on the appropriateness of investing time in 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 SRAs.

- (*Ben-B*) **Improved maintainability and reduced maintenance costs**. 22 out of 28 respondents perceived an improved maintainability and understandability of applications derived from the SRA mainly because of the modularity of the SRAs.

- (*Ben-C*) **Easier application development and increased productivity of application builders**. 21 out of 28 participants stated that the SRA artifacts help them to build applications in an easier way because artifacts abstract them from most technical problems (e.g., communication, backends, . . .). This is because architecturally-significant requirements were already addressed by the SRA artifacts, facilitating the development of applications.

- (*Ben-D*) **Incorporation of latest technologies**. 15 out of 28 agreed that SRAs were used as a way to foster the use of the latest technologies in the applications of the organization. Among other things, using latest technologies instead of older ones facilitates the recruitment of professionals with the required technological skills.

- (*Ben-E*) **Alignment with business needs**. 12 out of 28 mentioned that the design of the SRA inherently considers important organizational business processes, so that the applications that are based on it, are more aligned with business needs, e.g., by supporting the particular workflow in a process.

- (*Ben-F*) **Homogenization of the development and maintenance of a family of applications**. 9 out of 28 respondents considered that the standardization promoted by SRAs implies a higher control over what it is being done (supporting distributed teams in different locations), and helps creating a corporate style for all applications.

- (*Ben-G*) **Increased reliability of SRA's software elements that are common for a set of applications**. This idea, shared by 9 out of 28 participants, states that the SRA elements have been tested and matured. As a consequence, SRA common elements have fewer errors.

- (*Ben-H*) **Others benefits**. In this category we include the 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 of incorporating more functionality to applications. Application builders indicated: improved agility when requirements are changed; improved decision-making; good documentation of SRAs.

As we can see in Table 5.3 and Fig. 5.4, about 80% of participants agreed that client organizations mainly benefit by: reduced development costs (Ben-A) and improved maintainability (Ben-B). By means of the interviews done to software architects we got useful details to understand how reduced develop-

Table 5.3: Quotes from respondents about SRA use benefits.

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].	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, what implies reliability" [F].	5	3	1	33%
Ben-H	"We used good practices like removing 'dead code' and wrappers of the SRA" [F]. "The SRA can be found on the Web, what saves distribution costs" [H]. "If the application builder has problems, there is a support team that helps him/her to solve problems" [F].	2	3	2	25%

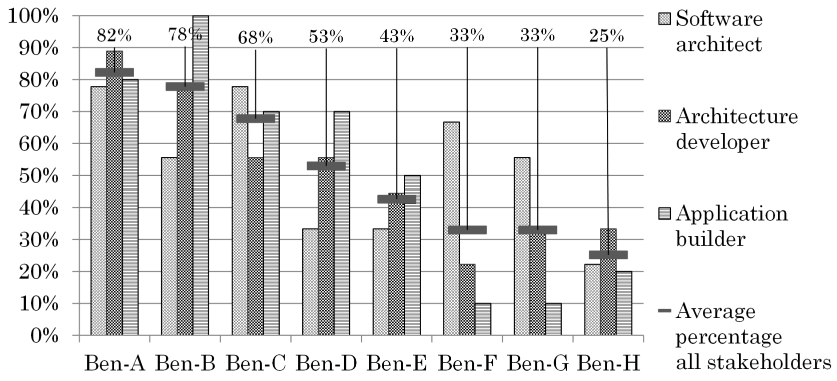

Figure 5.4: Benefits of the use of SRAs in organizations.

ment costs was achieved: because of the reuse of software elements, such as cross-cutting modules and services that implement business logic (in 5 out of 9 projects); agile and automatized development (4); improved configuration of software elements, e.g., setting up the modules to be reused by the application is fast (3); and technological and architectural decisions were already taken, i.e., it saves time during the architecture design of every new application and improves reliability (2).

In a lower extent but still highly supported (68% and 53% respectively), participants also mentioned easier development (Ben-C), and incorporation of latest technologies (Ben-D) as relevant benefits.

Next, we report the different stakeholders' perception about such benefits.

RQ C.1.1: Stakeholders' perception of benefits from 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. 5.5. The X-axis contains the frequency in which SRA designers mentioned the benefits whereas the Y-axis represents the same frequency for application builders. We divided Fig. 5.5 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

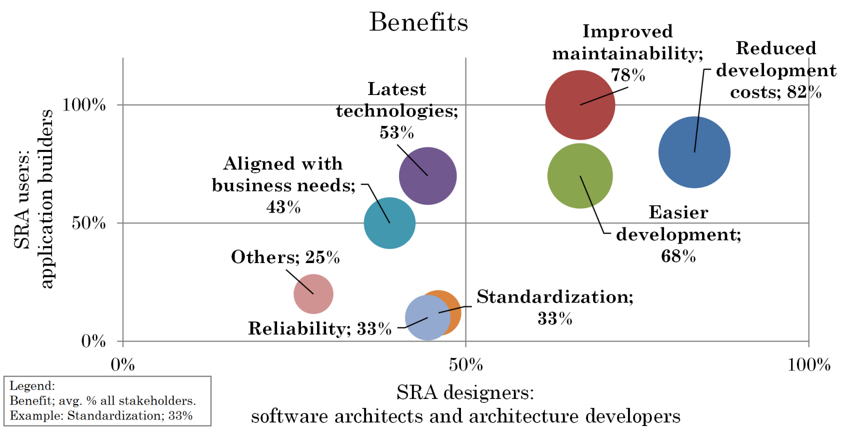

Figure 5.5: Comparison of SRA benefits between designers and users.

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 on benefiting from reduced development costs (Ben-A) and facilitating the development and maintenance of applications (Ben-C). These two benefits were mentioned by a similar percentage of SRA designers and users.

However, there was lesser stakeholders' agreement for 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 a better understandability of SRA-based applications. Application builders were also more concerned by the use of the latest technologies (Ben-D), since they daily use the chosen technology stack. Moreover, application builders considered the fact that applications are more 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 has 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 a higher knowledge of its business processes.

On the other hand, SRA designers disagreed about the standardization and reliability of SRAs as a benefit (see Table 5.3). 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 about the whole project and have more experience in other SRA projects.

RQ C.2: SRA drawbacks and risks from using SRAs in everis' client organizations

The three types of stakeholders mentioned the following drawbacks (see Table 5.4 and Fig. 5.6 for the details):

- *(Dra-A)* **Additional high or medium learning curve for using the SRA**. Application builders need to learn how to develop and maintain applications with the 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.
- *(Dra-C)* **Applications' dependency over 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 could be complex, especially when it grows.
- *(Dra-E)* **None**. Some of the responders indicated that the adoption of an SRA 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 that were not mature enough to be productive).
- *(Dra-G)* **Other drawbacks**. In this category we include the drawbacks that were not mentioned more than once by any type of stakeholder. Software architects indicated: difficulty to measure the 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.

The most mentioned drawback of using an SRA, mentioned by 63% participants, was that application builders need time to attend to 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

Table 5.4: Quotes from respondents about SRA use drawbacks.

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].	5	4	9	63%
Dra-B	“The SRA homogenizes and standardizes. Therefore, it implies less room for innovation. For instance, an application very ’rare’ will never be developed” [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, applications development is blocked” [C].	4	2	0	22%
Dra-D	There is no representative quote because this category come up from the online questionnaires.	0	2	2	14%
Dra-E	There is no representative quote because this category come up from the online questionnaires.	0	1	3	14%
Dra-F	“The used ESB 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 applications builders and business logic that they put inside" [F].	2	2	2	21%

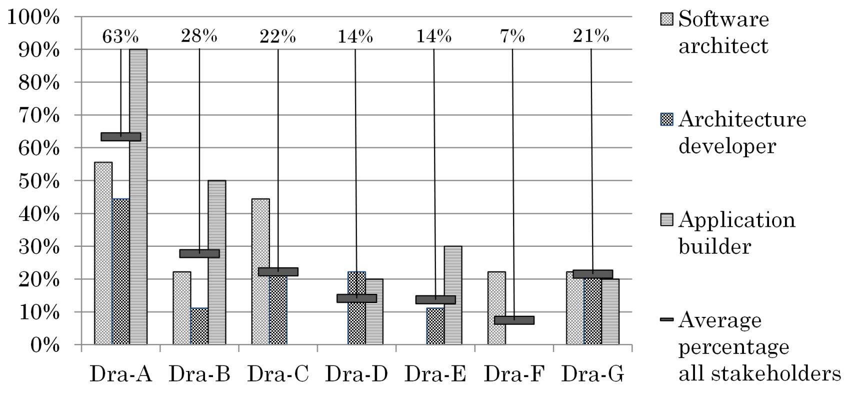

Figure 5.6: Drawbacks of the use of SRAs in organizations.

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).

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

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

RQ C.2.1: Stakeholders' perception of drawbacks of SRA use for client organizations. The findings about drawbacks clearly reflect the daily work of each role (see Fig. 5.7). Software architects are more worried about decisions of technologies that they make (Dra-F) and offering as soon as possible SRA common software elements to application builders, so that they do not block others (Dra-C). On the other hand, application builders are more worried about the learning curve (Dra-A), and the restriction of following SRA standards and procedures that forces them how to do things (Dra-B).

Surprisingly, only 14% of 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 to be a drawback. The main reason could be that they think that SRAs ease the development of applications,

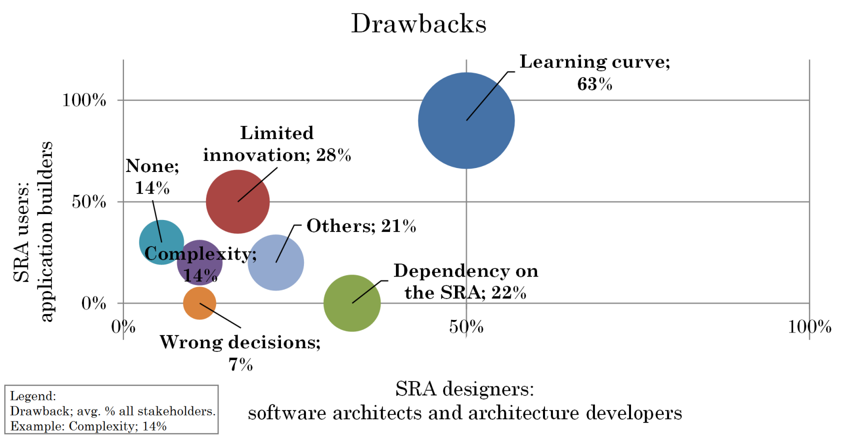

Figure 5.7: Comparison of SRA drawbacks between designers and users.

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

In addition, stakeholders were also inquired about potential improvements they would do to the SRA.

RQ C.2.2: Improvements that stakeholders would do. We asked to stakeholders what they think that should be changed, included or updated in future versions of the SRA. Stakeholders mentioned the following improvements to be done (see Table 5.5 and Fig. 5.8 for the details):

- *(Imp-A)* **Add functionality or modules to the SRA**. For instance, one interviewee suggested developing a visual plugin to make easier the development of components and automatize more the job for application builders.

Table 5.5: 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 easier the development and automatize more the development of components for the application builders" [C].	6	4	8	64%
Imp-B	"We are limited with JSF, migrating to other framework, like Sencha, would allow offering presentation not only for web browser, but also mobile devices" [G].	5	2	1	29%
Imp-C	"There is a module that is more complete than what it is really asked for. Thus, it is not aligned with business 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%

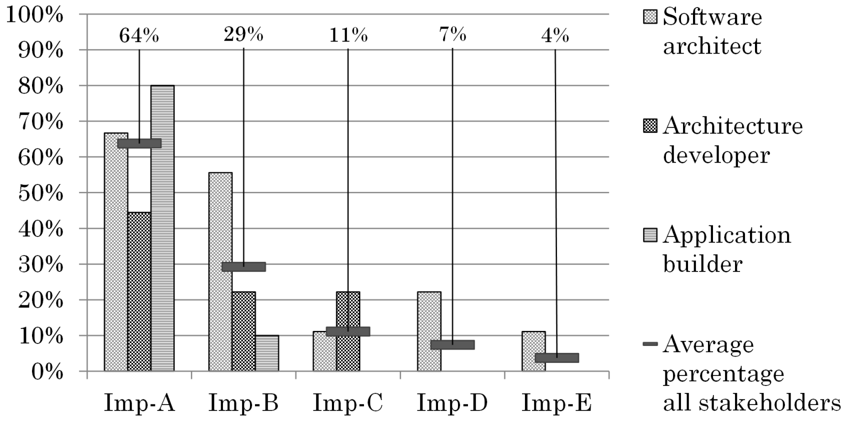

Figure 5.8: Identified to-do improvements in *everis*' SRAs.

- *(Imp-B)* **Technology change** because current one is not enough mature or appropriate, or it 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 many functionalities.
- *(Imp-D)* **Add new practices or guidelines to the SRA**, e.g., moving to a continuous integration approach.
- *(Imp-E)* **Migrate from legacy applications**.

The most mentioned improvement, by 64% of 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 their design as long as they are used.

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

5.4.2 Discussions and Comparison with the Literature

The aim of Subsection 5.4.1 was to show the results and to discuss the vision of different types of stakeholders (which was not addressed in previous literature). In this subsection, we compare the main findings of our study with respect to the literature (see Chapter 3.3.2), to analyze how our findings support the claims made by other researchers. Next, we respectively focus on: the benefits of SRA usage, the drawbacks of SRA usage, the analysis of benefits and drawbacks in certain application domains, and how to use these results.

Discussion of SRA benefits

Table 5.6 compares the theoretical benefits of SRAs (see Section 3.3.2) with the results of this case study. The first column indicates the benefit. In total, there are eleven benefits that were 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 refuse (×) these benefits. If the benefit was mentioned while asking about the benefits of designing SRAs (instead of using them), we indicate it with an asterisk (*), see Section 5.5.2. The third column represents the percentage of stakeholders that mentioned that benefit. Finally, some comments are made.

The results of the study show that *everis'* practitioners give different importance to specific benefits. For instance, they give more importance to reuse and facilitation than to other benefits. The most perceived benefit of the SRAs in the client organizations was systematic reuse (B3), that lead to the reduction of the time and cost to develop and maintain applications and shorter time-to-market. The second most perceived benefit was the facilitation by providing artifacts for the design and development of applications (B2), which have been specifically discussed in [38]. In our study, interoperability (B6) and best practices (B10) were not remarkably highlighted as a benefit of SRAs. This could be because the stakeholders did not have it explicitly in mind as they were not the main goals of their projects.

Discussion of SRA drawbacks

Table 5.7 shows the drawbacks from using SRAs following the same format of Table 5.6. In this study, three new drawbacks of SRAs were found, which are shown in the last three rows because they could not match with the theoretical ones. These three drawbacks are: learning curve, dependency in the SRA, and complexity. It is important that practitioners are aware of them so that they focus on lowering the risk from them when designing or restructuring their SRAs.

First, learning curve is the most common problem in SRA projects. Such learning curve mainly consists of the training of application builders in new technologies and specific design decisions, e.g., "the SRA is very specific. Although SRA is based on standards and de-facto technologies, there are extra features that need to be learnt" [B]; "SRA requires the 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 managing the dependencies that an SRA creates to 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 5.6: Summary of benefits of using SRAs.

Benefits	Diagnostic	%	Some further findings from this study
Standardization (B1)	±	33%	(Ben-F) More than the half of software architects, contrary to other stakeholders, indicated that SRAs homogenizes the development and maintenance of applications.
Facilitation (B2)	✓	68%	(Ben-C) Stakeholders claimed that SRA artifacts make easier the development of applications.
Reuse (B3)	✓	82%	(Ben-A) (Ben-B) Stakeholders indicated cost savings in SRA-based applications 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 SRA's software elements have been previously developed, tested and matured.
Enhanced quality (B5)	±	82%	(Ben-E) They highlighted the improvement of architecturally-significant requirements. However, it is not due to the use of an SRA, but any software architecture. Quality attributes clearly promoted by SRAs are reusability and maintainability due to reuse.
Interoperability (B6)	°	-	Although some SRAs integrated new applications with services, legacy applications and other back-ends, stakeholders do not explicitly mentioned it as a benefit.
Knowledge repository (B7)	✓*	67%	SRA designers, as vendors of an SRA, mentioned it only when asked about the reference model. They pointed out the importance of harvested experience from previous successful projects and making it explicit.
Improved communication (B8)	±*	11%	SRA designers, as vendors of an SRA, mentioned it only when asked about the reference model. SRAs help to share an architectural mindset between all stakeholders, even when they are from multiple vendors or work at multiple locations.
Elaboration of mission, vision and strategy (B9)	×	7%	They 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 applications more 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, see Ben-H). However, projects' documentation and 57% of software architects considered that it is an asset given to application builders, and highlighted their importance: "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.

Table 5.7: 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 over the SRA, but they did not strongly word it. A 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, what can 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 the opposite. We consider that since the studied SRAs were used in the 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 to 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 an application builder explicitly mentioned documentation as a benefit.
Bad quality (D6)	±	7%	(Dra-F) Software architects were concerned about consequences from a wrong decision in the SRA. In this context, it 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 less flexible the development of applications.
Learning curve	new	63%	(Dra-A) Additional learning curve for application builders.
Dependency in 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.

SRA benefits and drawbacks by application domain

After the comparison of these results with the literature, we were wondering if there are 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 5.2).

Regarding contextual aspects, we could see that in SRA projects from the banking domain the effort invested was higher, both in terms of persons-month and duration of the projects (i.e., years). With respect to certain benefits and drawbacks, we could not detect any pattern or correlation that determines that some benefits or drawbacks are exclusive of an application domain. However, some effects caused by SRAs were perceived stronger 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 to 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 latest technologies (Ben-D) and dependency over the SRA (Dra-C) was perceived in a higher way in the rest of application domains. In the other domains (i.e., industry and insurance domains), no correlations were found among the application domain and certain benefits/drawbacks.

How to use these results

These results aim to help SRA practitioners as follows.

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

Second, organizations that already have 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 additional learning curve was a commonly mentioned drawback in the *everis* SRA projects.

Third, organizations can find insights about how to improve their SRAs from the responses from the participants. Table 5.8 shows the main improvements that stakeholders mentioned, which type of role is interested the most in each improvement, and to which benefits and drawbacks it affects.

To sum up, organizations of our case study experienced in a different degree certain benefits. However, the achievement of certain benefits depends on the goals that the organization wants to solve with the help of the SRA

Table 5.8: Summary of improvements and trade-off analysis of the benefits that they promote and risks that need to be managed.

Improvement	Asked by	Promotes	Need to manage
Add functionality in the common SRA elements (Imp-A)	Applications builders	Facilitation for applications development and evolution (Ben-C)	Dependency over 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)	Limit the innovation (Dra-B)

(see Table 5.2). For instance, one organization may aim, with a different weight, at standardizing the developments of the applications, easing the application's development or interoperability. As a consequence, we think that every organization should clearly state the benefits they aim to with the SRA (a subset of the benefits of Table 5.6). Then, they can manage the SRA project in order to achieve them. Moreover, it is important to note that these goals may not be static, and can evolve with the time.

5.5 Analysis of Benefits of Reference Models (RQ D)

This subsection presents the results and discussions for the RQ D: "*What are the benefits and drawbacks of adopting SRAs in the industrial practice from the perspective of different stakeholders involved in its design?*" (see RQs in Section 5.1.2).

5.5.1 Results

Categories below come from the analysis of the answers from software architects and architecture developers, as they are the ones that use the *everis* reference model (see *everis* reference model in Chapter 2.4). They were specifically inquired: "What are the main benefits for *everis* from designing many SRAs from their corporate reference model?" Application builders were not asked about such benefits because they do not use the reference model (and sometimes they do not even know that it exists). Fig. 5.9 shows the distribution of responses among those two types of respondents.

- *(Ven-A) everis* harvests experience for prospective SRA projects. The main reason is that requirements are very similar between client organizations. Some respondent estimated 90% of architectural knowledge reused.
- *(Ven-B)* Reusing architectural knowledge can speed up prospective SRA projects and reduce time-to-market, by greatly reducing their planning and development time.
- *(Ven-C)* They gain reputation for prospective client organizations and gain organizational competence. It is a good point for *everis* to be able to announce themselves as SRA providers of, for instance, a large bank.
- *(Ven-D)* Previous experiences reduce the risks in future projects because a "to-be" reference model exists. This model can be used in all projects that do not have very specific architecturally-significant requirements.
- *(Ven-E)* It provides a shared architectural mindset that makes projects less dependent on particular architects.
- *(Ven-F)* It makes tacit knowledge explicit in the reference model. Some tool support (e.g., wiki technologies) helps in managing such knowledge.

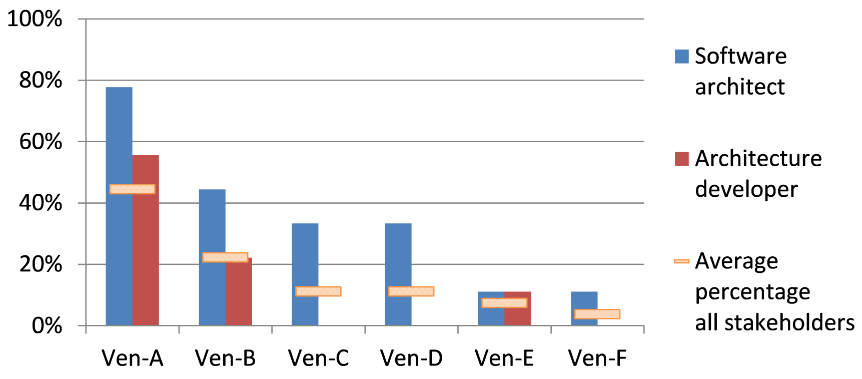

Figure 5.9: Benefits for *everis* from designing SRAs.

Table 5.9: Quotes from respondents about SRA design benefits.

Code	Representative quote
Ven-A	“The reference model receives continuous feedback from SRA projects” [G].
Ven-B	“It is a differential factor because it greatly reduces the planning and development times of the SRA, and hence the costs” [H].
Ven-C	“It gives prestige to announce yourself as provider of successful SRAs” [D].
Ven-D	“It reduces the risk of projects because a 'to-be' model exists” [F].
Ven-E	“The reference model allows architects to have the same way of thinking and working in the software company” [B].
Ven-F	“Architectural knowledge from the reference model is explicitly available in a wiki” [G].

5.5.2 Discussions

As a vendor, *everis* designs SRAs for its client organizations using its corporate reference model, see Figure 5.1. In this context, two theoretical SRA benefits (B7, B8) were only mentioned when asking about the advantages of designing many SRAs (i.e., reference model benefits). Therefore, in their point of view, they are not benefits of SRA use (i.e., the context of *everis'* client organizations), but reference model use (i.e., the context of *everis* itself).

We see very surprising that they have not mentioned having a shared architectural mindset (B8) as a benefit of using an SRA, and only 11% highlighted it as a reference model benefit. In our opinion, improving the communication among multiple stakeholders that 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 such 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 use 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 that have already tested their experiences 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 the top of that SRA, since you know it” [H]. For this reason, the use of reference models is becoming popular among SRA vendors [21].

5.6 Summary of the Second Cycle of RQ 1

With the goal of supporting organizations to understand, evaluate, and improve SRA engineering, this chapter presented empirical evidence from *everis* SRA projects to answer the following questions:

RQ A. Which are the main motives to use an SRA to design systems' concrete architecture in an organization?

RQ B. Which are the artifacts that compose an SRA and how such artifacts are designed, reused, and used?

RQ C. What are the benefits and drawbacks of adopting SRAs in the industrial practice from the perspective of different stakeholders involved in its usage?

RQ D. What are the benefits and drawbacks of adopting SRAs in the industrial practice from the perspective of different stakeholders involved in its design?

Chapter 6

Guidelines for Gathering Evidence of SRAs

In the two previous chapters, which represent the two first cycles of the action research with *everis* regarding RQ 1, we have gathered experience, feedback, and lessons learned. In this chapter, at the end of the formative stage, our goal is to package such experience, feedback, and lessons learned into guidelines for gathering empirical evidence of SRAs.

It is important to note that these guidelines are aimed to be useful not just for *everis*, but also other organizations with a similar context. In order to analyze whether other organizations deal with similar problems as *everis*, we highlighted the similarities of SRAs designed by *everis* with other SRA contexts that were reported in the literature and by practitioners. After this analysis, we were able to only package into the guidelines the material that could be used under other organizations context.

This chapter is organized as follows. Section 6.1 studies the similarities of the context of SRAs in *everis* and SRAs in other firms in industry. Section 6.2 briefly discusses the formative cycles of the action research with respect to RQ 1. Section 6.3 packages the results and provide guidelines that help to answer RQ 1. Finally, Section 6.4 briefly introduces how these guidelines for RQ 1 were validated.

6.1 Similar Contexts of SRAs in Practice

The results from our action research in *everis* are particular to the context described in Chapter 1.2.2 and Chapter 2.4. IT consulting firms, such as Accenture [21] and Capgemini [70], also fit into the context at *everis* (i.e., they use an industry-specific reference model, and they carry out the three types of projects described in Chapter 2.4). However, to properly create the guidelines for other SRA contexts, it is vital to first characterize SRA projects conducted by other companies besides IT consulting firms.

As we mentioned in Chapter 1.1, architecture-centric approaches to develop families of software applications are not new. Deelstra et al. give a classification of these approaches with respect to the level of reuse [105]:

1. standardized infrastructure,
2. platform,
3. software product line, and,
4. configurable product family.

In this classification, SRAs can be positioned as standardized infrastructures or platforms, whereas software product lines and configurable product families are based on product lines architectures. Several authors has stated that SRAs are more generic than product lines architectures (see Chapter 2.3.3). Next, we classify SRAs in the industry under the two former categories.

- On the one hand, *standardized infrastructures* have been used by public administrations in Germany [70], in the Netherlands [55], and in Spain [114]. These SRAs provide software assets as inspiration for the design of applications, but little domain engineering effort is performed (i.e., little domain-specific functionalities are included in the SRA). They are popular in public administration because there is a need to cover multiple organizations from different business domains (i.e., ministries or departments of the government) and little common functionality exist. Also, the high distribution of development teams implies that these SRAs play only an informative or instructive role rather than regulative.
- On the other hand, *platforms* additionally "require a certain amount of domain engineering to create and maintain the assets that implement the common functionality" [105]. There are several business domains that have used this type of SRAs:

– In the space domain, the NASA detected that "many Earth science data system components and architectural patterns are reconstructed for each mission" [71]. To reuse these assets in new systems, they created the NASA Earth science data system SRA [19].

– In the banking domain, SRAs are usually used to integrate legacy systems and new or migrated software systems that contain the business logic. The common scenario is that these SRAs provide common services that then may be reused in the different front-ends or channels (e.g., desktop applications, web client applications, mobile applications, and ATMs). An example is Credit Suisse [109].

– The most mature domain for SRAs may be the embedded systems domain. For instance, Océ, a copier manufacturer, uses an SRA to derive a concrete architecture for engines incorporated in a specific series of Océ printers [50]. Besides, in the automotive industry, car manufacturers such as Volvo [108] and BMW [115] started to use SRAs to develop the software of electronic/engine control unit based on basic software components that were unique to them. As a further step, AUTOSAR has become popular later because it standardizes basic software components for many car manufacturers, suppliers and other related companies [116]. This enables the reuse of software developed by original equipment manufacturers in multiple car manufacturers. This has led to software ecosystems that are characterized by a network of developers rather than a single organization providing the final product [117].

To sum up, we can conclude that the core idea of using an SRA for the development and maintenance of a family of software products is common in the above contexts. However, corporate reference models are only commonly used in IT consulting firms. As a consequence, the study of reference models during the application of the guidelines should be optional so that it can be used in these other contexts.

It is also important to note that all these SRAs are based on practical experience in the industry. This, the guidelines target this type of SRAs, also known as *classical* [4]. Conversely, the guidelines cannot be applicable in *preliminary* SRAs, i.e., those that are "defined when the technology, software solutions, or algorithms demanded for its application do not yet exist in practice by the time of its design" [4].

6.2 Formative Stage: Evolution of the Guidelines

The process of packaging the guidelines have been done incrementally from the feedback of the formative stage. The guidelines have mainly evolved to target more narrowed research questions and objectives. Different previous versions of the guidelines can be seen in [39, 26, 29, 30, 31].

As an example of such evolution, Figure 6.1 shows the guidelines as presented at the "X Workshop Latinoamericano de Ingeniería en Software Experimental" [26]. We can compare this previous version of the guidelines to the current one, summarized in Figure 6.2. The RQ 1 is currently much more narrow, since it does not target the evaluation of SRAs. At the doctoral symposium IDoESE, it was commented that the previous RQ was too ambitious [30]. It can also be observed that the last version of the guidelines focuses on SRAs, rather than reference architectures for any architecture discipline (e.g., enterprise and system, see different architecture disciplines in Chapter 2.1).

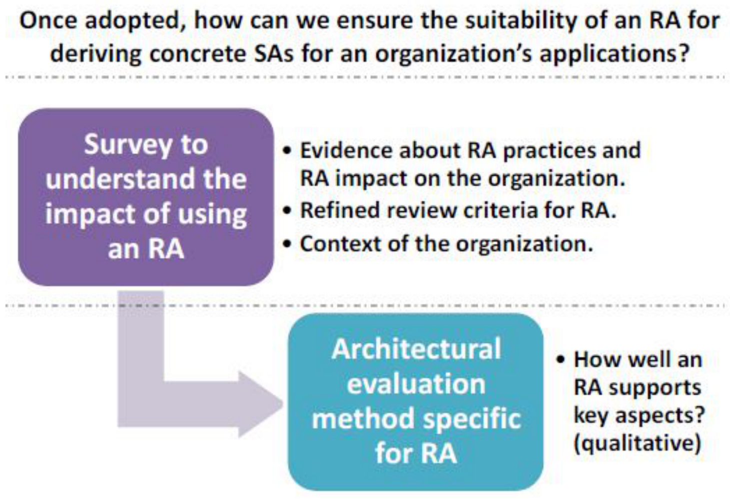
Once adopted, how can we ensure the suitability of an RA for deriving concrete SAs for an organization's applications?
Survey to understand the impact of using an RA
• Evidence about RA practices and RA impact on the organization.
• Refined review criteria for RA.
• Context of the organization.
Architectural evaluation method specific for RA
• How well an RA supports key aspects? (qualitative)

Figure 6.1: Previous version of the guidelines to gather empirical evidence of SRAs in industry [26].

6.3 Packaging the Guidelines

everis' results were suitably packaged with the aim of being applied in other SRA projects and also in similar organizations.

First of all, organizations that may want to use these guidelines need to fit into the context depicted in Section 6.1. This means that they need to design an SRA based on practical experience, and to use such SRA for the development and maintenance of a family of applications in industry. This is because the input for using the guidelines is evidence from real SRA projects.

The guidelines for RQ 1 support organizations to understand, evaluate, and improve SRA engineering based on corporate evidence by providing:

- A practitioner-oriented set of criteria to understand and evaluate an SRA (results of Chapter 4), i.e., a set of relevant aspects for SRAs is facilitated to check which ones are important for the organization.
- Templates of interview guides and online questionnaires to gather relevant evidence (results of Chapter 5). Then, an organization can use the provided template surveys to gather empirical evidence for its own important aspects.

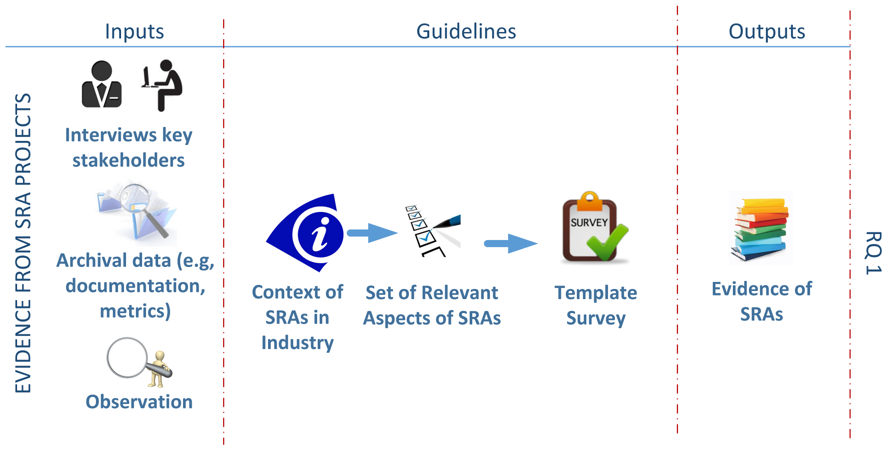
Inputs
Guidelines
Outputs
EVIDENCE FROM SRA PROJECTS
Interviews key stakeholders
Archival data (e.g, documentation, metrics)
Observation
Context of SRAs in Industry
Set of Relevant Aspects of SRAs
SURVEY
Template Survey
Evidence of SRAs
RQ 1

Figure 6.2: Guidelines to gather empirical evidence of SRAs in industry.

Figure 6.2 summarizes the guidelines for gathering evidence of SRAs in order to improve SRA engineering in an organization. The guidelines are composed of the context of SRAs in industry, and materials to conduct two empirical studies. Several data collection techniques exist [118]. For these two empirical studies, the guidelines recommend: a focus group to determine the set of criteria about SRAs important for an organization (see Section 6.3.1), and a case study or survey (designed with a template survey) to gather evidence about the previously identified relevant aspects and to improve SRA engineering (see Section 6.3.2).

6.3.1 A Focus Group to Study the Relevant Criteria of SRAs for an Organization

Below, we explain the context, objective, method, support material, and the output in this empirical study of the guidelines.

- **Context:** Typically, organizations drive the design and use of SRAs in an unsystematic manner [108]. To drive SRA engineering based on evidence, it becomes fundamental to identify the relevant aspects of SRAs as seen by practitioners.

- **Objective:** The objective of this study is to identify the aspects that are important for each organization in order to support SRA engineering.

- **Method:** A focus group with relevant stakeholders (e.g., manager, architect, developer) to find out which aspects of SRAs are important to them. A focus group is considered a proven and tested technique to obtain the perception of a group of selected people on a defined area of interest [118]. An example of conducting this empirical study and its approach for data collection is described in Chapter 4.

- **Support material:** A set of relevant aspects of SRAs (available at the Appendix C).

- **Output:** The relevant aspects of SRAs that are interesting for the organization that carries out the focus group.

6.3.2 A Survey or Case Study to Gather Evidence to Improve SRA Engineering

Next, we explain the details of this empirical study of the guidelines.

- **Context:** To reuse architectural knowledge and improve SRA engineering in prospective SRA projects, organizations need to understand SRA's characteristics, as well as its potential benefits and limitations. Gathering evidence from previous SRA projects is a feasible way to start gaining such an understanding.

- **Objective:** The purpose of the empirical study is to understand the impact of using SRAs for designing the concrete architectures of the applications of an organization. This is a descriptive case study or survey that measures what occurred while using SRAs rather than why. The following questions are important for organizations in order to understand relevant Aspects 1 to 5 of SRAs (defined in Table C.1):

  1. Why is an SRA adapted for creating concrete architectures of the organizations' applications? What type of SRA is being designed and used in the organization?
  2. What is the state of practice in requirement engineering for SRA projects in the organization?
  3. What is the state of practice in architectural design for SRA projects in the organization?
  4. Which tools and technologies are currently being used in SRAs projects by the organization?
  5. How does the adoption of SRAs provide observable benefits to the different actors involved in SRA projects in the organization?

- **Method:** Exploratory surveys or case studies with personalized questionnaires applied to relevant stakeholders (e.g., software architects, application builders,. . .) to gather their perceptions and needs. An example of conducting this empirical study and its approach for data collection is described in Chapter 5.

- **Support material:** Template surveys to gather empirical evidence. Such templates and guidelines for interview guides and questionnaires are available at the Appendix C.

- **Output:** A corporate knowledge base about these aspects.

6.4 Summative Stage: Validating the Guidelines

Once the guidelines were adequately shaped and improved, the summative stage took place. The primary role of this stage was to obtain feedback to check the utility of the guidelines and to validate them with more practitioners. This evaluation consists of the use of the guidelines in other organizations to design and conduct empirical studies. As part of this validation and summative stage of these guidelines, the next two chapters present:

- A survey to gather empirical evidence about the benefits and drawbacks of AUTOSAR, an SRA for automotive applications (see Chapter 7). This survey used the support materials of these guidelines.
- The aggregation of the empirical evidence from *everis* with the empirical evidence from other works (see Chapter 8). Such aggregation proves that empirical evidence gathered by the support materials of these guidelines can be synthesized with other studies.

Organizations facing the design and use of SRAs based on evidence will benefit from these guidelines.

Chapter 7

The Benefits and Drawbacks of AUTOSAR

This chapter aims to validate the guidelines for gathering empirical evidence of SRAs proposed in Chapter 6. To this end, we used the materials of the guidelines (i.e., the practitioner-oriented set of criteria to understand and evaluate an SRA, and templates of interview guides and online questionnaires to gather relevant evidence) out of our industry-academia collaboration with *everis*. As part of the validation and summative stage of RQ 1, we decided to study an SRA widely used and publicly available. As a consequence, we chose AUTOSAR, a mature and accepted SRA for automotive applications used worldwide by more than 180 organizations. We designed a web-based survey by using the guidelines of this PhD thesis, and contacted to 51 practitioners with experience in using AUTOSAR.

This chapter is structured as follows. Section 7.1 provides a background on AUTOSAR. Section 7.2 shows the research methodology of this empirical study. Section 7.3 presents the results of this survey. Section 7.4 discusses limitations of the survey. Finally, Section 7.5 summarizes the contributions of the third action-research cycle of RQ 1.

7.1 Background on Automotive Software and AUTOSAR

Software development for automotive applications has steadily increased over the last decades. In the automotive domain, software is a key area for innovation and development costs. Electronics and software lead over 90% of all innovations and determine up to 40% of a vehicle's development costs, of which 50% to 70% are dedicated for the software of Electronic Control Units (ECU) [18].

Due to the importance of software development for automotive innovation and development costs, the standardization of a software architecture, methodology, software platform, and application interfaces may support to manage growing systems complexity and their integrations, as well as keeping the costs feasible.

Under this scenario, AUTOSAR (AUTomotive Open System ARchitecture) was founded in 2003, and first released in 2005. AUTOSAR is a worldwide development partnership to "establish an open industry standard for the automotive software architecture between suppliers and manufacturers" [119]. The partnership include different types of stakeholders: Original Equipment Manufacturers (OEM), suppliers, tool developers, and new market entrants.

AUTOSAR is an SRA that has become mature and accepted [120]. Due to the success of AUTOSAR in industry, being used by many organizations, we believe that investigating its benefits and drawbacks could help us to validate the guidelines presented in Chapter 6. Our research goal is *to gather evidence of benefits and drawbacks of using AUTOSAR in the industrial practice from different stakeholders involved in its usage*. To get an in-depth understanding of the benefits and drawbacks of AUTOSAR usage for automotive software development, we designed and executed a web-based survey. We obtained 51 valid responses.

The results of this web-based survey could be of interest for researchers and practitioners. On the one hand, for researchers who would like to get insights about the real benefits/drawbacks of this type of SRAs in an industrial setting; in order to better shape their approaches for exploiting such potential benefits and mitigating potential drawbacks. On the other hand, results are relevant not only for AUTOSAR practitioners to get directions for improvement of current drawbacks and risks; but also for practitioners in general that can better understand and polish their expectations from an SRA.

In particular, these results may be relevant for other business domains besides automotive software. For instance, this is the idea of the initiative "derive applications" of AUTOSAR, which aims to extend the scope to non-automotive areas [75].

7.1.1 AUTOSAR Software Reference Architecture

A car includes a number of ECUs or micro-controllers (μC) modules, most of them dedicated to drive sensors and actuators [117]. For instance, the software than run on an ECU can first read data from the car sensors (e.g., engine speed and the speed that is requested by the driver), and then process such data to control actuators (e.g., changing the amount of fuel or the timing or its ignition). This is only an example of application of the 80 to 100 ECUs that today's luxury-class cars include [121].

AUTOSAR provides a layered component-based software architecture to structure the software for an ECU. AUTOSAR is an SRA with these characteristics (see SRA characteristics in Chapter 2.2.2):

1. It aims to *standardize* the ECU software architectures, aiming at components interoperability.

2. It targets *multiple organizations* (e.g., OEMs, suppliers, tool developers and new market entrants) that share the automotive market domain. AUTOSAR is a global standard with 186 partners by March 2015 (91 in Europe, 67 in Asia, 27 in America and 1 in Africa) [116].

3. It is a *classical* SRA that was defined when technology, software, and algorithms required for the software architecture of automotive applications had already been tested in practice.

Figure 7.1 shows that AUTOSAR distinguishes between three main software layers [116]:

1. *Application layer:* it consists of AUTOSAR software components that are mapped on the ECU. AUTOSAR software components are atomic software components of type application software components or sensor/actuator software components. All interactions between AUTOSAR software components are routed through the AUTOSAR runtime environment. The AUTOSAR interface assures the connectivity of software elements surrounding the AUTOSAR runtime environment.

2. *Runtime environment (RTE):* it provides a communication abstraction by providing the same interface and services whether inter-ECU communication channels are used (e.g., CAN, LIN, FlexRay and MOST) or communication stays intra-ECU.

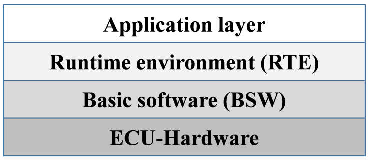

Figure 7.1: AUTOSAR layered ECU component-based software architecture.

3. *Basic software (BSW):* basic software is the standardized software layer, which provides services to the AUTOSAR software components and is necessary to run the functional part of the software. It does not fulfill any functional job itself and is situated below the AUTOSAR runtime environment. For instance, it is responsible for handling the communication between different ECUs on the electronic buses and the diagnostic services which are read when a car is taken to a repair shop.

At the bottom of Figure 7.1, we can see the ECU-hardware resources, and how AUTOSAR offer mechanisms for software and hardware independence.

Related Work of AUTOSAR

Recent research studies have addressed several problems while migrating to AUTOSAR by assisting automotive software designers in planning long term development projects based on multiple AUTOSAR meta-model versions [122], and by migrating a partner's specific, legacy models to their AUTOSAR equivalents [123].

Besides AUTOSAR, other automotive software architectures exist. For instance, JasPar (Japan Automotive Software Platform and Architecture) is an industry partnership with the objective to promote automotive software technology and to cut development costs by encouraging Japanese companies to collaboratively develop non-competitive technologies[1]. Another software architecture for smaller systems is presented in [124]. Concerning standards, several complementary and partly overlapping standards with AUTOSAR (e.g., IP-XACT) are reviewed in [125].

[1]JasPar's site: `https://www.jaspar.jp/`

7.2 Research Method

The guidelines for gathering empirical evidence of SRAs propose to use exploratory surveys or case studies with personalized questionnaires applied to relevant stakeholders (e.g., architects, developers) to gather their perceptions and needs. To capture a snapshot of the current benefits and drawbacks of using AUTOSAR, we performed a web-based survey [126].

We followed the six-step process for surveys defined in [127]. These six steps are survey definition, design, implementation, execution, analysis and packaging.

To ensure rigor and repeatability of our study, and to reduce researcher bias while conducting the survey, we designed a survey protocol. Next subsections briefly present details of such protocol: the research questions of the survey, the target population and sampling, the questionnaire that was devised for data collection, and techniques for data analysis of the survey.

7.2.1 Research Questions

Based on the aforementioned goal of the study in Section 7.1, we devised two research questions:

1. RQ A: Which are the benefits of using AUTOSAR?

2. RQ B: Which are the drawbacks and risks of using AUTOSAR?

7.2.2 Research Design and Sampling

Our population was the global community of practitioners that use AUTOSAR. To recruit participants, we advertised the survey at professional meetings, specifically the 6th AUTOSAR Open Conference celebrated in Munich in 2013. At this conference we collected some responses in situ and also got some contacts to whom we sent an invitation to participate by e-mail. Furthermore, we advertised it in two LinkedIn groups ("Autosar" that has around 5,000 members, and "AUTOSAR" that has more than 1,000 members). Finally, we spread the survey over other social networks (e.g., Twitter) indexed by the hashtag #AUTOSAR. We did not advertise it through academic communities, blogs, conferences or workshops, because we targeted practitioners with experience in AUTOSAR.

7.2.3 Data Collection and Instruments

To devise the instrument to collect the data, we based the questions about benefits and drawbacks and their responses on the support materials available at the Appendix C.

As instrument to collect the data, we decided that an online questionnaire was the most convenient, because it allows the collection of data from a large, remotely-located population, which could be used to contact AUTOSAR practitioners.

The questionnaire of this survey was based on two groups of questions. The first group of questions consisted of two questions about the benefits and drawbacks of AUTOSAR (see Table 7.1). This group was mandatory to fill. We prioritized its simplicity so that it could be filled out in less than 10 minutes. We believe that the simplicity of these questions was key to get a sufficient number of responses. The second group of questions consisted of personal data about the respondent, such as contact information, his/her company, experience, and so on (see Table 7.2). This group of questions was optional. We made it optional because some practitioners are reluctant to provide personal data, and we did not want to discourage them.

It is important to note that we provided room to add any comment or observation in both groups of questions to partially mitigate the rigidness of the online questionnaire.

The survey was available online[2]. For the survey implementation, execution and analysis, we used the same open source tool as in the online questionnaires of Chapter 5: LimeSurvey. In order to get more responses, the survey is still open. We encourage the interested reader with experience in AUTOSAR to refer to the previous link.

7.2.4 Data Analysis

We analyzed the data gathered in each of the categories given as options of the survey. In addition, we gathered several comments as a result of the open questions added. To analyze such comments, we created new categories for refining/polishing the ones given by the survey. These categories were then further discussed and analyzed by the research team to better interpret and describe evidence. Section 7.3 shows this analysis.

[2]http://www.essi.upc.edu/~e-survey/index.php?sid=13916&lang=en

Table 7.1: Group 1 of questions (mandatory).

Id	Question	Options
1	Which are the benefits of using AUTOSAR?*	List of benefits: standardization, facilitation, increased productivity, reuse, reduced development costs, reduced maintenance costs, reduced time-to-market, risk reduction, enhanced quality, interoperability, knowledge repository, improved communication, elaboration of mission, vision and strategy, best practices, novel design solutions, reputation, none, other.
2	Which are the drawbacks and risks of using AUTOSAR?*	List of drawbacks: initial investment, inefficient instantiation, too abstract, term confusion, bad documentation, bad quality, too specific or limiting, learning curve, dependency in AUTOSAR, complexity, none, other.

*Note: These questions were multiple choice, so that the respondent could choose several options. Also, for each choice, the respondent could add a comment.

7.3 Results

We got a total of 51 valid responses[3]. Out of these 51 valid responses, 36 respondents (71%) filled both groups of questions whereas 15 respondents (29%) preferred not to give personal data in the second group of questions.

For the respondents that filled the second group of questions, we had data about their education area, the role of their company with respect to AUTOSAR, and their years of experience with AUTOSAR.

Figure 7.2 shows the education area of the survey respondents: 13 respondents had an automotive background, 11 respondents studied software engineering or related courses, 9 respondents had academic training in electronics, and 3 respondents had other background. 15 respondents did not reply to this question (i.e., n/a). We can see that respondents had higher education, what contribute to a better understanding of AUTOSAR benefits

[3] The valid responses, of which we removed name, e-mail and company of the participants due to confidentiality issues, are available at `http://www.essi.upc.edu/~smartinez/public/responses-WASA15.xls`

Table 7.2: Group 2 of questions (optional).

Id	Question	Options
3	First name and surname	Free text.
4	E-mail	A valid e-mail.
5	Your education area	A list of education areas: automotive, informatics, telecommunications, administration and management, industrial, mathematics, physics, economy, chemistry, statistics, electronics, biology, other.
6	Name of your company	Free text.
7	The role of your company with respect to AUTOSAR	A list of roles: OEM, supplier, tool developer, new entrant market, other.
8	Briefly describe the project in which you have used AUTOSAR	Free text.
9	What was your role in the project? What was your responsibility?	Free text.
10	How many years of experience do you have with AUTOSAR	A valid positive number.
11	Before sending the survey, would you add a comment to help understanding the context of your answers?	Free text.

and drawbacks. Also, we can see that AUTOSAR partners look for recruiting professionals with automotive, software and electronics academic training.

Figure 7.3 shows a pie chart with the role of the company of respondents with respect to AUTOSAR: 12 respondents worked for an OEM, 10 practitioners for a supplier, 8 respondents are tool developers, 4 participants are consultants, and 2 practitioners belonged to a new market entrant. In this survey, we got representatives for all the types of stakeholders in AUTOSAR.

Figure 7.4 shows a box plot with the years of experience of the 36 respondents that replied to the second group of questions of the online questionnaire.

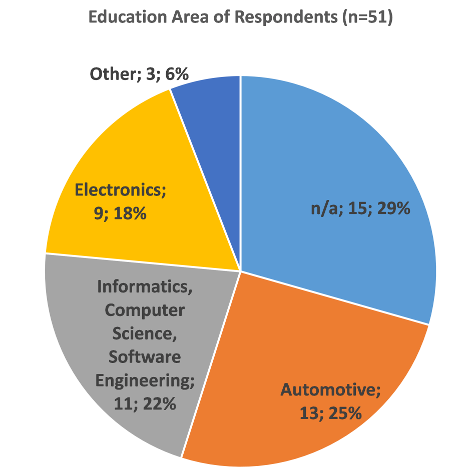

Figure 7.2: Education area of respondents.

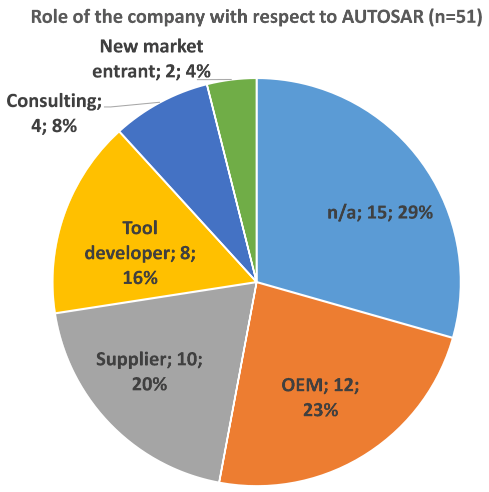

Figure 7.3: Role of the company of respondents with respect to AUTOSAR.

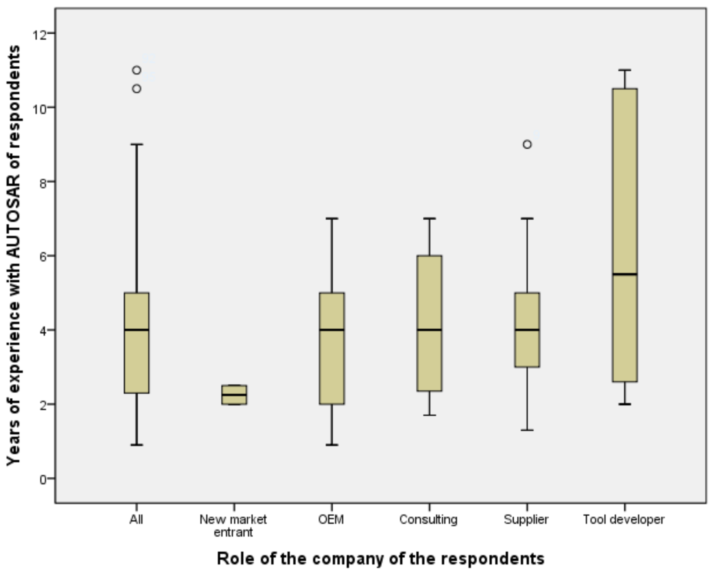

Figure 7.4: Box plot of the years of experience of respondents.

It has six boxes: the first box has all respondents together; the rest of boxes are subsets by the role of the respondents' company with respect to AUTOSAR. In the first box, we can see that there are two respondents with more than 10.5 years of experience in AUTOSAR (extreme cases). The upper quartile is 5 years of experience. The median is 4 years of experience. The lower quartile is 2.15 years of experience. Finally, the minimum is 0.9 years of experience. We can see that respondents had experience in AUTOSAR by the moment of participating in the survey. Finally, tool developers were the respondents with more experience.

Next subsections respectively present the results of the survey about the benefits and drawbacks of AUTOSAR, and the highlights of these results.

7.3.1 RQ A: Results on AUTOSAR Benefits

Figure 7.5 shows the responses about the benefits of AUTOSAR. The X-axis contains the frequency in which respondents mentioned each benefit. The Y-axis represent the options that were given in the online questionnaire as bene-

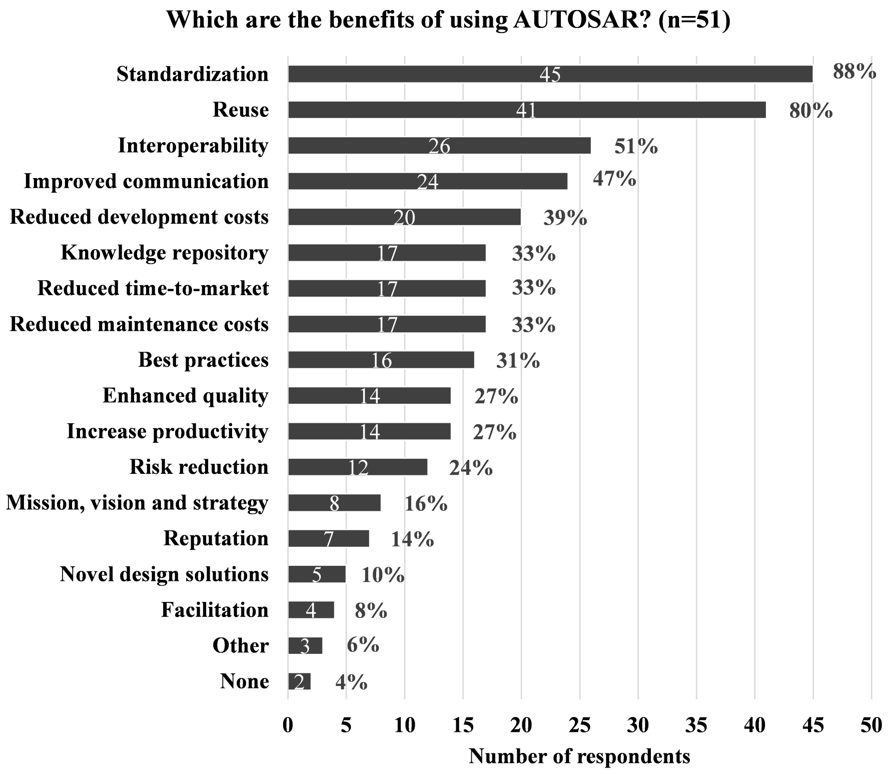

Figure 7.5: Results of the question "Which are the benefits of using AUTOSAR?"

fits. Next, we explain AUTOSAR benefits and provide some of the comments provided by the respondents in the online questionnaire between quotation marks. The benefits are shown in order from the most to the least mentioned one, indicating among brackets the percentage of respondents.

The most mentioned benefit of AUTOSAR was *standardization (88%)*. This is not surprising. Indeed, in its website AUTOSAR is defined as "a de-facto open industry standard for automotive E/E architecture which will serve as a basic infrastructure for the management of functions within both future applications and standard software modules" [116]. This is a relevant benefit, since if a car is compliant with AUTOSAR, the software developed by different stakeholders (e.g., OEM) could be used in many cars, no matter its automotive manufacturer. Some of the respondents commented that standardization is a benefit "if it does not affect competition". A respondent argued that AUTOSAR stakeholders should "cooperate on standards, and compete on implementation". Finally, s/he explicitly stated that standardization is a "trade-off with novel design solutions". This trade-off refers to the "too specific or limiting" drawback.

The second most popular benefit was *reuse (80%)*. As one practitioner stated, "standardized interfaces allows us to reuse components in different projects". Besides the BSW layer (see Figure 7.1), reuse in application software can reach up to 80% [128]. Another practitioner warned that in spite of such reuse, "efforts are often needed, not 100% reuse".

Interoperability (51%) was mentioned as a benefit by half of the respondents. One respondent indicated that it is one of the "goals" of AUTOSAR. Interoperability in AUTOSAR refers to the RTE that acts as a communication center for inter- and intra-ECU information exchange.

The fact that AUTOSAR stakeholders share the same architectural mindset, fosters an *improved communication (47%)*. As one respondent indicated, "people talk the same language".

As one respondent claimed, reuse could lead to "cost and time saving". The results of this survey indicated that *reduced development costs (39%)* is the fifth most popular benefit of AUTOSAR. One practitioner noted that such cost reduction happen "in BSW but also in application software".

AUTOSAR has a lively community that maintains a *knowledge repository (33%)*. Such repository consists of "documents, releases (SVN), and discussions (change management)".

Other benefit related to reuse is the *reduced time-to-market (33%)*. Automotive software can reach the market faster because "component reuse lowers the development time of new products". One practitioner warned that the reuse of a component "reduce time-to-market only if it is already in the standard, otherwise not".

Establishing a standard software architecture helps to *reduce maintenance costs (33%)*.

In a lower extent, respondents supported the following benefits: *best practices (31%)*; *enhanced quality (27%)*; *increase productivity (27%)*; *risk reduction (24%)*; *mission, vision, strategy (16%)*; *reputation (14%)*; *novel design solutions (10%)*; *facilitation (8%)*; *other benefits (6%)*; and *none (4%)*.

Three benefits were written down in the "other" option: "*electronic exchange*", "*scalability* because AUTOSAR was designed from the beginning to handle growing complexity", and "*design flexibility*".

7.3.2 RQ B: Results on AUTOSAR Drawbacks

Figure 7.6 shows the responses about the drawbacks and risks of AUTOSAR in the same way as Figure 7.5.

Below, we explain in descendent order these drawbacks and provide some of the comments given by the respondents in the online questionnaire.

The most mentioned drawback of AUTOSAR in this survey was *complexity (65%)*. Respondents gave several comments about the consequences of complexity, such as that it "is a trade-off with increased productivity". They also gave indications where this complexity gets bigger: "large projects with many developers and highly interconnected functionality is where using AUTOSAR becomes very tough". In the direction of giving suggestions about how to

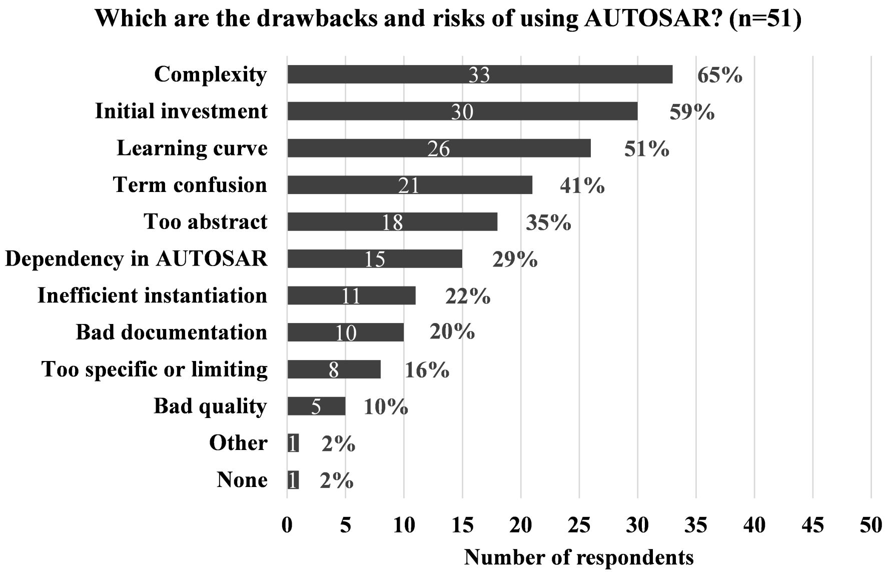

Figure 7.6: Results of the question "Which are the drawbacks and risks of using AUTOSAR?"

handle complexity, two respondents agreed on the importance of tools to ease automotive software development, e.g., "expertise is needed but a tool environment helps. Tools are a must"; "AUTOSAR should be more tool oriented so as to overcome this complexity".

The second most mentioned drawback was *initial investment (59%)*. Due to the characteristics of AUTOSAR, we should not only consider the investment on training personnel on AUTOSAR, but also the "membership fee" to become a partner as organization.

The *learning curve (51%)* to master in AUTOSAR was mentioned by half of the respondents. As one respondent stated: "many engineers have difficulty learning the standard".

Practitioners also face problems with *term confusion (41%)*.

Some respondents found AUTOSAR *too abstract (35%)*. As a solution to overcome abstraction, a practitioner proposed a "tool environment" (as to overcome with complexity).

All the developments based on the standard have *dependency in AUTOSAR (29%)*. Automotive software systems based on AUTOSAR are "statically defined systems". Therefore, new releases of AUTOSAR should consider "looking for backward compatibility".

In a lower extent, respondents indicated the drawbacks below: *inefficient instantiation (22%)*; *bad documentation (20%)*, however a practitioner indicated that there is "no bad documentation (about 20,000 pages of documentation), and that such documents are available for the community", hence they may refer to a more digestive or lightweight documentation; *too specific or limiting (16%)*, e.g., "as a design philosophy AUTOSAR is a desirable standard. However AUTOSAR specifies too many things and leaves little latitude for custom components in all layers beneath the application software. This is not a model that all OEMs can work with effectively"; *bad quality (10%)*; *other drawbacks (2%)*, and *none (2%)*.

The drawback that was mentioned in the "other" option was "*repetitive investment*" because "it is hard and costly to migrate to a new AUTOSAR version". This extra cost while migrating was also mentioned by another practitioner: "we just started migrating towards AUTOSAR, and found that even after 10 years, it makes delays and confusion and instead of increasing the quality it reduces it. Also, the cost of the tools is high".

7.3.3 Highlights of the Results

These results help to increase the empirical evidence about SRA as follows.

First, this survey uncovered AUTOSAR benefits, being the most popular ones standardization (88%), reuse (80%) and interoperability (51%). With respect to the drawbacks of AUTOSAR, the study revealed mainly complexity (65%), initial investment (59%) and learning curve (51%).

Second, survey respondents gave directions to handle the major drawbacks. Results about the drawbacks of AUTOSAR show that experience reports about negative experiences are also needed.

- With respect to complexity, they remarked that AUTOSAR should be more tool oriented to improve its usability. Several initiatives are already working on making AUTOSAR less complex and improving the tool environment, e.g., the AUTOSAR Tool Platform (Artop)[4].

- Furthermore, the repetitive investment while migrating to a new release of AUTOSAR was uncovered as a drawback of SRAs. This drawback was not reported in previous studies of SRAs (see Chapter 3.3.2). It becomes necessary to balance between stability and updates of AUTOSAR, since some practitioners find that there are too many releases. This leads to a costly migration to new AUTOSAR versions. Recent research have addressed this issue by assisting automotive software designers in planning long term development projects based on multiple AUTOSAR meta-model versions [122], and by migrating a partner's specific, legacy models to their AUTOSAR equivalents [123].

7.4 Validity

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

7.4.1 Construct Validity

To strengthen this aspect we made sure to perform a rigorous planning of the study and establishing a rigorous protocol. We paid special attention to design our data collection instrument (i.e., the online questionnaire) in such a way that it was fully understood by the respondents. We made sure of polishing the instruments with suitable vocabulary that the participants were familiar with. Furthermore, we included specific mitigation actions for evaluation

[4]AUTOSAR Tool Platform User Group: `http://www.artop.org/`

apprehension by ensuring the confidentiality and aggregation of the answers, so the respondents could freely share their real perceptions. In the online questionnaire, we added open questions to let respondents to express the response that better reflected their opinion.

7.4.2 Internal Validity

Regarding individuals that participated in the study, there is always the possibility that they forget something or do not explicitly state it when they are asked about. To reduce this risk, we designed the online questionnaire in such a way that the respondent must answer all the corresponding questions while s/he could complete the questionnaire at any time, so it gives them the possibility of consulting registries and documentation in case s/he needs to remember something.

Another limitation regarding the participants is that they might not have answered truthfully to the questions. To address this problem, we made participation voluntary and ensured that personal data would be treated confidentially. Furthermore, participants spent personal time on answering the online questionnaire. We can therefore assume that those who volunteered to spend time have no reason to be dishonest [117]. Still, there were couple of responses that were removed because it was clear that they were invalid (e.g., just indicating none benefits and none drawbacks, or introducing fake personal data). One reason may be that they just entered to see the questions of the survey.

Furthermore, when using surveys like this, there exists always the threat that respondents tend to be strong supporters or strong opponents of the analyzed technology; thus biasing the results. To reduce this threat, we tried hard to foster many people to participate by attending to an AUTOSAR related conference and explaining them the importance of having the opinion of all of them. In addition, we added in the online questionnaire the group of questions about personal data (see Table 7.2) in order to further contact them in cases where we detect suspicious situations. Most of the respondents replied to these questions.

Also, to reduce the potential researcher bias, several meetings were held among the researchers to discuss the course of the data analysis and the preliminary results.

7.4.3 External Validity

We had a limited number of participants. However, this is due to the fact that our survey targeted a very specific population and required participants with experience with AUTOSAR. The participation of this study (51 participants) compared to other empirical studies in software architecture is similar [129]. In [129], the authors analyze the sampling of four studies with the following participation: 56 participants, 11 software companies, 53 industrial software architects, and 22 students.

We recognize that our results cannot be generalized to other SRAs without further work. However, we remark that there exist organizations with similar contexts to AUTOSAR that could benefit from the results of this survey [31]. Thus, we made available our instrument (see Section 7.2) to foster other researchers and practitioners to use them and compare results. We expect that our results strengthen the evidence regarding SRAs and encourage others to provide similar evidences that help to mature SRA research and practice.

7.5 Summary of the Third Cycle of RQ 1

With the goal of validating the guidelines for gathering empirical evidence of SRAs, this chapter presented a web-based survey based on such guidelines. The web-based survey analyzes how AUTOSAR, a mature and accepted SRA for automotive applications used worldwide by more than 180 organizations, is perceived by industrial practitioners.

With a successful design and conduction of the web-based survey, we validated the guidelines for gathering evidence of SRAs.

Chapter 8

Aggregating Empirical Evidence of SRAs

After applying the guidelines of Chapter 6 to gather evidence of SRA engineering in many organizations, it becomes necessary to aggregate such evidence from diverse empirical studies. For instance, regarding the benefits and drawbacks of SRAs, we have different cases at *everis* (Chapter 5) and AUTOSAR (Chapter 7). Besides, other researchers have also empirically studied SRAs benefits and drawbacks in other contexts. Specifically, previous empirical studies have reported the strengths and weaknesses of SRAs to ease their industrial uptake [3, 22, 37, 40, 55]. However, the results of these single empirical studies have not been analyzed together. Therefore, there is not a consolidated and unified evidence about the benefits and drawbacks of SRA adoption.

The main goal of this chapter is to strengthen the evidence about the benefits and drawbacks of SRAs. This can be done with the Structured Synthesis Method (SSM) [130], which allows performing a research synthesis study of existing evidence. The SSM consists of a method able to aggregate qualitative and quantitative evidence through the use of diagrammatic models. Such synthesis helps to gain more confidence on the effects of SRAs that have been reported by more than one empirical study, and to determine the context of those effects that only appear under specific contexts of SRAs. The aggregated evidence of this chapter aims to help practitioners to understand and analyze the benefits and drawbacks of adopting and using SRAs in their organizations, and to support researchers to identify areas where further research is needed to consolidate/understand the actual evidence.

This chapter is structured as follows. Section 8.1 presents the research methodology: the SSM. Section 8.2 represents evidence that was reported in previous single studies. Section 8.3 aggregates the empirical evidence modeled in Section 8.2. Section 8.4 discusses the results of this study. Section 8.5 discusses the threats to validity of this study. Finally, Section 8.6 summarizes the contributions of the fourth action-research cycle of RQ 1.

8.1 Methodology

Due to the existence of many empirical studies, we need to analyze the trends on available empirical evidence about the benefits and drawbacks of SRAs for acquisition organizations. We focused on the benefits and drawbacks for organizations that introduce an SRA for designing and constructing a family of software systems. Therefore, we focused on the SRA "usage" perspective, rather than other perspectives, e.g., SRA "design".

To this end, we aggregated the research results of previous works by using the SSM [130]. As both qualitative and quantitative research synthesis method, the SSM briefly depicts the important contextual aspects, and informs the trend of the effects (e.g., positive or negative), as well as a certain estimation about them. Therefore, SSM neither aggregates precise quantitative findings nor rich qualitative descriptions.

We decided to use the SSM because it is able to conceptualize about the context, and to integrate studies' results. Therefore, it has an interesting blend of integrative and interpretive synthesis [131]. In the SSM, interpretative synthesis aspects are concerned with the organization and development of concepts to describe contextual aspects of evidence whereas integrative features are focused on pooling data about *cause-effect* or *moderation* relations. Moreover, studies about SRA benefits and drawbacks report both qualitative and quantitative evidence. The SSM can aggregate these types of evidence, and it takes into account the uncertainty estimated for each evidence. Besides, the SSM offers tool support to model and synthesize evidence [132], including facilities for graphical modeling, evidence search, and support for the synthesis. Another important functionality is the evidence model comparison used to aggregate evidence, which has mechanisms for 'conflict resolution' between the models. The tool and all the results of the synthesis presented in this chapter can be accessed at: `http://evidencefactory.lens-ese.cos.ufrj.br/`.

The SSM is composed of three main phases. First, papers are identified and selected according to predefined criteria. Existing approaches for study selection (e.g., SLR search process) can be used for this purpose. Second, information from each paper is extracted, and the pieces of data are organized and put into the same perspective. Evidence are modeled with a diagrammatic representation, which describes the context, *cause-effect* relationships, and their *moderators*. Each model can have more than one *cause-effect*, so multiple outcomes/effects can be analyzed together in the same aggregation. Third, with all evidence under the same perspective, the last phase is dedicated to consolidate and synthesize the results. This synthesis shows what the main trends or conflicts among the analyzed evidence are. The primary interest of the SSM is to combine *cause-effect* relationships from many SE empirical studies.

Next, we report how we applied the SSM method to synthesize the research on SRA benefits and drawbacks.

8.1.1 Step 1: Selecting Primary Studies

To select the primary studies, we defined a systematic search strategy, and the inclusion and exclusion criteria.

a) Search strategy: For the search strategy, we considered the same data sources and search string as in the SLR about SRA engineering of Chapter 3.1. Also, we used the forward and backward snowballing strategy for the included papers. Experts or reviewers suggestions were also accepted, which is essential for studies that are not indexed or published yet (i.e., in press).

b) Inclusion and exclusion criteria: The SSM is flexible regarding the type of data collected in studies and the amount of their outcomes. So, as inclusion criteria, we defined any empirical study reporting findings based on evidence about the benefits and drawbacks of adopting an SRA.

Concerning exclusion, we defined three exclusion criteria: (i) studies whose findings were based on opinions rather than evidence; (ii) studies that were not the primary source of the reported study or data (i.e., secondary studies); and, (iii) studies that were not reported in English.

c) Study selection: The search string, performed in September 2014, retrieved 492 non-duplicated studies. From these studies, the empirical studies reporting evidence were manually identified. From this list, we looked for those focusing on reporting the benefits and drawbacks of SRAs.

Three papers that report empirically grounded results about SRA benefits and drawbacks were found [3, 22, 37]. Searching through the references and citations of these three papers, [55] was added to the included studies. Also, [40] was included by convenience, as it was not published yet, but we were aware of its existence because it was conducted by three of the authors.

Finally, we ended up with five included primary studies reporting evidence on the benefits and drawbacks of using an SRA in an organization. The most important details of each of the five papers can be found in Table 8.1.

8.1.2 Step 2: Evidence Representation

The SSM uses a diagrammatic representation to support the aggregation of evidence. Following the understanding of most research synthesis methods [131], the idea is that once all evidence are put under the same format their combinability can be better analyzed, and the decision for combination more objective. The representation used in SSM is called *theoretical structure* and, as the name suggests, it is based in the notion of theories, from which the SSM stems its capability of accommodating most diverse types of evidence.

The ten semantic constructs used in the representation are shown in Figure 8.1. There are three possible types of *structural* relationships in the representation: *is a*, *part of* and *property of*. All of them have counterparts in UML, respectively: generalization, composition, and class attributes. The *is a* and *part of* relationships use the same UML notation for generalization and composition. *Properties* are denoted by dashed connections. The relationships are used to link two types of concepts – *value* and *variable*.

A *value* concept represents a particular variable value, usually an independent variable. Value concepts are represented by rectangles, and they are classified in *archetypes* (the root of each hierarchy), *causes* (indicated by the use of a bold font and a 'C1' following the name denoting that it is the 'cause 1' (e.g., 'Reference Architecture'), and *contextual aspects* (e.g., 'Enterprise Software'). The four archetypes – activity, actor, system, and technology – were suggested by Sjøberg et al. [133] in an attempt to capture the typical scenario in SE described by an actor applying a technology to perform activities in a software system.

A *variable* concept focuses on value variations usually associated with a dependent variable. *Variable* concepts are represented by ellipses or parallelograms symbolizing *effects* (e.g., 'communication') and *moderators* (e.g., 'maturity'), respectively. In addition, *effects* are not connected to *cause* using lines as they are assumed to exist when reading the diagram. Lines are also lacking in the link between *moderators* and the (moderated) *effects*. In this case, a textual hint (e.g., 'M1') is shown besides both the moderated effect and moderator. Both relationships, *cause-effect* and *moderation*, are denominated *influence* relationships.

Table 8.1: Primary studies

Study Id.	**Study Type: Instruments**	**Participants**	**SRA Application Domain**	**SRA goal**[a]	**SRA used in**[a]	**SRA type**[a]	**Belief**[b] **&evidence type**	**Year**
S1 [3]	Expert meeting: presentations, discussions	Architects from the System Architecture Forum	Defense and commercial equipment	Standard. & Facilitation	Single & multiple Organizations	Preliminary & classical	0.25+0.10=0.35 qualitative	2010
S2 [37]	Case study: interviews, questionnaires, docs.	28 sw. architects and developers from IT consulting	Banks, insurers, public administration, utilities, and industries	Standard. & Facilitation	Single Organizations	Classical	0.25+0.19=0.44 qualitative & quantitative	2013
S3 [22]	Survey: questionnaires	90 sw. architects and developers from worldwide	n/a	Standard. & Facilitation	Single & multiple Organizations	Preliminary & classical	0.25+0.15=0.40 qualitative & quantitative	2013
S4 [55]	Case study: interviews, docs., meetings	20 sw. architects, managers and experts from locale-goverment	Variability-intensive service-oriented systems	Facilitation	Multiple Organizations	Classical	0.25+0.15=0.40 qualitative	2013
S5 [40]	Survey: questionnaires	51 practitioners from AUTOSAR partners	Automotive systems	Standard.	Multiple Organizations	Classical	0.25+0.17=0.42 qualitative & quantitative	2015

[a]To check the possible values for SRA goal, used in, and type, see Chapter 2.2.2. [b]The belief value is calculated as shown at the end of this section.

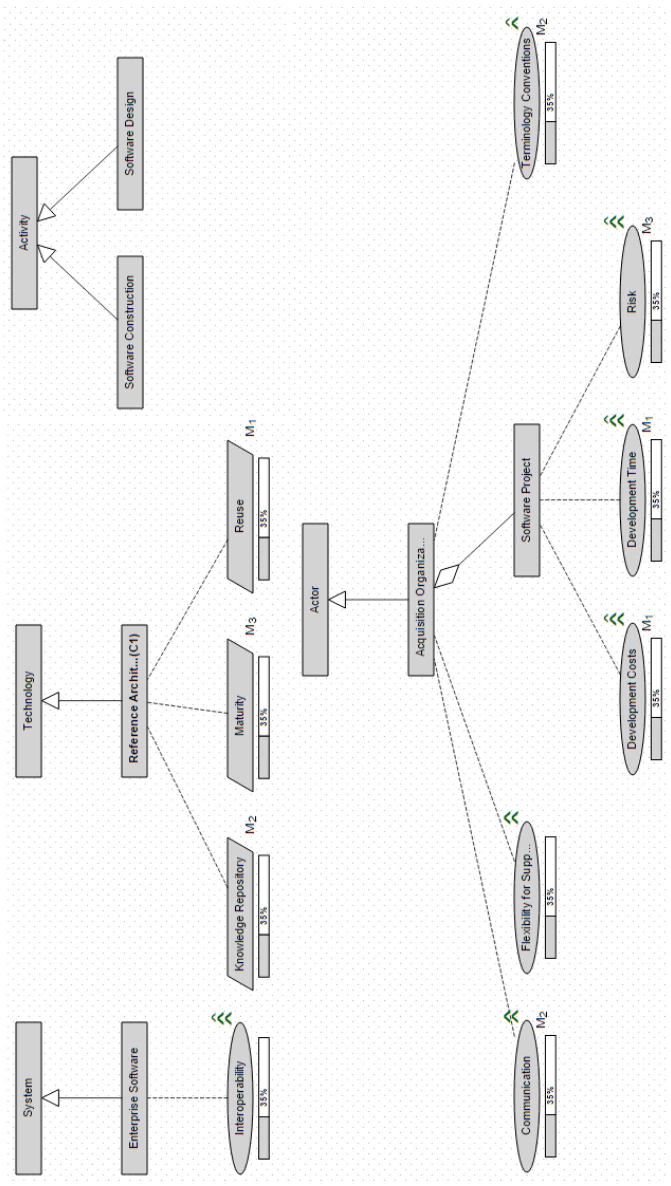

Figure 8.1: Evidence model representing the results of the study S1.

To indicate the effect size, a seven-point Likert scale is used. The scale ranges from strongly negative to strongly positive and is indicated above the ellipse (e.g., '︽' indicates that 'Flexibility for Suppliers' is positively affected by 'Reference Architecture'). The other type of *variable* concepts, namely *moderators*, indicates that some positive or negative effect is moderated (i.e., reduced) when it increases or decreases. For instance, a moderator is how a 'knowledge repository' influences 'communication'. A last aspect related to *variable* concepts is the association of a *belief value* (ranging from 0% to 100%) to estimate the confidence in the observed effects and moderations. The bar under each element represents the belief value, e.g., 'interoperability' has 35% belief value.

a) Extracting data to model evidence: The data extraction and evidence modeling activities are intertwined and, together, are very similar to the text coding and analysis process [134]. The major orientation in creating the theoretical structures comes from the *thematic synthesis* and its increasing abstraction level, where text is translated into codes, which are translated into concepts and relations, and, from them, the *theoretical structure* representing an evidence is modeled. The SSM also contains recommendations from *meta-ethnography*, such as how the text should be coded, and papers translated into one to another to identify concepts and relations. The inductive approach from *qualitative comparative analysis*, where concepts are identified inductively from the collection of studies, complements these recommendations. To improve the synthesis reliability, the participation of more than one researcher is recommended as is in case studies and many other qualitative methods. A resume of all these research synthesis methods can be obtained in [131]. Last, instructions for identifying *cause-effect* relationships are also included, since they put qualitative and quantitative evidence in the same perspective. The instructions are based on [135]: qualitative research "explains individual cases; using the causes-of-effects approach" whereas quantitative research "estimates average effects of independent variables; using the effects-of-causes approach".

Although these heuristics for evidence modeling are used to make the process more systematic and transparent, evidence modeling in SSM is still a subjective process with some influence of the researcher abstraction skills, and knowledge about the topic of interest. Nevertheless, all these orientations were used to model the evidence related to the five studies identified. Two researchers divided the five papers into two sets and then individually modeled each evidence. After that, we reviewed the models created by each other, including several meetings to discuss whether we had a common understanding about the models. The other three researchers (in total a group of five) performed a final revision of the models, and the resulting aggregated model.

It is interesting to notice that the identification of concepts and relationships is an iterative process, and the modeling of evidence can be a trigger to review the others. This is important to make concepts and evidence structures more consistent, and particularly important for evidence synthesis, which is described in the next subsection. A last step of data extraction is the evidence quality assessment, which is used for estimating a *belief value* used in the synthesis (also described in next subsection).

8.1.3 Step 3: Evidence Synthesis

To aggregate evidence, it is necessary to determine evidence combinability. For that, all *value* concepts (*archetypes*, *cause* and *contextual aspects*) and *structural* relationships (*is a*, *part of* and *property of*) between the different models must match. For instance, if an evidence model describe that 'Enterprise Software' is *type of* 'System' as part of the context, then the other evidence model should have a the same relationship as part of its context description. If there is not a direct correspondence with the use of the same concepts (in the example, 'Enterprise Software'), the researcher can decide if their meaning are similar enough for the aggregation purposes, and still aggregate the evidence. The other option is to keep both concepts separated in the aggregated results – that is, the effects associated with 'Enterprise Software' in an evidence model are not aggregated with the effects associated with the system described in the other model. A third option, when an aspect is only present in a model, is to add or remove the concept from the aggregation. Then, the resulting aggregated evidence *value* concepts and *structural* relationships are defined.

After determining which evidence can be combined, and grouping the ones that can, uncertainty formalism is necessary to combine the results – otherwise, a simple vote counting strategy would be used. In the SSM, the Mathematical Theory of Evidence [136] (also known as Dempster-Shafer theory, DST) is the mathematical formalism that enable to combine results. While *value* concepts are used to determine aggregability, the aggregation itself is focused on the *variable* concepts and their relationships (*cause-effect* and *moderation*). DST uses two main inputs to combine two pieces of evidence. One is the hypotheses believed to have a chance to be true – belief value greater than zero – and the other is the belief values themselves. Hypotheses are defined as sets of the powerset of the defined frame of discernment elements, which in the case of SSM is formed by the values of the seven-point Likert scale: Θ = {SN, NE, WN, IF, WP, PO, SP} – the elements values are abbreviations for the Likert scale terms, e.g., SN is 'strongly negative', IF is 'indifferent', and WP is 'weakly positive'. It is interesting to notice that since hypotheses are sets from the powerset, a hypothesis can be a singleton (e.g., {PO}) or a compound set (e.g., {WN, PO} – meaning an imprecision about weakly positive and positive).

The other input is the belief value assigned to each hypothesis. Belief values are estimated using the study type and a quality assessment. First, based on GRADE evidence hierarchy [137], study type level split the 0-1 belief value range into four subranges: unsystematic observations [0.00, 0.25]; observational studies [0.25, 0.50]; quasi-experiments [0.50, 0.75]; and randomized controlled [0.75, 1]. Second, the quality assessment value is translated to the 0.25 subrange. The SSM method proposes to use two checklists to assess the quality of each study, which are explained in [130]. Based on this, the belief values listed in Table 8.1 are calculated, e.g., the study S1 was observational (0.25), and in the quality assessment done by the checklists it got 0.10 out of 0.25. The reader is referred to the tool to check the quality assessment questionnaire for each study.

Once hypotheses and belief values are defined for each evidence, then the *Dempster's Rule of Combination* is applied, see equation 8.1. Equation 8.1 shows that the aggregated belief value for each hypothesis C is equal to the sum of the product of the hypotheses belief values whose intersection between all hypotheses A$_i$ and B$_j$ of both evidence is C.

$$m_3(C) = \frac{\sum\limits_{\substack{i,j \\ A_i \cap B_j = C}} m_1(A_i) \times m_2(B_j)}{1 - K}, \textit{where} \quad K = \sum_{\substack{i,j \\ A_i \cap B_j = \emptyset}} m_1(A_i) \times m_2(B_j) \tag{8.1}$$

When the intersection between two hypotheses is an empty set, we say that there is a conflict. Conflict is, then, redistributed to the aggregated hypotheses – that is the function of 1 - K in the denominator. More details about how DST is used in SSM can be obtained in [130].

8.2 Representation of SRA Effects

In this section, we show how we extracted the evidence from the included studies, and created models to represent such evidence. Evidence modeling has a significant interpretation, coding, reasoning and analysis of components. We translated the evidence from text-based studies into evidence models that are diagrams.

We describe the model associated with the study S1 in this chapter (see Figure 8.1). It is the shortest model (fewer concepts and relationships), and it includes all ten semantic constructs detailed in Section 8.1.2. In the study S1, the driving forces for SRAs are elicited from the discussions of the System Architect Forum (`http://architectingforum.org`). The authors also present real-world SRAs from different domains to help justifying some of the driving forces elicited. Since the results presented in the study S1 are an outcome of an analysis of discussion between professionals with different background, we decided to use general *value* concept for context description including 'Enterprise Software' and 'Acquisition Organization'. In models of other studies where the context is specific, such as the study S5 describing an SRA for the automotive domain, specific value concepts were used to model it (e.g., 'Automotive Software', and 'AUTOSAR partner').

Apart from the context description, the effects were relatively straightforward to identify as they were listed in the text of the study. For instance, 'Terminology Conventions' concept in S1 was identified from the following excerpt: '*Reference Architecture can also serve as a framework and lexicon of terms and naming conventions, as well as structural relationships within a company, industry or a domain*'. In fact, this is usually expected since describing the study results is one of the most important parts of any paper. For instance, we can see in Figure 8.1 that an SRA *is a* technology. Moreover, the 'C1' notation indicates that SRAs are the *cause* of all the *effects* (represented by *ellipses*) in the model. However, moderators were not so unequivocal, since the authors do not report them as moderators, but rather as particular conditions important to augment some effects. As an example, we can see that SRAs have a positive – strongly positive influence on the 'Development Costs' of a 'Software Project' (35% belief). The M1 notation indicates that such influence has a *moderator* (represented by parallelograms), which is 'Reuse'.

All evidence models created for each paper can be found on the tool at the link previously informed. To make this document self-contained, Table 8.2 provides a summary of each evidence model with the list of all effects caused by SRAs. For each study, we give the effect intensity in the Likert scale along with the belief value. For instance, 'Interoperability' was reported by study S1 as positive – strongly positive effect of SRAs with a 35% of belief. Studies S3, S4, and S5 also reported 'Interoperability'.

Regarding the effect intensity, it is interesting to say that we had to interpret it from the textual descriptions. Thus, if the textual description did not qualify the effect with particular adjectives indicating the intensity, then we chose a default value (e.g., PO), rather than a weak (e.g., WP) or strong value (e.g.,

Table 8.2: SRAs effects as reported in selected studies.

	Representation of evidence from single studies, shown as: intensity (belief value)				
Effect	**S1**	**S2**	**S3**	**S4**	**S5**
Interoperability	PO, SP (0.35)		PO (0.15)	WP (0.40)	PO, SP (0.22)
Development costs	PO, SP (0.35)	PO (0.36)	PO (0.04)		PO (0.16)
Communication	PO (0.35)		PO (0.09)	PO (0.40)	PO, SP (0.20)
Risk	PO, SP (0.35)			PO (0.40)	PO (0.10)
Best practices			PO (0.31)	PO (0.40)	PO (0.13)
Learning curve		SN, NE (0.36)	NE (0.13)	NE, WN (0.40)	NE (0.22)
Development time	PO, SP (0.35)		PO (0.14)		PO (0.14)
Maintenance cost		PO (0.35)			PO (0.14)
Productivity		PO, SP (0.30)			PO (0.11)
Ease of developing		PO (0.30)	PO (0.07)		WP, PO (0.03)
Alignment		WP, PO (0.19)			WP (0.07)
Restriction		NE (0.13)	NE (0.06)		NE, WN (0.07)
Standardization		WP, PO (0.14)	PO (0.16)	WP (0.40)	SP (0.37)
Latest technologies		WP (0.30)			
Investment					NE (0.25)
Reliability		WP, PO (0.14)			
Dependability		SN, NE (0.09)			NE, WN (0.12)
Reputation					WP (0.06)
Software quality			NE (0.06)		WN (0.04)
Novel design solution			PO (0.05)		WP (0.04)
Complexity		WN (0.06)			SN, NE (0.27)
Terminology conventions	WP, PO (0.35)				NE (0.17)
Flexibility of suppliers	PO (0.35)			WN, IF (0.40)	

SP). Similarly, if the paper gave an ambiguous description we defined a lower range for the effect intensity (e.g., WP, PO). For instance, the 'learning curve' drawback in the study S2 is described as *"additional high or medium learning curve for using the SRA features"*. The belief values were based on the study type and the quality assessment as described in Section 8.1.3. Yet, in survey papers (which also report quantitative evidence), we weighted the belief values with the number of respondents that actually perceived the effect as result of SRA usage. This weighting was performed using the DST discount operation. The idea of the discount operation is to adjust the mass distribution (i.e., the belief values assigned to the hypotheses) to reflect the source's credibility – a full discount (discount=1) represents a completely unreliable source. For instance, for survey studies, we used the number of respondents as estimation for the discount value calculated as: (1 - number of respondents for the question / total participants). The DST discount operation is also used in studies considering *p-values*.

The evidence modeling process produces interesting results from the aggregation perspective: the individual results provide a basic understanding to combine results. In addition, since results are already translated into diagrams in a more condensed form, they can be practical for other uses.

8.3 Results of the Aggregation

Once we individually processed the evidence of the selected studies in the previous section, in this section we show how we aggregated the results.

Table 8.3 shows the results after performing the aggregation of evidence on the benefits and drawbacks of SRAs. The first column shows the reported effect (i.e., benefit or drawback) caused by the introduction of an SRA in the organization. The second column indicates the number of papers that have reported this effect. The third column shows the aggregated intensity about how the SRA causes such effect (e.g., positive or negative). The fourth column represents the aggregated belief on such effect. This is one of the most interesting results of the aggregation. The individual study with the highest belief for an effect was S4, with 40% belief for the 'Interoperability' effect (see Table 8.2). However, after aggregating the results from single empirical studies, some effects caused by SRAs were reinforced. Table 8.3 shows in bold those effects that have higher belief of 40% after the aggregation. The fifth column shows whether there was a conflict while aggregating that effect. This is important to analyze and to characterize different contexts from which evidence was gathered. Lastly, the sixth column shows the difference between the maximum value of the belief in individual papers and the gained confidence after the aggregation. Therefore, a positive difference indicates the effects that have been reinforced after the aggregation whereas a negative difference shows that evidence is somewhat contradictory. The effects are ordered by the difference on the belief after the aggregation.

Aggregation was performed using the Dempster's Rule of combination (see Equation 8.1). For instance, Maintenance Cost effect was computed in the following manner: $m_{aggregated}(\{PO\}) = m_{s1}(\{PO\}) \times m_{s2}(\{PO\}) + m_{s1}(\{PO\}) \times m_{s2}(\Theta) + m_{s1}(\Theta) \times m_{s2}(\{PO\}) = 0.049 + 0.301 + 0.091 = 0.441$. This is the value

Table 8.3: Aggregated effects of SRAs (ordered by belief strengthening).

Effect caused by an SRA	**Aggregation Results**				
	#Papers	***Intensity***	***Belief***	***Conflict***	***Difference***[a]
Interoperability	4	PO, SP	**74%**	-	34%
Development costs	4	PO, SP	**67%**	-	31%
Communication	4	PO	**65%**	-	25%
Risk	3	PO, SP	**65%**	-	25%
Best practices	3	PO	**64%**	-	24%
Learning curve	4	NE, WN	**60%**	-	20%
Development time	3	PO, SP	**52%**	-	17%
Maintenance cost	2	PO	**44%**	-	9%
Productivity	2	PO	38%	-	8%
Ease of developing	3	PO	35%	-	5%
Alignment	2	WP, PO	24%	-	5%
Restriction	3	NE	18%	-	5%
Standardization	4	WP, PO	43%	-	3%
Latest technologies	1	WP	30%	-	0%
Investment	1	NE	25%	-	0%
Reliability	1	WP, PO	14%	-	0%
Dependability	2	NE, WN	12%	-	0%
Reputation	1	WP	6%	-	0%
Software quality	2	NE	6%	-	0%
Novel design solution	2	PO	5%	-	0%
Complexity	2	SN, NE	26%	0.017	-1%
Terminology conventions	2	WP, PO	31%	0.060	-4%
Flexibility of suppliers	2	WN, IF	31%	0.140	-9%

[a]The "Difference" column measures the difference among the max value of belief in previous single papers, and the gained confidence after the aggregation.

found in Table 8.3. It should be noticed that for cases where more than one intensity is involved, the belief function is used (see [130] for details).

Next, we respectively report the effects that: a) increased, b) slightly increased, c) did not change, and d) decreased their belief after the aggregation.

8.3.1 Effects of SRAs that Increased their Belief

Seven effects caused by SRAs increased their belief values after the aggregation. These effects have greater confidence value than any effect before aggregation (i.e., greater confidence level than 40%, see Table 8.2), and have been reported by at least three out of the five studies. Next, we enumerate these seven effects and their moderators.

SRAs positively - strongly positively improve the *interoperability* of the software systems (74% belief). Studies reported that SRAs: aim at *"interoperability to improve compliance for a given context"* [S1]; *"act as communication center for information exchange"* [S5]; and integrate software into (and become part of) an SRA [S4]. As we can see in the last example, *existing software* in the organization proportionally moderates interoperability.

SRAs positively - strongly positively impact the *development costs* of software projects (67% belief). *Reuse* of common assets proportionally moderates development costs from not having to start from scratch [S1]-[S2].

SRAs positively improve the *communication* inside their acquisition organizations (65% belief). SRA stakeholders share the same architectural mindset, fostering an improved communication, i.e., *"people talk the same language"* [S5]. *Organizational thinking* proportionally moderates such communication: *"when a service-based SRA is implemented, different departments within an organization need to a) share information with other departments, but also b) get things from other department"* [S4]. Also, the role of an SRA as a *knowledge repository* proportionally moderates knowledge transfer and communication. To sum up, an SRA aids the understanding of architectural and design principles [S1].

SRAs positively - strongly positively influence the *risk* of software projects (65% belief). The *maturity* of an SRA proportionally moderates its risk. Maturity relates to the degree of formality and optimization of processes, from *ad-hoc* practices, to formally defined steps, to managed result metrics, to active optimization of the processes, e.g., *"a mature architecture follows principles for 'good' design, such as high cohesion, high modularity and low coupling"* [S4]. Risk reduction is achieved through the use of proven and partly prequalified architectural elements. The general maturity and experience level associated with an SRA also bears the promise of a higher quality end-product [S1]. If no mature architecture exists, designing/introducing an SRA is likely to fail [S4].

SRAs positively improve the use of *best practices* inside their acquisition organizations (64% belief). The studies do not report the type of best practices. This is proportionally moderated by the *maturity* of an SRA.

SRAs negatively - weakly negatively influence the *learning curve* of developers (60% belief). Developers that use an SRA need to learn its features [S2]. As a consequence, "*many engineers have difficulty learning*" some SRAs [S5]. *Organizational thinking* indirectly proportionally moderates the learning curve: "*changing organizational thinking in employees is often achieved through training that takes place when introducing SRAs*" [S4].

SRAs positively - strongly positively impact the *development time* of software projects (52% belief). This benefit is also proportionally moderated by *reuse*, which can lead to shorter development cycles. However, it is not the same effect as development costs, because it refers to lower time-to-market of the constructed software [S1]-[S2].

8.3.2 Effects of SRAs that Slightly Increased their Belief

Six effects have slightly increased their belief.

Three of these effects have been reported by only two studies: reduced *maintenance costs* of software projects, improved *productivity* of developers, and *alignment* of applications to an organization's business needs. Since the studies agree on them, these effects have increased their value, but more research is needed to corroborate them.

However, the other three effects were reported by at least three studies. In the case of *ease of developing* and *standardization*, we can see in Table 8.2 that these effects are stronger for some types of SRAs (see Section 8.4). In the case of regulative SRAs that *restrict* the development on software systems, the percentage is low because the three studies reporting it gave a really low confidence value, thus, it seems that it is not seen as a very important drawback for practitioners.

8.3.3 Effects of SRAs that did not Change their Belief

Seven effects did not change their belief. Three of them, *dependency* of the software systems over the SRA, propagation of bad *software quality* and wrong decisions of the SRA, and *novel design solutions* are reported by two studies, with different degrees of effect intensity, which did not contribute to increase the evidence level during the aggregation. In fact, the two latter effects have a negligible conflict level of 0.002. We can conclude that the confidence level on these three effects is very low, so they rarely appear in practice, and it seems that they are not considered as fundamental benefits or drawbacks.

The use of *latest technologies*, up-front and migration SRA *investment*, *reliability* of SRA artifacts used in the software systems, and *reputation* of acquisition organizations have been reported by only one study. It does not mean that these effects caused by SRAs are not important, but more investigation effort by the research community is needed to understand them. Still, some of the effects, as reputation, may depend on the SRA type, e.g., an SRA for a market domain so that other companies may be interested in outsourcing [S5].

8.3.4 Effects of SRAs that Decreased their Belief

Three effects have lower confidence level after the aggregation due to contradictory evidence in the single studies. These effects are: a) the *complexity* of the software construction process due to using an SRA; b) how an SRA affects the establishment of *terminology conventions*; and c) how an SRA influences the *flexibility of suppliers* or outsourcing companies that develop software systems based on the SRA. We further discuss the reasons why these effects have decreased their belief in Section 8.4.

8.4 Discussions

In this section, we respectively discuss the effects that were present in different SRA contexts, contradictory results, and the utility of the aggregation with respect to SRAs theory.

8.4.1 Effects of SRAs Present in Different Contexts

The context varies among different studies (see Table 8.1). Still, we have seen common SRA effects reported in different contexts and application domains. This is the case of improved *interoperability*, reduced *development costs*, better *communication*, and higher *learning curve*, which have been reported in four out of five studies without contradictions. These SRA effects, described in Section 8.3, are the strongest results of the aggregation.

8.4.2 Contradictory SRA Effects from Different Studies

In all studies, the context was the use of SRAs for the design and construction of software systems. However, these SRAs were of different type, for instance, they had different goals (e.g., standardization and facilitation), targeted several domains (e.g., automotive software and e-government), involved different stakeholders (e.g., vendors and client organizations), and coexisted with different software and constraints (e.g., reference models). Due to their different contexts, there are some effects that are caused by some types of SRAs, but are not present in other types of SRAs. Still, even with those differences, we understood that it was possible to generalize the concept of SRA for software systems, independently from their types, in order to analyze its most prominent effects, and then, examine the conflicting results from the perspective of their contextual differences.

Next, we discuss those effects that have contradictory evidence and form hypothesis to contextualize them.

Are SRAs *complex* or do they *ease the development* of software systems? The study S5 reported that AUTOSAR influences negatively - strongly negatively the *complexity* of the software construction process with 27% belief. However, the study S2 showed that nine SRAs for information systems weakly negatively influence this complexity with 6% belief. Therefore, we can see different SRAs that differently affect to the complexity of software development. For these two studies, related effects to complexity had different effect intensity and confidence level. For instance, AUTOSAR have worse results (i.e., lower intensity and confidence level) for the effects *ease of developing* and *productivity* that the other SRAs.

The reasons that we posit for this conflict is that AUTOSAR has the goal of standardization of concrete architectures (aiming at system/component interoperability), whereas the SRAs of the study S2 focus more on facilitation of the design of concrete architectures (aiming at providing guidelines and inspiration for the design of systems). Also, the study of AUTOSAR mentioned two moderators of this effect: a) the size of the concrete architecture project, e.g., large projects with many developers and highly interconnected functionality is where using AUTOSAR becomes very tough; b) the existence of a tool environment, e.g., tools that help developers while using an SRA. The first moderator, the size of the project, can be present in both contexts (i.e., standardization and facilitation SRAs). However, we can see that facilitation SRAs tend to include more guidelines and a tool environment to facilitate the development of applications, e.g., user manuals, tool prescriptions and plugins, and sample instantiations [38].

Hypothesis: Complexity depends on the type/goal of the SRA (i.e., standardization and facilitation) and on the guidelines that it delivers to facilitate the development. SRAs that aim to standardize tend to be more complex that those that aim to facilitate software systems construction. Providing a tool environment seems to reduce the complexity for both types of SRAs.

Is it always positive to establish *terminology conventions*? The study S1 claimed that *"an SRA can serve as a framework and lexicon of terms and naming conventions"* whereas the study S5 stated that *"AUTOSAR practitioners face problems with term confusion"*.

One reason we posit for this conflict is that although an SRA could aim to establish term conventions, they do not always reach this benefit. For the case of AUTOSAR [S5], documentation is large (about 20,000 pages), what could discourage users to completely read these lexicons of terms.

Hypothesis: Although an SRA can define a common lexicon of vocabulary, the success of establishing term conventions depends on the design and documentation size. If documentation is not 'digestible', it may lead to terms confusion when stakeholders are not familiar with those terms.

Do SRAs allow *flexibility of suppliers*? In the context of organizations that outsource suppliers to develop their software systems, flexibility of suppliers refers to the capability of an organization to change these suppliers. With the effect *"flexibility of suppliers"* there was also a conflict during the aggregation. In the study S1, the authors state: *"An acquisition program backed up by a strong SRA that ensures interoperability and 'form, fit, and function' compatibility promotes flexibility in the choice of suppliers, as well as a lower risk through multi-sourcing"*. However, vendor lock-in moderator of study [S4] shows that *"customers are restricted in changing their system without the involvement of the vendor, despite the use of open standards. Customers try to reduce vendor lock-in, but this is not always possible, given the small market of software vendors in certain domains and the required expertise"*. Therefore, SRA adoption does not guarantee flexibility of suppliers, which also depends on other approaches such as the use of open source software.

Hypothesis: Despite the use of open standards, mature architectures, and the construction of knowledge repository, outsourcing the construction of SRA-based software may imply vendor lock-in for organizations, jeopardizing the flexibility of suppliers.

8.4.3 Contribution of this Aggregation to the Theory on SRAs

Previous studies reported the effects (i.e., benefits and drawbacks) caused by the use of SRAs, as well as the percentage in which they appeared in practice [22, 37, 40]. Other works have focused on analyzing the practices and constraints of SRAs, and qualitatively reported how they moderate or imply the aforementioned effects [3, 55]. By aggregating the results, this is the first study considering the percentages given in previous quantitative studies, and explaining how specific characteristics of SRAs moderate their own effects.

This work contributes to the body of knowledge of SRAs bringing stronger evidence of their benefits and drawbacks. For most of the effects, the results followed the trends of previous research. However, for three effects these results were contradictory. This observation helped to see that some effects are not general to all types of SRAs, but rather to specific types and contexts. We believe that the aggregated results points to more generalized perceptions and stronger indications of its applicability. Thus, it is expected that practitioners benefit from these indications to support the decision making in practice. Moreover, the stated hypotheses or even the aggregated results themselves can be target of further studies in the future.

8.5 Validity

The risks of aggregating diverse evidence (i.e., qualitative and/or quantitative) from different studies have been mitigated by using the SSM method. The SSM method aims to support the SE community to construct and consolidate empirically-grounded knowledge [130]. This section discusses possible threats to validity and emphasizes the mitigation actions used.

To mitigate the threat of missing important primary studies, we systematically searched empirical studies about the benefits and drawbacks of SRAs. We obtained a set of five studies reporting evidence on real SRAs, which is a high number in SE considering that they report the same effects (i.e., benefits and drawbacks of SRAs). During this process, we discarded studies that only reported opinions, rather than empirically-grounded evidence. Even though we found five studies, more studies are needed to reach definitive results.

We are aware that each selected study poses its own validity threats; therefore, we carefully assessed them together with the studies' context to properly interpret their results. Furthermore, while representing empirical evidence from individual studies, researchers can reflect their own opinion and, therefore, bias the representation. To mitigate these subjective issues, the definition and analysis of each individual evidence model from each selected study was first done by a researcher and validated by another one. The studies S2 and S5, conducted by the author of this PhD thesis, were modeled by other researcher to avoid bias and not to include "extra" knowledge that was not reported in the papers. Then, the aggregated models were assessed and discussed by the whole team of researchers. During this process, we experienced some semantic issues, meaning that different studies referred to the same concept using different terms. This would lead to a wrong aggregation. To avoid this, we created a glossary of terms that were represented in the evidence models and kept track of the matching terms.

To improve the interpretation of the aggregated evidence, we used some suggested strategies in [130]. For instance, given that the SSM method does not consider the different size of sampling of different studies, whenever possible, we refined the confidence level of each effect applying the discount of participants who did not mention it, as suggested by the SSM [130]. In addition, we recorded the diverse context of each individual study, so we could better reflect and understand the aggregated evidence. It is important to note that our aggregated results are based on what the authors reported in their papers. Hence, there is always the risk that important information might not be reported. Anyway, in case of doubts we considered the option of contacting the authors to clarify any issues, but this was not the case in this study.

Finally, one of the goals of aggregation in SE is to consolidate empirically-grounded knowledge, to increase whenever possible the generalization of the results and the understanding of the contexts that might cause any effect. Given the mixed nature (i.e., qualitative/quantitative) of the assessed studies, the aggregated resulting effects could not be statistically but analytically assessed [138]. For this reason, it was highly relevant to properly define the individual contexts of the studies, so we could better understand and interpret the diverse effects of SRAs. We paid special attention to identify the mechanism that produced the studied effects (i.e., moderators). This helped us to explain how the SRA characteristics influence the acquisition organizations. All these strategies have increased the confidence of our results.

Our results show that some effects got higher degrees of belief while others did not. It is important to be aware of the correct interpretation of these results. On the one hand, the effects that got higher belief are potentially those that have been further studied and agreed among the studies. On the other hand, those effects that got lower belief values (or even negative) are those that were just partially approached by the existing evidence (or got contradictory results among the studies). Therefore, these effects are relevant topics that need to be further studied. We highly encourage the SE community to investigate the effects that do not have a high confidence value yet, in order to increase knowledge and consolidation of the benefits and drawbacks of SRA.

8.6 Summary of the Fourth Cycle of RQ 1

Aggregating evidence of empirical studies helps to increase the confidence with respect to single studies by formulating new theories. Besides, such synthesis reduces the effort of researchers and practitioners that are interested in a particular phenomenon. In order to progress as a discipline, we believe that more aggregation studies are needed in SE. These studies help to join forces while conducting empirical studies, which becomes more important in SE due to the low number of empirical studies.

In this chapter, we have applied the Structured Synthesis Method (SSM) to aggregate existing evidence about the benefits and drawbacks of SRAs from five empirical studies: the case study from the Chapter 5, the survey from Chapter 7, and other three empirical studies from other researchers. The aggregated results point to more generalized perceptions and stronger indications in order to answer the RQ 1 of this PhD thesis.

Part III

The Business Case for Software Reference Architectures

Chapter 9

A Survey to Discover Existing Data in SRA Projects

As we mentioned in the Chapter 1, this Part III details our action research initiative with regard to the RQ 2 of the PhD thesis:

> *Is it worth for an organization to invest on the adoption of an SRA?*

To answer RQ 2, we conducted four tasks (see Table 1.3, and Figure 1.4). Each of these four tasks is a chapter in this Part III.

In the current chapter, we start with the first cycle of RQ 2. We diagnosed the problem of the lack of approaches to justify the investment on SRAs to *everis*' clients in monetary terms. As a consequence, we designed a survey to ask stakeholders the metrics available in SRA projects. In this survey, a sample of 5 *everis*' SRA projects and 5 concrete architecture projects were selected on the basis of their suitability and feasibility to contact at least with one person that participated in the projects[1].

As we have seen in Chapter 5.4, the main perceived economic benefits on the use of SRAs are the cost avoidance in the development and maintenance of systems due to the reuse of software elements, and the adoption of best practices of software development that increase the productivity of developers. To

[1]To see a complete vision of the types of projects at *everis*, the reader is referred to Section 2.4 of Chapter 2.

quantify these cost avoidances, we used online questionnaires to ask project technical managers and application builders about existing information from past projects. When the client organization has no experience in SRAs, these data need to be estimated, which could be potentially error-prone. The questions of this survey were divided in three types of questions: about the data available for the SRA, about data available for the applications based on the SRA, and about adding comments and proposing new metrics to calculate the ROI of the SRA. The questions for software architects to check existing value-driven data in SRA projects are available on Appendix D.1.

9.1 Results: Costs and Benefits Metrics for SRAs

In this section we describe the information that was available in *everis* to calculate the costs and benefits of adopting an SRA. We divide existing information in two categories: effort and software metrics. First, the invested effort from the tracked activities allows the calculation of the project costs. Second, software metrics help to analyze the benefits that can be found in the source code.

Effort metrics to calculate projects' costs. In SRA projects, 4 out of 5 client organizations tracked development efforts, while maintenance effort was tracked in all 5. In concrete architecture projects, 4 out of 5 client organizations tracked development and maintenance effort.

The development effort is the total amount of hours invested in the development of the SRA and the concrete architectures of applications. It could be extracted from the spent time for each development activity of the projects. The maintenance effort is the total amount of hours invested in the maintenance of the SRA and the concrete architectures of applications. Maintenance activities include changes, incidences, support and queries. These metrics can be collected with issue tracking tools, such as JIRA[2] or Redmine[3]. For instance, in *everis*, JIRA [139] was used to collect the invested effort from training, development and maintenance activities. Keeping track of activities is common in practice for project management and auditing.

Software metrics to calculate benefits in reuse and maintainability. Source code in SRA and concrete architecture projects was obviously available in all projects. However, due to confidentiality issues with client organizations, it is not always allowed to access source code.

[2]JIRA, `http://www.atlassian.com/es/software/jira/overview`
[3]Redmine, `http://www.redmine.org/`

The analysis of the code from SRA and concrete architecture projects allow quantifying the size of these projects in terms of Lines Of Code (LOC) or function points (e.g., number of methods). Having calculated the project costs as indicated above, we can calculate the average cost of a LOC or a function point. Since the cost of applications development and maintenance is lower because of reuse, we can calculate the benefits of SRA by estimating the benefits of reusing SRA modules. These metrics can be collected with software for code quality. For instance, in *everis*, Sonar[4] [140] was used to gather software metrics to analyze the benefits that can be found in the source code. Sonar offers tool support for obtaining general software metrics such as LOC, dependencies between modules, technical debt, and percentages of tests and rules compliance.

Poulin defines a model for measuring the benefits of software reuse [87]. Maintenance savings due to modularity could be calculated with DSMs [97]. Chapter 10 explains how such metrics can be used in a cost-benefit analysis.

9.2 Next Steps and Lessons Learned

Improvements in the quality attributes of an SRA (e.g., reuse, maintainability, security) are extremely difficult to evaluate in an analytic and quantitative fashion contrary to the efficacy of the business (e.g., sales) [96]. This is because software development is a naturally low-validity environment and reliable expert intuition can only be acquired in a high-validity environment [141]. In order to evaluate SRAs based on an economics-driven approach, software development needs to move to a high-validity environment. The good news is that it could be done with the help of good practices like time tracking, continuous feedback, test-driven development, and continuous integration. In order to get the metrics defined above, tools such as JIRA [139] and Redmine [142] allow managing the tasks and their invested time, general software metrics (like LOC) and percentages of tests and rules compliance can be calculated by Sonar [140] and Jenkins [143]. We think that adopting good and repeatable practices to collect data is the basis for moving software development to a high-validity environment and consequently being able of performing an accurate cost-benefit analysis.

Due to the aforementioned lack of research in the SRA area, we have aimed at adopting and adapting existing results in related areas, from classical software reuse to product line engineering (see Chapter 3.3.4).

[4]Sonar, `http://www.sonarsource.org/`

9.3 Summary of the First Cycle of RQ 2

With the goal of supporting organizations to analyze which value-driven data exist in SRA projects, we have analyzed several projects at *everis*. We have identified the following common data in SRA projects:

1. Effort metrics to calculate projects' costs.
2. Software metrics to calculate benefits in reuse and maintainability.

Chapter 10

REARM: Calculating the ROI on SRA Adoption

This chapter presents a pragmatic economic model to perform cost-benefit analysis on the adoption of SRAs as a key asset for optimizing architectural decision-making. Therefore, we work further to answer the RQ 2 of this PhD thesis: *Is it worth for an organization to invest on the adoption of an SRA?*

The purpose of our research is to create a method for extracting costs and benefits of SRAs based on data identified in the previous Chapter 9. With this goal in mind, we envisaged REARM, our economic model for SRAs (the acronym REARM stands for software REference ARchitecture Model). As part of the collaboration with *everis*, we had the chance to provide an initial validation of the economic model. Hence, we designed and conducted an retrospective study to analyze one SRA projects from *everis* with the help of REARM. It comprises a retrospective evaluation of an SRA created by *everis* for the IT department of a public administration center in Spain.

This chapter is structured as follows. Section 10.1 presents a method to formulate an economic model for SRAs. Section 10.2 follows this method to create REARM. Then, in Section 10.3, REARM is used in an *everis* SRA project for the IT department of a public administration. Section 10.4 discusses limitations of applying REARM. Finally, Section 10.5 summarizes the contributions of the second action-research cycle of RQ 2.

10.1 A Method for Formulating an Economic Model

An SRA cost-benefit analysis should be based on giving an economic value to its activities. We designed our economic model through the three following steps:

1. **Identifying the costs and benefits stemming from the use of an SRA.** Although cost modeling is already a mature field within SE, benefits have traditionally been far more elusive to quantify [96]. For this reason, it is necessary to identify the SRA quality attributes that bring more benefit to the development and maintenance of applications, and the costs of constructing these applications [93]. These attributes may vary depending on the architecturally-significant requirements coming from the applications based on the SRA. It is crucial to involve relevant stakeholders to ensure the trustworthiness of the collected information [144].

   The outputs of this step are the costs factors of adopting an SRA and the list of quality attributes in which the SRA brings more benefit.

2. **Adopting metrics to quantify the costs and benefits identified in the first step in order to convert them into a monetary value.** The metrics to quantify costs and benefits may vary depending on the data available in the organization involved.

   The output of this step is providing guidelines to collect simple metrics making possible to calculate the cost and benefits factors in practice.

3. **Making the business case for the adoption of the SRA.** Adding the costs and benefits calculated in the second step to the formula for calculating the ROI (see Equation 10.1, and Chapter 3.3.3), where the benefits are the improvements of applications quality attributes, and the costs are the expenses in constructing the systems and the SRA.

   The output of this step is a business case that captures the reasoning for adopting an SRA. The SRA business case analysis involves determining the relative financial costs, benefits, and ROI across its life-cycle.

$$ROI = \frac{Benefits - Costs}{Costs} \tag{10.1}$$

10.2 REARM: the Economic Model for SRAs

The action-research collaboration with *everis* provided us the opportunity of implementing the general-purpose method from previous section in a particular case.

10.2.1 Step 1 of the Method for Formulating an Economic Model

After conducting the case study of Chapter 5.4, we could see that the main perceived economic benefits on the use of SRAs were: (1) an increased value from the improvement of quality attributes, since their reused architectural knowledge is incrementally improved with previous successful experiences from its application domain; (2) cost savings in the development and maintenance of systems due to the reuse of software elements and the adoption of best practices of software development that increase the productivity of developers. Therefore, SRAs bring most of the benefit because of the improvement of reusability and maintainability quality attributes. One of the reasons why SRAs were adopted in these organizations is that the most important architecturally-significant requirement was reusability. Thus, we decided to focus our cost-benefit analysis over reusability and maintainability.

We found that some of the potential metrics to be used were not as pragmatic as the organization needed. In other words, the organization should have been invested extra time which was not an option. Furthermore, we faced the problem that some of the required data to apply the proposed metrics was not previously registered by the organization. Thus, we stressed the emphasis on formulating a practical model that incrementally deals with diverse cost-benefit aspects.

We identified six cost-benefit factors for SRA adoption. We started the formulation of factors by adopting Poulin's method for measuring code reuse [87, 145]. We adapted Poulin's model by offering parameters to operationalize it, and we could feed it with available data in *everis* (see Step 2 below). We adopted its benefit factors (DCA, SCA) published in [145]. Conversely, we consider more appropriate for SRAs to adopt the cost factors defined for software product lines (CSW_{dev_costs}, $CSW_{service\_costs}$) in [145], instead of the additional development costs [87].

To complete the model we add the unique development costs of applications. Also, with the help of the propagation cost metric [97], we also consider necessary changes to reusable elements (which are not considered by Poulin's method) and, therefore, evolution. These two new factors include parameters to operationalize them.

The former three factors are for development and the latter ones for maintenance. When possible, these factors are explained below by comparing them to the factors from SIMPLE (see Chapter 3.3.4):

- DCA (Development Cost Avoidance). It is the *benefit* from reusing SRA's software modules in applications compared to building the applications independently.
- UDC (Unique Development Costs). It is the *cost* to develop the unique parts of an application that are not already implemented in the modules of the SRA. UDC is equivalent to C_{reuse}+C_{unique}.
- CSWD (Common Software Development costs). It is the *cost* of the initial investment, i.e., developing an SRA. CSWD is equivalent to C_{org}+C_{cab}.
- SCA (Service Cost Avoidance). It is the *benefit* of modifying reused code once.
- CSWS (Common Software Service costs). It is the *cost* of fixing bugs in the (reusable) SRA modules. CSWS calculates the cost of changes due to bugs in C_{cabu}.
- CSWE (Common Software Evolution costs). It is the *cost* of changing or adding functionalities to the SRA modules. CSWE calculates the cost of evolutions in C_{cabu}. Therefore, CSWS+CSWE are equivalent to C_{cabu}.

Putting everything together, given a number *n* of applications built in top of the SRA, and a number *m* of SRA modules changed as it evolves, the benefits and costs of adopting an SRA are respectively defined by Equation 10.2 and Equation 10.3:

$$Benefits = \sum_{i=1}^{n}(DCA_i + SCA_i) \tag{10.2}$$

$$Costs = CSWD + CSWS + \sum_{i=1}^{n} UDC_i + \sum_{j=1}^{m} CSWE_j \tag{10.3}$$

10.2.2 Step 2 of the Method for Formulating an Economic Model

We divided the second step in two activities: checking the data available in practice and guiding the information extraction from this data.

Data commonly available in practice that should be collected. The data typically available to calculate the aforementioned costs and benefits are effort and software metrics (see Chapter 9). It allows converting cost-benefit factors into a monetary value.

On the one hand, the invested effort from the tracked activities allows the calculation of costs. We distinguished between three types of activities: training, development and maintenance. JIRA and Redmine are tools that support keeping track of activities and their invested time (see Chapter 9.1). Keeping track of activities is common in practice for project management and auditing. Activity tracking is also known as tickets [96].

On the other hand, software metrics help to analyze the benefits that can be found in the source code. For example, since the cost of applications' development is lower because of the reuse of an SRA, we could estimate the cost avoidance of reusing its LOC. Sonar offers tool support for obtaining general software metrics such as LOC, dependencies between modules, technical debt [100], and percentages of tests and rules compliance (see Chapter 9.1).

We experienced difficulties collecting historical data (as other researchers did in [86]), especially for the "before" state of adopting an SRA. We noted that C_{prod} and C_{evo} were seldom available since the "before" state did not exist. For this reason, we proposed to use RCR and RCWR.

Using commonly available data in practice to quantify the costs and benefits. In Table 10.1, we present ten basic parameters that are required for calculating the six cost-benefit factors of the Step 1. Table 10.2 shows the formulas to calculate these six cost-benefit factors as well as parameters that are needed for these calculations.

10.2.3 Step 3 of the Method for Formulating an Economic Model

As final step, we can use calculated factors in order to calculate the ROI:

$$ROI = \frac{[\sum_{i=1}^{n}(DCA_i + SCA_i)] - [CSWD + CSWS + \sum_{i=1}^{n} UDC_i + \sum_{j=1}^{m} CSWE_j]}{CSWD + CSWS + \sum_{i=1}^{n} UDC_i + \sum_{j=1}^{m} CSWE_j} \tag{10.4}$$

Table 10.1: Basic parameters in order to feed the factors of Table 10.2.

	Description of the parameters (adapted for the SRA context)
RCR	*Relative Cost of Reuse*: effort that it takes to reuse a component without modification versus writing it new one-at-a-time [87]
RCWR	*Relative Cost of Writing for Reuse*: effort that it takes to write a reusable component versus writing it for one-time use only [87]
ER	*Error Rate*: the historical error rate in new software developed by your organization, in errors per thousand lines of code [87]
EC	*Error Cost*: your organization's historical cost to fix errors after releasing new software to the customer, in euros per error [87]
NMSI	*New Module Source Instruction*: the LOC that the changed or new module has, which can be the average of previous ones
PC	*Propagation Cost*: the percentage of code affected in the SRA when performing evolutions (i.e., changing modules) [97]
CPKL	*Cost per KLOC*: the historical cost to develop a KLOC of new software in your organization [87]
USI	*Unique Source Instructions*: the amount of unique software (i.e., not reused) that was written or modified for an application
RSI	*Reused Source Instructions*: it is the total LOC of the SRA's modules that are reused in an application. It supports variability. In other words, reuse of SRA might not be complete but partial, since different applications can reused different SRA's modules. Therefore RSI depend on each application [87].
TSI	*Total Source Instructions*: it is the total LOC of the SRA that can be reused [87].

We suggest using these cost-benefit factors to make a business case for calculating the ROI of building an SRA vs. building the applications independently. Table 10.3 shows an example of business case and how to calculate the cost and benefits for three years since the SRA adoption. The parameters n_1, n_2, n_3 indicate the number of applications developed per year respectively, and m the number of evolved modules.

As Boehm points out [82], two additional factors may be important in business case analysis: unquantifiable benefits, and uncertainties and risk.

First, the economic model that we propose promotes benefits in reusability and maintainability. However, other quality attributes, such as security, could be as relevant as those for this analysis, even when they may be difficult to quantify. These other benefits should also been taken into account when

Table 10.2: Cost-benefit factors to calculate the ROI of adopting an SRA in an organization.

	Description of the cost-benefit factors (adapted for the SRA context)
DCA	*Development Cost Avoidance*: the benefits from reusing SRA's modules [87] DCA = RSI * (1-RCR) * CPKL
CSWD	*Common Software Development Costs*: the costs to develop the SRA [145] CSWD = RCWR * TSI * CPKL
UDC	*Unique Development Costs*: the costs to develop the unique part of an application UDC = USI*CPKL
SCA	*Service Cost Avoidance*: benefits from maintaining only once SRA's modules [87] SCA = RSI * ER * EC
CSWS	*Common Software Maintenance Costs*: cost of fixing bugs in reusable modules [145] CSWS = TSI * ER * EC
CSWE	*Common Software Evolution Costs*: the costs of changing or adding a new functionality and maintaining it to the SRA CSWE = evolution development + evolution maintenance + propagation = (NMSI*RCWR*CPKL)+(NMSI*ER*EC)+(TSI*CPKL*PC)

Table 10.3: Example of design of a business case with the cost-benefit factors of the model.

	Year 1	**Year 2**	**Year 3**
Total benefit	n_1*(DCA+SCA)	n_2*(DCA+SCA)	n_3*(DCA+SCA)
Total cost	CSWD+ n_1*UDC+CSWS*$^1/_5$	n_2*UDC+ CSWS*$^2/_5$+m*CSWE	n_3*UDC+ CSWS*$^2/_5$+m*CSWE

adopting and SRA. Unquantifiable benefits are also considered as "flexibility" in TEI [84], the economic model of Forrester.

Second, to adjust cost and benefits to risk, they can be multiplied by percentages that generally increase the costs and reduce the benefits (assuming the worst case). For instance, TEI proposes to multiple costs by values that range from 98% to 150% and benefits by values between 50% and 110%.

10.3 Preliminary Validation

To assess the feasibility of the economic model, we conducted a retrospective analysis of a particular case. We calculated the costs and benefits (and hence the ROI) of an SRA adoption driven by *everis*. In this analysis, a sample of 1 *everis'* client organization SRA project in an IT department of a public organization and 1 concrete architecture project was selected. These projects were selected because the public administration that adopted the SRA was interested in the study results. Besides, by the time we conducted the study, *everis'* started the aforementioned concrete architecture project, being highly feasible to collect quantitative data. Although we were aware of other concrete architecture projects with participants that do not belong to *everis*, it was not possible to contact with them.

By the time we performed the validation, the public organization had already:

1. adopted an SRA,
2. created an application using the SRA –which we consider "exemplar" application–, and
3. fixed errors discovered in the SRA software elements that were reused by the application.

The validation consisted of 4 parts. First, a post-mortem analysis in which our challenge was to extract the parameters of Table 10.1 from already collected data. The values that we got are shown in Table 10.4.

Recommended values for RCR range from 0,03 and 0,25, and for RCWR from 1 to 2,2 [87]. Therefore, with the values that we got in the study, we can see that both RCR and RCWR are low for SRAs. A low RCR could show the trend of moving the complexity to the architecture in order to simplify the development of applications. We can also see this trend comparing the code of the SRA software elements with the code of applications. SRA code present higher values for complexity metrics such as coupling and cohesion. A reason why RCWR is low could be that SRA architectural knowledge speeds up the development.

Second, with the data of Table 10.4, we had real data to calculate the following 4 (out of 6) cost-benefit factors of REARM (see Table 10.5):

- CSWD, the SRA initial investment, which lasted 6 months.
- DCA, the benefit of reusing SRA code in the exemplar application development.
- SCA, the cost from fixing the errors of the reused code in the exemplar application.
- UDC, the cost of developing the application.

The above costs were accurately computed because *everis* kept track of activities with their invested time. Third, it was necessary to estimate the rest of factors:

- CSWS, the cost of fixing all bugs in SRA code. Since we knew the SCA for the exemplar application and the percentage of reuse, we calculated the error rate and error cost, which we used to estimate CSWS.

Table 10.4: Values of the basic parameters in the study

Parameter	**Value**
RCR	0,064
RCWR	1,243
ER	2,879 err./kLOC
EC	7,02 hours/err.
NMSI	1.526 LOC/module
PC	9,7 %
CPKL	75,22 hours/kLOC
USI	2.885 LOC
RSI	8.364 LOC
TSI	41.189 LOC*

**Note*: In TSI, 9.231 LOC were refactored from previous project. So, 31.958 were new.

Table 10.5: Values of the cost-benefit factors in the study*.

DCA	CSWD	UDC	SCA	CSWS	CSWE
589 hours	**2.988 hours**	**217 hours**	**169 hours**	*≈832 hours*	*≈474 hours*

**Note*: The symbol '≈' indicates estimated values. The other values are real data.

- CSWE, the cost of: (1) changing or developing a module with new functionality, (2) fixing its bugs, (3) making changes in the rest of the SRA to integrate it.

Fourth, we made the business case analysis with two different scenarios.

10.3.1 Scenario 1: Is it Worth to Invest on the Adoption of an SRA?

We constructed a business case for 3 years starting when the organization decided to adopt the SRA, in order to calculate the ROI. For the first 8 months of those 3 years, we had real data about the SRA development and the exemplar SRA-based application. To estimate the costs and benefits for the rest of these 3 years, we conducted some additional interviews to the involved stakeholders. Stakeholders were carefully selected according to their knowledge and experience to increase the degree of confidence on the data gathered. After these interviews, we made the following assumptions:

- Future applications will have similar characteristics and complexity as the exemplar one.
- The public organization will develop 8 applications per year. Since the SRA creation lasted 6 months, the first year they will develop just 4 applications.
- The totality of CSWS is computed proportionally starting the seventh month.
- A module is evolved (with new functionality) or added to the SRA every year since the second year.

Under these assumptions, the costs and benefits in hours for the future can be calculated as shown in Table 10.3. They can be converted into a monetary value by multiplying them by an hourly rate. Assuming a rate of €30 per hour for an application developer (which affects to DCA, SCA, UDC) and a rate of €40 per hour for a developer and maintainer of the architecture (which affects to CSWD, CSWM, CSWE), Figure 10.1 summarizes financial results for first three years of the SRA. This organization will realize a ROI within 2 years through gains in systematic reuse.

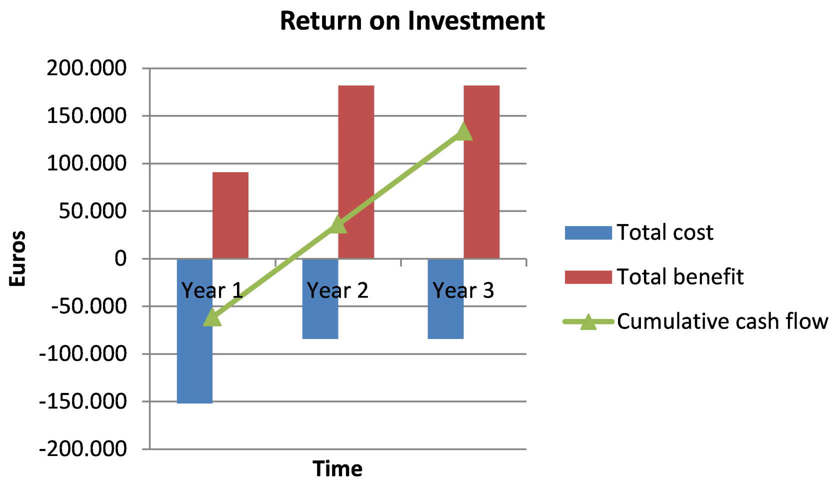

Figure 10.1: Summary financial results.

10.3.2 Scenario 2: How Many Instantiations are Necessary before Savings Pay Off for the Up-front Investment?

In this scenario we calculated how many applications need to be build based on the SRA to have a positive ROI. Figure 10.2 shows the ROI due to developing and maintaining applications based on an SRA rather than in a stand-alone fashion.

As Figure 10.2 shows, after building 7 applications, savings pay off for the up-front investment in the SRA. It must be noted that the exemplar application is small and only 20% of the SRA is being reused (RSI/TSI). On the other hand, the application has a high reuse percentage of 74% (RSI/USI+RSI). The higher these percentages are (likely in medium to large applications), the greater the benefit from the SRA is.

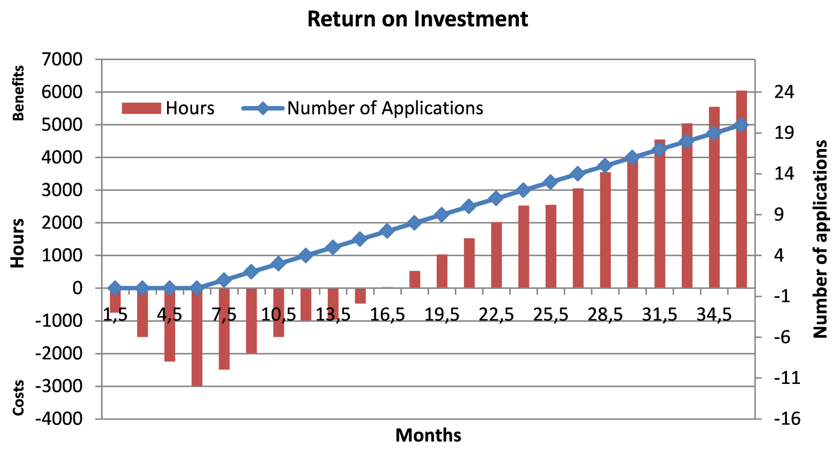

Figure 10.2: ROI of developing and maintaining SRA-based applications vs. stand-alone fashion.

Moreover, applications are introduced into the market earlier from the seventh month on. This is due to the effort avoidance of 589 hours (DCA) of reusing the SRA.

To sum up, this study illustrates the potential way in which an organization can evaluate the value of SRA adoption. We calculated a three-year ROI of 42% with a payback period of 16,5 months and 7 applications.

10.4 Discussion

Once we applied the economic model and calculated the ROI, a last question remains: *How accurate are these calculations and the obtained quantitative data?*

If REARM is applied with existing data (as we have done in Section 10.3), the calculation of the ROI reaches a high degree of correctness, since the data that feeds the model is trustworthy. The metrics coming from code analysis (e.g., size in LOC) do not reflect any error. Also, we saw that time tracking is reliable. During data collection we found invested time in activities in two different sources: JIRA, which is optionally used by the project team and keeps the invested time of the project's activities; and a mandatory corporate financial tool, which is used by the financial department. This data differ in

8,75%, being lower internally time tracking of the project. The reason could be that JIRA does not include other activities out of the scope of the project like traveling. To adjust the calculations to this risk, we have always considered the worst case (i.e., greater costs).

Contrary, when the economic model is used to predict the ROI of a completely new SRA adoption in an organization, there is not real data since it does not exist yet. In this case, the accuracy totally depends on expert intuition and historical data. Historical data can be scarce in small and medium organizations; especially considering that reuse of architectures is still a research area in progress. In addition, historical data must be continuously updated, since some values of effort-related parameters (such as RCR) are expected to decrease each time a developer instantiates the SRA.

As a final remark, the construction of an economic model from the data available in software companies is yet-another-instance of research question which needs to balance soundness with applicability.

10.5 Summary of the Second Cycle of RQ 2

With the goal of supporting organizations to analyze whether it is worth to adopt an SRA, this chapter presents REARM. REARM is an economic model to translate measured or estimated data (i.e., metrics) into monetary terms (i.e., cost-benefit analysis). REARM provides the following artifacts to build the business case on SRAs:

- 10 basic parameters from which we can calculate the cost-benefit factors (see Table 10.1).
- 6 cost-benefit factors to calculate the ROI of adopting an SRA in an organization (see Table 10.2).
- The formula to compute the ROI of SRA adoption (see Equation 10.4).

Besides, we have conducted a preliminary validation to calculate the ROI of adopting an SRA in a real organization. This organization will realize a return on their investment within two years through gains in systematic reuse and applications maintainability.

Chapter 11

Guidelines for Building a Business Case for SRAs

In the two previous chapters, which represent the two first cycles of the action research with *everis* regarding RQ 2, we have gathered experience, feedback and lessons learned. In this chapter, at the end of the formative stage, our goal is to package such experience, feedback and lessons learned into guidelines for building the business case for SRAs.

It is important to note that these guidelines are aimed to be useful not just for *everis*, but also other organizations with a similar context. In order to analyze whether other organizations deal with similar problems as *everis*, we highlighted the similarities of SRAs designed by *everis* with other SRA contexts that were reported in the literature and by practitioners. This analysis was presented in Chapter 6.1, before the construction of the guidelines for RQ 1. Bearing in mind the common aspects of many SRA contexts, we were able to only package into the guidelines the material that could be used under the context of organizations described in Chapter 6.1.

This chapter is organized as follows. Section 11.1 briefly discusses the formative cycles of the action research with respect to RQ 2. Section 11.2 packages the results and provide guidelines that help to answer RQ 2. Finally, Section 11.3 briefly introduces how these guidelines for RQ 2 were validated.

11.1 Formative Stage: Evolution of the Guidelines

The process of packaging the guidelines have been done incrementally from the feedback of the formative stage. The guidelines have mainly evolved to provide the results in a way that is more understandable for the upper management of organizations. Besides, threats to validity and reliability of cost-benefit factors have incrementally been taken into account. Different previous versions of the guidelines can be seen in [39, 26, 29, 30, 31].

As an example of such evolution, Figure 11.1 shows the guidelines as presented at the "X Workshop Latinoamericano de Ingeniería en Software Experimental" [26]. We can compare this previous version of the guidelines to the current one, summarized in Figure 11.2. The main improvement has been that we realized that the economic model should consider, besides cost-benefit factors, variables of the business case. Examples of this variables are the number and size of the applications that will be developed, and how often changes to the SRA would be needed.

11.2 Packaging the Guidelines

everis' results were suitably packaged with the aim of being applied in other SRA projects and also in similar organizations.

First of all, organizations that may want to use these guidelines need to

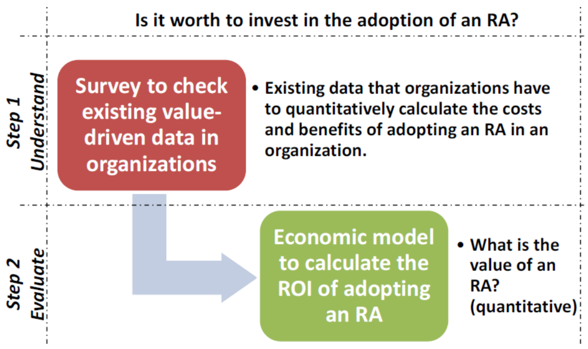
Is it worth to invest in the adoption of an RA?
Step 1
Understand
Survey to check existing value-driven data in organizations
• Existing data that organizations have to quantitatively calculate the costs and benefits of adopting an RA in an organization.
Step 2
Evaluate
Economic model to calculate the ROI of adopting an RA
• What is the value of an RA? (quantitative)

Figure 11.1: Previous version of the guidelines to build the business case for SRAs in industry [26].

fit into the context depicted in Chapter 6.1. This means that they need to design an SRA based on practical experience, and to use such SRA for the development and maintenance of a family of applications in industry. This is because the input for using the guidelines is evidence from real SRA projects.

The guidelines for RQ 2 support organizations to build the business case for SRA adoption based on corporate evidence by providing:

- A checklist to analyze existing value-driven data in SRA projects (results of Chapter 9), i.e., a checklist of value-driven data that an organization might have is facilitated to check if the REARM economic model provided can be executed.

- An economic model that uses such value-driven data to calculate the ROI of adopting an SRA (results of Chapter 10). Then, software architects can feed the economic model to build the business case on SRAs.

Figure 11.2 summarizes the guidelines for building a business case for SRAs in industry in order to analyze whether it is worth to invest on an SRA. The guidelines are composed of the context of SRAs in industry, and materials to conduct two empirical studies. For these two empirical studies, the guidelines recommend: a survey to check existing value-driven data of SRAs in organizations (using the checklist of the guidelines, see Section 11.2.1),

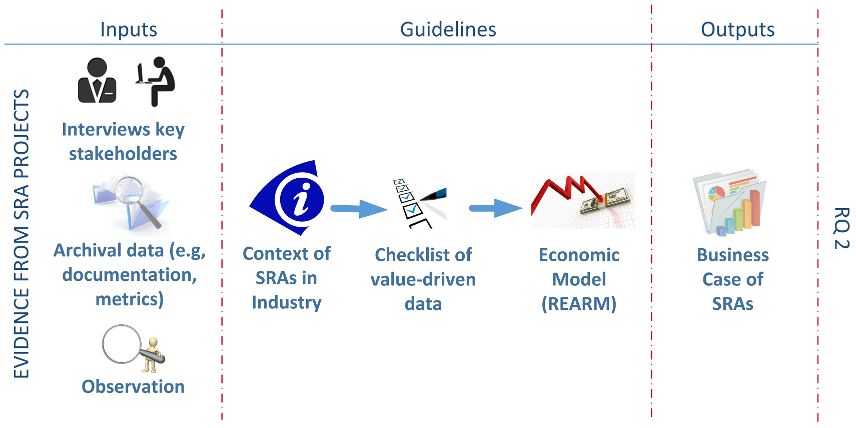
Inputs
Guidelines
Outputs
EVIDENCE FROM SRA PROJECTS
Interviews key stakeholders
Archival data (e.g, documentation, metrics)
Observation
Context of SRAs in Industry
Checklist of value-driven data
Economic Model (REARM)
Business Case of SRAs
RQ 2

Figure 11.2: Guidelines to build the business case for SRAs in industry.

and a case study to calculate the ROI of adopting an SRA (using REARM, see Section 11.2.2).

11.2.1 A Survey to Check Existing Value-driven Data in SRA and Concrete Architecture Projects

Below, we explain the context, objective, method, support material, and the output in this empirical study of the guidelines.

- **Context:** Typically, organizations do not have resources to compare the real cost of creating applications with and without an SRA. Besides, historical data may be scarce. Thus, alternatives should be considered.
- **Objective:** The objective of this survey is to identify the quantitative information that can commonly be retrieved in SRA projects in order to quantitatively calculate the costs and benefits of adopting an SRA in an organization. This is an initial step to create repeatable techniques for performing a cost-benefit analysis.
- **Method:** Exploratory surveys with personalized questionnaires applied to relevant stakeholders (e.g., manager, architect, developer) to find out the quantitative data that has been collected in SRA projects and concrete architecture projects. An example of conducting this empirical study and its approach for data collection is described in Chapter 9.
- **Support material:** Appendix D provides the template surveys to check existing value-driven data in SRA and concrete architecture projects. An example of use can be seen in Chapter 9.
- **Output:** Identification of existing value-driven data in SRA and concrete architecture projects. If the organization has such data, REARM can be applied without estimations.

11.2.2 A Case Study to Apply REARM to Calculate the ROI of Adopting an SRA

Below, we explain the context, objective, method, support material, and the output in this empirical study of the guidelines.

- **Context:** Before deciding to launch an SRA, organizations need to analyze whether to undertake or not the investment. Offering organizations an economic model that is based on former SRA projects data can help them to make more informed decisions.

- **Objective:** The objective is to analyze whether it is worth investing on an SRA with the help of an economic model, in order to improve the communication among architects and management, and to improve their decisions.
- **Method:** A case study that applies an economic model to calculate the ROI of adopting an SRA. Depending on the maturity of the organization, two approaches can be applied. If the organization does not have experience with an SRA, the economic model should be fed with estimated data. Nevertheless, when the organization already has experience with SRAs (i.e., the case of IT consulting firms), real data can be gathered by means of an exploratory quantitative post-mortem analysis. Then, the economic model quantifies the potential advantages and limitations of using an SRA. Some related works explain how to calculate the ROI of a product [84], software reuse [91][87], and software product lines [83]. We suggest the use of REARM, following the example of conducting this empirical study and its approach for data collection as it is described in Chapter 10.
- **Support material:** Appendix D provides the five steps of the REARM economic model. An example of use can be seen in Chapter 10.
- **Output:** A business case to evaluate whether it is worth or not to invest on an SRA.

11.3 Summative Stage: Validating the Guidelines

Once the guidelines were adequately shaped and improved, the summative stage took place. The primary role of this stage was to obtain feedback to check the utility of the guidelines and to validate them with more practitioners. This evaluation was performed by presenting the results of applying our guidelines to different experts. As part of this validation and summative stage of these guidelines, the next chapter presents the feedback that we got from three different meetings with experts (see Chapter 12).

Organizations analyzing whether to make the strategic move to SRA adoption and planning the adoption of SRAs based on evidence will benefit from these guidelines.

Chapter 12

Workshops to Evaluate the Business Case for SRAs

Once that we conducted the survey to check existing value-driven data in SRA projects, and a case study to apply REARM to calculate the ROI of adopting an SRA at *everis*' client organizations in Chapters 9 and 10, it was necessary to evaluate these results and to analyze lessons learned. This task is summative, since its primary role is to evaluate the guidelines for building the business case for SRAs, and identify further areas of improvement. To evaluate such guidelines and their materials (e.g., REARM), we presented the results to several stakeholders:

- The upper management of the *everis*' client organization in which we conducted the study of Chapter 10.

- The audience of the International Conference on Software Reuse 2013 (ICSR 2013).

- The Experimental Software Engineering group (ESE) of the Federal University of Rio de Janeiro (UFRJ).

Next, we respectively present the feedback that we got, and our conclusions from such feedback.

12.1 REARM Validation at an *everis* Client Organization

After applying REARM for an SRA project of an *everis*' client organization (see Chapter 10), we sent the results to the upper management of the organization. They gave us a positive feedback and the following suggestions for improvement:

- They considered more useful to give the outputs of applying REARM in terms of effort (i.e., hours) rather than in monetary terms (i.e., €) for two reasons. First, because REARM is a reused-based economic model, and it considers costs from developing and maintaining software. Therefore, REARM does not consider other costs such as training courses to application builders. As a consequence, the initial costs of adopting the SRA was a subset of the real costs of adopting the SRA in the organization. Second, they considered more appropriate to show the benefits from reusing the SRA elements in percentages. The reason is that avoiding development and maintenance costs in applications through reuse is expressed and understood better by percentages.
- They suggested that REARM should be useful to compare the scenario in which an SRA is used versus the scenario in which an SRA is not used. In this direction, we tailored REARM for such scenario, instead of comparing the costs and benefits from using different versions or releases of the same SRA.
- They also indicated that applications usually reuse an SRA in different percentages. Therefore, some applications benefit more than others by reusing more modules from the SRA than other applications. For this reason, they recommended to apply REARM for three kind of applications: low level of reuse (up to 33% of the SRA), medium level of reuse (from 33% to 66% of the SRA); and high level of reuse (from 66% up to 100% of the SRA).

These three reasons helped us to improve initial versions of REARM.

12.2 REARM Validation at the ICSR 2013

We presented the results of the Chapter 10 in a full paper at the International Conference on Software Reuse 2013 [5]. RQ 2 was considered by a reviewer as "a very important topic in the field. It is also of extreme importance within industry; i.e., my company demands justification in our investments into SRAs, which are difficult to quantify". About REARM, its major novelty was highlighted as "it considers costs and benefits beyond those typical in a straight code reuse situation or a software product line situation. These costs and benefits include the need to continually mature the SRA to reflect new knowledge".

Besides the positive reviews, we got many questions and feedback in the conference, which are summarized below:

- It was remarked that REARM can only be applied to SRAs that have been mapped to software elements already implemented. The reason is that it uses metrics from code reuse (e.g., LOC).

- About the type of metrics, it was remarked than other activities besides development and maintenance avoidance from reuse can be added to REARM. Some examples of these activities are training programs, organizational changes, and SRA engineering and architecting activities (such as requirements engineering and testing).

- About the concrete architecture project in which REARM was used, it was highlighted that it reused the SRA in a lower extent (i.e., 20%). For this reason, two ideas were proposed as future work. First, applying REARM in many concrete architecture projects (e.g., 15-20) with different sizes and levels of reuse. Second, since this may not be possible in many organizations due to lack of information, it could be possible to apply sensitivity analysis to study how the uncertainty in the output of REARM can be apportioned to different levels of reuse from applications.

- About the results presented, the audience commented that the ROI was very low (with 7 applications). Normally, the ROI from complete software reuse is positive in three applications. Obviously, the ROI varies because SRAs are not reused 100% in every application. Still, REARM should be used in projects from private organizations to see if the reason for such low ROI is because it was used in a public administration whose SRA did not have a regulative role and could not reach high levels of reuse due to heterogeneity.

- REARM includes some parameters from Poulin economic model. In our study, we could adapt those parameters (e.g., RCR and RCWR) because *everis* had historical data. They highlighted the importance of adjusting these parameters accurately, and the contribution of the paper by providing RCR and RCWR parameters unique for SRAs.

We think that many of the aforementioned points should be addressed in future work.

12.3 REARM Validation at another Research Group

Besides the ICSR 2013, we also presented the results of the Chapter 10 at a meeting with all the researchers from the Experimental Software Engineering Group of Federal University of Rio de Janeiro. We got interesting feedback in the meeting, which is summarized below:

- They highlighted that REARM should consider differently "effort" and "costs". Although related, they are not the same term. Then, REARM should give the results in effort terms, which can be translated to costs terms.
- Other point is that it is difficult to demonstrate the quality of an SRA in quantitative terms. They remarked that is also very important to show it in qualitative terms, by making hypothesis and testing them in interviews. In this direction, the additional factors at the end of the Step 3 of REARM (see Chapter 10.1), and the qualitative studies from RQ 1 about the benefits and drawbacks of SRAs are very useful.
- The remarked that the "error" concept of REARM could be broader. Therefore, there could be different types of errors and more defect metrics besides the *error rate* (ER) and *error cost* (EC).
- A very important issue is that all the cost-benefit factors should represent an effect caused by SRAs. In other words, those costs and benefits should be a consequence of using SRAs. For each cost-benefit factor, it should always be explained why the factors measure what they are intended to measure, and how they are related to SRAs (see Chapter 10.1).
- There is always the risk that REARM is used wrong. For instance, if you reuse an SRA module, the parameter *Reuse Source Instructions* (RSI) would be considered, changing the output of REARM. However, we have to ensure that such SRA module is indeed being reused (e.g., it may not be enough if a stakeholder claims it).

- In Chapter 10, we considered two scenarios. They recommended that scenarios should be useful for practitioners. Therefore, other scenarios may be added. These scenarios should be evaluated by software architects, in terms of congruency (e.g., does it make sense?) and usefulness (e.g., what can I gain for this?).

- A last issue is the importance of trying to generalize, so that we can give an answer to readers that may wonder what they can win with SRAs. In this sense, more effort is necessary not only in *everis*, but other organizations to get a representative sample.

12.4 Summary of the Third Cycle of RQ 2

With the goal of validating the guidelines for building the business case for SRAs, this chapter describes how such guidelines were presented two three different audiences. Such audiences gave feedback and further areas of improvement of the support materials of the guidelines (e.g., REARM), which have been summarized in this chapter.

Part IV

Conclusions and Future Work

Chapter 13

Discussion: Evaluating our Collaboration

More and more, SE researchers are motivated to solve real problems that bring value to industry. An example is the industry-academia collaboration described in this PhD thesis.

The goal of this chapter is twofold:

1. to evaluate the success of the collaboration, and
2. to report the experience with conducting empirical studies in *everis* and lessons learnt.

First, we evaluated our collaboration with an existing model for technology transfer [146]. Second, we organized a focus group discussion to identify challenges we have faced. Both tasks were done in January 2014, before the end of the collaboration. Therefore, evaluating the collaboration was positive, since we could identified the steps to be taken to achieve a high degree of technology transfer and innovation dissemination. In summary, we think that this type of evaluations are a needed step in the conduction of any industry-academia collaboration in order to improve its success.

We intentionally involved one person from each view (i.e., industry and academia) to reduce the bias of the report and to be as objective as possible. Still, we are aware of the self-report threat.

The chapter is structured similarly to other experience reports on industry-academia [147]. Section 13.1 describes a background of models for technology transfer. Section 13.2 reports the activities that were performed since the beginning of the collaboration to this evaluation, and Section 13.3 evaluates maturity of the collaboration with respect to these research activities and the research results. Section 13.4 presents the lessons learnt that we identified in a jointly focus group among *everis* and GESSI. Finally, Section 13.5 concludes the chapter and present several improvements to be performed.

13.1 Models for Technology Transfer

ESE serves as support for transferring innovation [148]. The conduction of empirical studies is thus increasingly gaining attention to fulfil industry-relevant issues, as several recent experience reports show [149, 150]. Another example is the collaboration of this PhD thesis, which relies on empirical studies to provide a solution to the current challenges that *everis* faces in SRA projects.

In order to improve the body of knowledge on conducting empirical studies in industry, models for technology transfer have arisen, such as [146, 151]. These models provide guidelines to conduct industry-academia research and evaluate it. On the one hand, Gorschek et al. [151] present seven sequential steps that they consider relevant and interdependent for overall transfer success (see Figure 13.1). On the other hand, Sandberg et al. define ten factors for successful projects [146]. Figure 13.2 shows the success factors in which the project depended on and the effects they had on the collaboration. In next two sections, we explain further and use these two models ([151] and [146]), in order to report and evaluate our collaboration. Both models are descriptive, i.e., they derive from experiences on performing industry-relevant research.

13.2 The Collaboration

In this section, we focus on the industry-academia collaboration between *everis* and GESSI. Although we did not follow the full Gorschek et al.'s process [151], we applied the first steps.

In order to show how the collaboration has been conducted and to analyze Gorschek's steps (see Figure 13.1) that need to be taken in the future, we report step by step our research since the beginning of the collaboration (May 2011) until the moment of performing this evaluation (January 2014).

Step 1: Identify potential improvement areas based on industry needs. When the collaboration was signed, the goal in the research area between *everis* and GESSI was: "to boost applied technological research related to SE

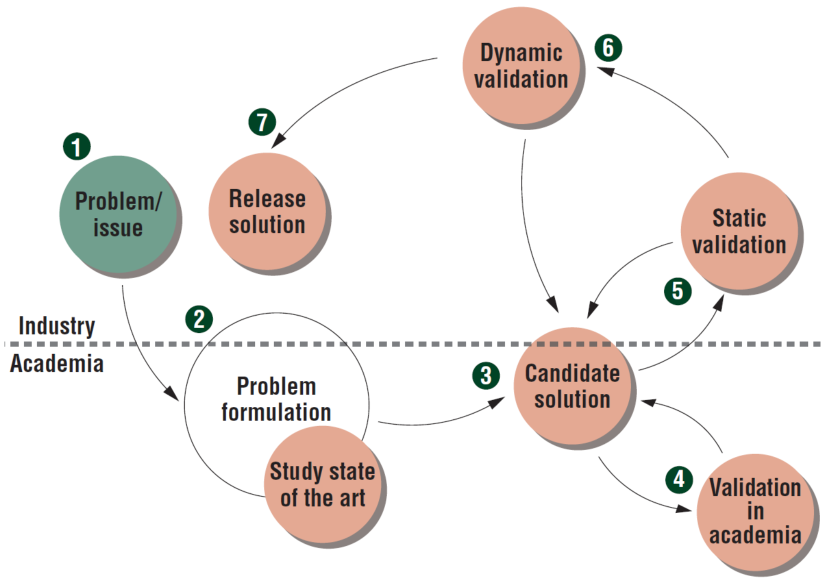

Figure 13.1: An activity model for technology transfer in industry-academia collaboration, from Gorschek et al. [151].

in areas that will be identified as priorities within the sector". This goal was too broad and the so-called priorities needed to be identified.

To identify potential improvement areas, several joint meetings among *everis* and GESSI members were held between May and December 2011. During these eight months, the following activities were conducted. First, the collaboration team was created with two practitioners from *everis* and four researchers from GESSI. Second, the way of working was defined. Regular meetings were celebrated every two weeks, minutes of meetings were written down, and a collaborative environment was set up to make collaborative work and exchange ideas, thoughts, and material. Third, presentations lead by *everis*' employees were given to explain the current state of real projects and their challenges; similarly, presentations lead by GESSI were given to show our current research in SE. After these presentations, we discussed potential improvement areas in *everis*' projects and how the research conducted at GESSI could be of help. After several iterations, we decided to focus on SRA projects at *everis*. It is important to note that this improvement area (i.e., SRA

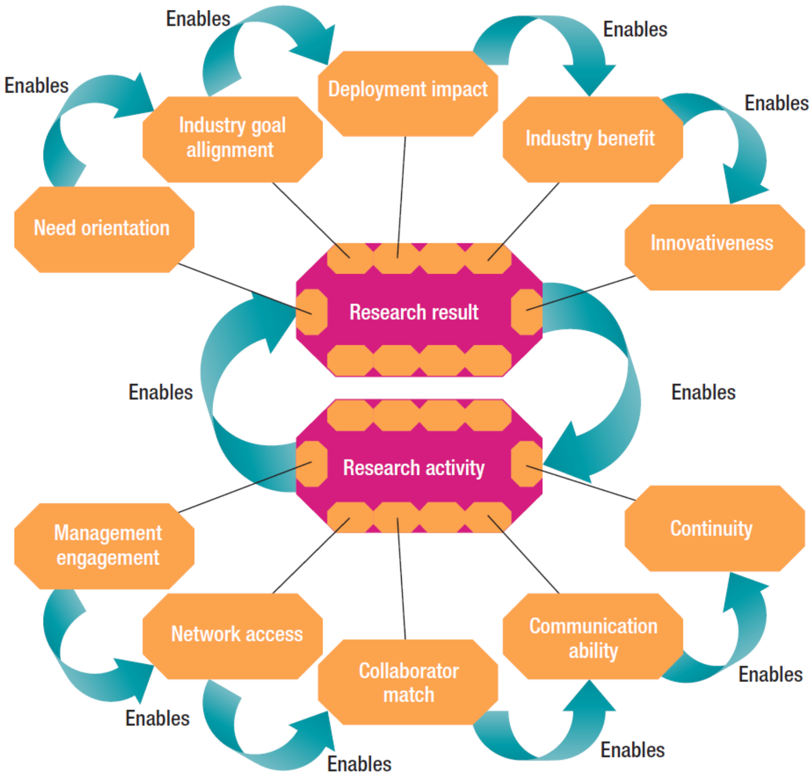

Figure 13.2: A relational model for industry-academia research, from Sandberg et al. [146].

projects) evolved over time, and other objectives that were previously considered were discarded as *everis* demanded (remarkably process monitoring and model-driven development to generate Create, Read, Update and Delete (CRUD) interfaces). Finally, *everis*-relevant needs were divided in two factors: organizational and technical [152], leading to the RQs of Chapter 1.4.

Step 2: Formulate a research agenda. The champions (one from *everis* and another one from GESSI) of the collaboration were defined since the beginning. They have been responsible of formulating the research agenda.

The author of the PhD thesis started to work full-time in the collaboration since October 2011. Also, since November 2011 *everis* provided him an access card and a working space next to the managers at *everis*.

Once the RQs were stated, the first problem for the researchers was to learn the context of SRA projects at *everis* and the vocabulary that practitioners used. It was a tough task. By April 2012, an internal report was created with this information and later published in [26]. When this point was reached, the two managers from *everis* were moved to other projects, given a company's policy about people rotation. A new manager became the contact person at the *everis* side. She has kept internal meetings with the champion at *everis* when necessary. Since then, she has had a long-term commitment to the collaboration. At the GESSI side, also two researchers moved to other projects and one new researcher started to work in the collaboration. Since then (April 2012), the team was composed of one *everis*' manager that is GESSI's contact person at *everis* (plus the champion at *everis*, who is part of the upper management) and three researchers from GESSI (one of them is the champion at GESSI).

During this time, we designed the two main studies of the collaboration: on the one hand, qualitative empirical studies to gather evidence about relevant aspects that could help in the design and evaluation of SRAs; on the other hand, case studies to calculate the ROI of SRA adoption.

The contact person at *everis* helped in contacting multiple practitioners, and monitoring when new SRA projects were conducted to include them in the research.

Step 3: Formulate a candidate solution. From the beginning to the half of the second year of the collaboration (May-October 2012), we formulated the candidate solutions for our two problems.

On the one hand, we jointly identified relevant aspects for the design and use of SRAs. The GESSI team also studied the state-of-the-art about SRA engineering and proposed the proper type of empirical studies to be conducted. In joint meetings, we discussed about the design of the interview guides and the questions of the online questionnaires.

On the other hand, we defined REARM. The GESSI team played an important role in the creation of REARM by studying existing economic models of software reuse and software architecture metrics whereas *everis* provided evidence about available data in their company that makes REARM pragmatic and realistic.

Step 4: Conduct lab validation. Because of the characteristics of the RQs, little validation "inside laboratory" has been performed. First, the qualitative empirical studies were validated to be aligned with existing literature. Second, we needed to perform sensitivity analysis in order to test the robustness of REARM output and to search for errors by encountering unexpected relationships between its inputs and outputs.

Step 5: Perform static validation. In this step, we validated the two solutions devised for the RQs. For the validation of the semi-structured interviews and online questionnaires, two pilot iterations were performed and provided the following feedback:

- Inadequate vocabulary was used to refer to SRA projects' artifacts.
- Researchers did not understand the context of SRA projects properly, additional questions about the SRA project context were included.
- Questions that dealt with several variables disconcerted the interviewee and made the analysis more difficult. It was better to split them to cover only one variable.
- If a survey targets several stakeholders, their questionnaires should be designed having into account their knowledge and interest about architectural concerns.
- The questions should be designed to be easy to follow to avoid that participants reply in questions different than the one intended.
- In online questionnaires, it is recommendable to allow the interviewee to write any comments or clarifications in some field and also include an "n/a" option when necessary. Besides, a previous button is useful to make changes in prior questions.
- Contacting stakeholders from client organizations was harder than contacting interviewees from *everis*. This is mainly because *everis* requested the study, so they had a clear interest on it.

For the validation of REARM, we gathered data of one SRA project in an *everis'* client organization. We performed an internal report for the upper management and the champion in *everis* and the client organization that adopted the SRA. They validated the results, and provided the feedback presented in Chapter 12.1.

A negative point at this step is that, although we have contacted practitioners that participated in our studies to get feedback, we have not addressed widespread presentation of the candidate solution in *everis* or to the upper management.

Step 6: Perform dynamic validation (piloting). In this step, the collaboration team conducted real pilot studies with the candidate solutions.

With respect to the qualitative studies, we conducted a case study in nine SRA projects. The aim was to gather relevant data about: SRAs artifacts (Chapter 5.3), benefits and drawbacks of SRAs (Chapter 5.4), architecturally-significant requirements, and architectural decisions in SRA projects.

With regard to the economic model, we performed a business case for the adoption of an SRA in a public administration in Spain. The results are presented in Chapter 10.

Step 7: Release the solution. The last step is to release the solution to show practitioners how it works and that it is better to use it rather than working as usual. After realizing the solution, practitioners should use it even without the intervention of other people. This step has not been achieved yet. Nevertheless, it is vital to realize industry benefit. Section 13.5.1 describes the next actions to fulfil this step.

13.3 Collaboration Evaluation

In this section we evaluate our industry-academia collaboration following the collaboration model of Sandberg et al. [146]. We evaluate the collaboration maturity and its management with respect to the factors stated by Sandberg et al. [146] (the former five factors relate to *research activities* whereas the second five, to *research results*, see Figure 13.2). To do so, the phase since the beginning of the collaboration to the moment of the evaluation (January 2014) is considered. For each factor we use a Likert scale to assess maturity, with 1 representing low maturity, and 5 representing high maturity.

13.3.1 Research Activity

Management engagement: 5. The problem formulation was defined in the beginning of the collaboration after several meetings in which both representative of *everis* and GESSI were present. The two champions of the collaboration jointly manage the research, and have meetings when necessary (although not as frequent as the rest of members of the collaboration team).

Network access: 3. We have been able to contact best-in-class employees in *everis*. However, since *everis* is a consulting company, sometimes they did not have the competence to provide specific data because of confidentiality issues. Another challenge is to involve them in the data collection process when they are short of time (e.g., busy with other projects).

Collaborator match: 4. Upper management at *everis* is utterly interested in the results of the research, and practitioners have been willing to participate with researchers during the empirical studies.

Communication ability: 3. A very positive point is that GESSI has the option to communicate when necessary to *everis'* managers and other practitioners involved in SRA projects. On the other hand, once we have contacted other practitioners from *everis*, we have not followed their progress in SRA projects.

Continuity: 3. The topics defined in the beginning of collaboration are still being studied. Also, new client organizations are adopting SRAs, so the context is still a current challenge. One representative from GESSI spent one day per week in *everis* from November 2011 to July 2013. From August 2013, he has only attended to meetings because of limited space in *everis*. This is not a big problem due to the geographic proximity of the two institutions and the flexibility of both sides for meeting organization.

13.3.2 Research Result

Need orientation: 4. The collaboration fully addresses a perceived real-life industry problem at *everis*.

Industry goal alignment: 3. Collaboration goals are aligned to current *everis* unit goals, while results are still in an early stage.

Deployment impact: 1. Results have not been deployed by *everis'* practitioners out of the collaboration yet. With the exception of pilots conducted by the joint team, results have not had an impact on practice.

Industry benefit: 2. Results are starting to be valuable to *everis* after the conduction of the first pilots. Yet, practitioners cannot see the results in daily work.

Innovativeness: 2. Internal reports and scientific publications are written by researchers and available by the entire collaboration team. Although they are not used widespread in *everis* yet, they have generated new ideas, knowledge, and publications for the research agenda of the collaboration.

13.4 Lessons Learned

In the previous sections we reported the process that have been followed during the collaboration and evaluated the research activities and results under existing models for industry-academia collaboration. In this section, we report the challenges that we have faced and dealt with in the *"Cátedra everis-UPC"*. Also, we show the most important benefits that have been realized because of mutual collaboration.

Our approach to collect such data was a focus group, which it is considered a proven and tested technique to obtain the perception of a group of selected people on a defined area of interest [152]. The focus group encouraged structured discussions involving participants from the collaboration team. The discussion was largely free-flowing, and everyone has an opportunity to participate. Focus group discussion enables to identify how both industrial and academic partners feel and think about the issues of the collaboration [152]. We reported separately the issues brought by industrial and academic partners.

13.4.1 Challenges

Throughout the collaboration, we have encountered diverse challenges that required special attention. Next, we divide them inside four areas: general, industry, academia and research as defined by Wohlin in [150]. The goal is not to discuss reported challenges in the literature (e.g., [149, 150]), but to discuss the challenges we experienced during the collaboration.

General Challenges. This group relates to challenges to the general relationship between industry and academia.

The general challenges highlighted by the *everis* side are described as follows. First, the *identification of goal* of the collaboration was successfully defined jointly in face-to-face meetings. We focused on solving an industry-relevant problem that could be solved with the expertise of GESSI. Second, *follow-up meetings* have been held regularly. The flexibility of both teams was vital for proper coordination.

Academic partners highlight the following success factors. First, *fluent and direct communication* when necessary among the partners is vital for progressing in the research. The communication between upper management of *everis* and the lead researcher at GESSI required special attention to evolve the goals to up-to-date industry needs. Second, the definition of a *work methodology* (e.g., use of a collaborative environment with a platform to share the results, meetings calendar, internal deliverables roadmap) enabled team work among people that were unknown before the collaboration.

Both partners highlighted the problem of the *changes of people in the collaboration team* due to policies on people rotation or any other event.

Industry Challenges. Challenges in this group concern specific issues to be addressed at the industry side of the collaboration.

Industrial partners uncovered as a weak point *not being leaders of an SRA project* being studied. It is vital to be close to the SRA project to give the most accurate information. In cases in which the *everis*' managers of the collaboration team were not involved in an SRA project or did not know the specifics of such project, it involved extra-effort of another practitioner who was highly involved in the SRA project to work in the collaboration. As a consequence, it is important that the practitioners who temporally join the collaboration have the *adequate role and are able to find the balance to dedicate time* in the research collaboration besides their SRA projects.

Researchers needed to face *difficulties while contacting practitioners* out of the collaboration since their availability is limited. Also, some candidate SRA projects could not be studied as deeper as desired since it was not always possible to *convince management of the everis' client organizations* that were involved in an SRA project. In our consulting context, it was a two-step job (first asking to *everis* champion and then to the client organization). The reasons why we did not study specific SRA projects were mainly confidentiality issues and bureaucratic issues (e.g., it was needed to ask for credential cards to access the client organization, insurance and so on for the researcher to observe or work in an SRA project).

Academia Challenges. In a similar way as for industry, there are some specific challenges related to academia.

At the *everis* side, they found a key issue the *experience of the researcher in SRA projects* (e.g., knowing the technologies being applied). A wrong perception of the context and low experience can jeopardize the results of the collaboration. We paid special attention to this issue in the beginning of the collaboration, in which researchers received tutorials and even developed a demo application based on an SRA to master this technological approach. Another solution, although we did not apply it, could have been to offer training to the researchers as it is done to new practitioners when they are recruited.

Researchers stated the following academia challenges. First, it is important to write *internal reports presenting results, which are not intended to end up as a scientific publication*. This way, deliverables are more relevant to the industry needs (e.g., executive summaries for managers, annual reports, and specific reports for *everis'* clients). Second, additional empirical studies should be conducted only to *understand the real context* in the industry. Third, the *results should be adequately presented to upper management* so that they continue to provide resources needed for taking the next steps.

Research Challenges. The actual conduction of the research comes with some challenges too.

Industrial partners stated the following challenges. First, the importance of *identify realistic sources of data*. In case of quantitative research and economic analysis usually happened that there was not as much historical and project data as needed. The search of data that did not exist, led to dangerous risks such as blocking points. Second, for the economic analysis *adequate scenarios should be designed*. Understanding the alternatives of SRA adoption enables better design of scenarios for decision-making. Third, the obtained *results need to be validated* to analyze that they correspond to the reality. This can be done by iteratively explaining experts the outcome of the research and studying their opinions until they agree that the results are realistic.

For the academia members, the research challenges are the following. First, a big risk is the *period required to start providing value* to the industry. The first results of the collaboration were delivered in the second year, and this situation is not common for industry, which may see that the research is not progressing. Second, in the collection of data our main challenge was how to *face with the incomplete information* that SRA projects may have. This is a serious threat to validate REARM, not just in post-mortem analysis, which could be something expected, but also with ongoing projects in which we experienced obstacles. Third, due to the diverse nature of SRA projects, it is difficult to create *repeatable techniques and results*, since not all SRA projects have the same data. Fourth, it is important to *present results to practitioners*. If this presentation is missed, two big risks potentially arise: the incorrect validation of the results and the no adoption of the techniques devised during the research.

In our opinion, this type of challenge (i.e., research challenges) is the most difficult to overcome. Research challenges highly depend on the context of the research (SRA projects in our case), and sometimes even require ad-hoc solutions.

13.4.2 Mutual Benefits from Collaboration

With respect to the benefits that each partner organization has received from collaboration, we highlight the following ones.

On the one hand, researchers helped practitioners to shape the results of SRA projects into publications and explicit architectural knowledge, since this task was difficult for them from their practical experience. This promotes innovation dissemination and technology transfer inside *everis*. Also, researchers provided feedback from existing research and other tools and techniques from the scientific community, such as experience in the conduction of empirical studies.

On the other hand, the GESSI members appreciate the willingness of *everis*' practitioners to collaborate in the research, which is much harder to achieve without formal industry-academia collaboration. Besides, the involvement of *everis* in the research enabled the possibility to make research to solve real problems in industry.

13.5 Contributions of the Evaluation of the Collaboration

"Collaboration between industry and academia supports improvement and innovation in industry and helps to ensure industrial relevance in academic research" [149].

This chapter describes an industry-academia collaboration: the "*Cátedra everis-UPC*". First, we reported the steps of the collaboration following Gorschek et al. steps [151], and we evaluated our collaboration with an existing model for technology transfer [146]. Second, we held a focus group discussion to identify challenges and problems that we have faced in the collaboration as well as benefits.

On the one hand, after reporting and evaluating the collaboration, we can conclude that it has reached a high maturity. After two years and nine months of collaboration, first results could be seen in form of proposed solutions, internal reports, executive summaries, scientific publications, and pilots run in real projects. However, in order to improve the low levels of maturity in deployment impact and industry benefit, new actions need to be undertaken (see Section 13.5.1). On the other hand, challenges and lessons learned from our collaboration have been discussed. We believe that they are a good contribution to the body of knowledge on conducting empirical studies in industry. Among the most important challenges are: identification of goal of the collaboration, fluent and direct communication, contacting and involving best-in-class employees, industrial experience of researchers, creating internal reports presenting results that are not intended to end up as a scientific publication, understanding the real context in industry, adequately presentation of results to upper management, identifying realistic sources of data, validation of results, facing incomplete information, devising repeatable techniques and results, and last but not least presenting results to practitioners.

13.5.1 Future Steps

Despite the aforementioned progresses, deployment impact, industry benefit, and innovativeness are still ongoing goals. The low level of maturity of these research results is the current main problem of the collaboration. We posit two reasons for this problem: the collaboration is still in an early phase, and the results are not yet articulated to provide lightweight support utilities (i.e., guidelines and artifacts) to support practitioners.

On the one hand, not realizing deployment impact, industry benefit, and innovativeness at early phases of industry-academia collaboration is a common situation, as reported in [151, 147]. For this reason, we consider highly recommendable the evaluation of industry-academia collaboration in order to incrementally improve its success throughout all phases. On the other hand, we need to pay special attention to the packaging of the results in order to release a lightweight solution that can be used by practitioners without the intervention of the other people.

The evaluation of our collaboration has enabled to identify the next steps to be taken to achieve a high degree of technology transfer and innovation dissemination:

1. Providing tool support to practitioners so that they can easily apply the envisaged economic model (i.e., REARM), as well as demo applications and case studies that use REARM as example.

2. Reporting the evidence of the qualitative studies about benefits and drawbacks of SRAs, SRAs artifacts, architecturally-significant require-

ments in SRA projects, and architectural decisions. This promotes innovation dissemination and technology transfer of SRA engineering inside the company. Scientific publications are not a good approach for dissemination in industry. Instead, lightweight materials (e.g., presentations, executive summaries) are being created.

3. Creating practitioners-oriented prescriptive support utilities (i.e., guidelines and artifacts). Besides, widespread celebration of workshops and training courses from the collaboration team to *everis*' practitioners involved in SRA projects should be performed.

Chapter 14

Conclusions and Future Work

This document has described my PhD thesis. In this chapter we present the conclusions of the research conducted by:

- Answering the initial RQs.
- Stating the contributions with respect to the SRA and ESE research fields.
- Discussing the possible future work from the current state of the research.

14.1 Conclusions and Answers to RQ 1 and RQ 2

As we stated in Chapter 1.4, the research goal of this PhD thesis is to support software architects making informed decisions about SRAs acquisition, design, and use based on empirical evidence. To operationalize this goal, we stated two RQs. In this section, we provide answers to these two RQs:

RQ 1: How can an organization get corporate evidence that is useful for the SRA engineering?

This PhD thesis offers guidelines for organizations in order to understand, evaluate, and improve their SRA engineering. Software architects can apply such guidelines, available at Chapter 6, to gather architectural knowledge. The guidelines consist of:

- The context of SRAs in industry. Software architects who may want to use these guidelines need to study whether their organizations fit into this context.
- A set of practical important aspects of SRAs, to check which ones are important for the organization.
- Template of interview guides and online questionnaires to gather data about such aspects. By using this template in the design of a survey or online questionnaire, organizations can get a corporate knowledge base about practical aspects of their SRA engineering.

RQ 2: Is it worth for an organization to invest on the adoption of an SRA?

This PhD thesis offers guidelines to build the business case for SRAs in an organization. Software architects can apply such guidelines, available at Chapter 11, which consist of:

- The context of SRAs in industry. As in the previous RQ 1, software architects who may want to use these guidelines need to study whether their organizations fit into this context.
- A set of commonly available value-driven data in SRA projects, to check the metrics that could be gathered from realistic sources of data.
- A reuse-based economic model for SRAs that facilitates building the business case, called REARM. REARM indicates how to calculate the ROI of an SRA adoption from the available data and metrics.

It is important to note that, although it is not strictly necessary, these guidelines are complementary and support each other (e.g., results from a preceding study can be used to corroborate or further develop other results). For instance, in our industry-academia collaboration, the qualitative results about the benefits and drawbacks of SRAs (RQ 1) supported the unquantifiable benefits, and uncertainties and risk of the business case (RQ 2).

14.2 Contributions to the SRA and ESE Theories

ESE had already been used in the software architecture area before this PhD thesis. However, as a consequence of designing and applying the aforementioned guidelines, this PhD thesis has contributed to the SRA theory and the ESE theory. Falessi et al. indicate: "empirical research can provide the results on which to build and/or assess the theoretical foundations underpinning various software architecture-related technologies" [27]. Therefore, we emphasize below how the empirical studies of this PhD thesis have contributed to SRA theory. Besides, they add: "experiences and lessons learned from empirically assessing software architecture research represent a valuable -but often underestimated- means of improving the application of the empirical paradigm to software architecture research and practice" [27]. Thus, we also present our lessons learned from designing and conducting our empirical studies to advance the ESE theory. These contributions are summarized in Figure 14.1, and respectively presented in the following two subsections.

14.2.1 Contributions to the SRA Theory

The research tasks of this PhD thesis increased the empirical evidence available on SRAs and led to the following novel contributions to the SRA theory:

- **Identification of the existing topics covered in the scientific literature regarding SRA engineering**. The SLR presented in Chapter 3 revealed the eight big topics that have been addressed by SRA engineering research: understanding theoretical concepts of SRAs; deciding on the adoption of SRAs; looking for information to build SRAs; eliciting requirements of SRAs; taking decisions about the design of SRAs; evaluating the design of SRAs; using SRAs to design concrete architectures of software systems; and, evolving SRAs.

- **Identification of practical review criteria for SRAs**. The main result of the initial meetings of our industry-academia collaboration, presented in Chapter 4, was the identification of five relevant aspects for SRA engineering: overview and type of an SRA; requirements and quality attributes analysis; architectural knowledge and decisions; supportive technologies; and, business qualities and architecture competence.

- **Study of the artifacts of SRAs in *everis'* client organizations**. The artifacts that compose an SRA in the *everis* context, presented in Chapter

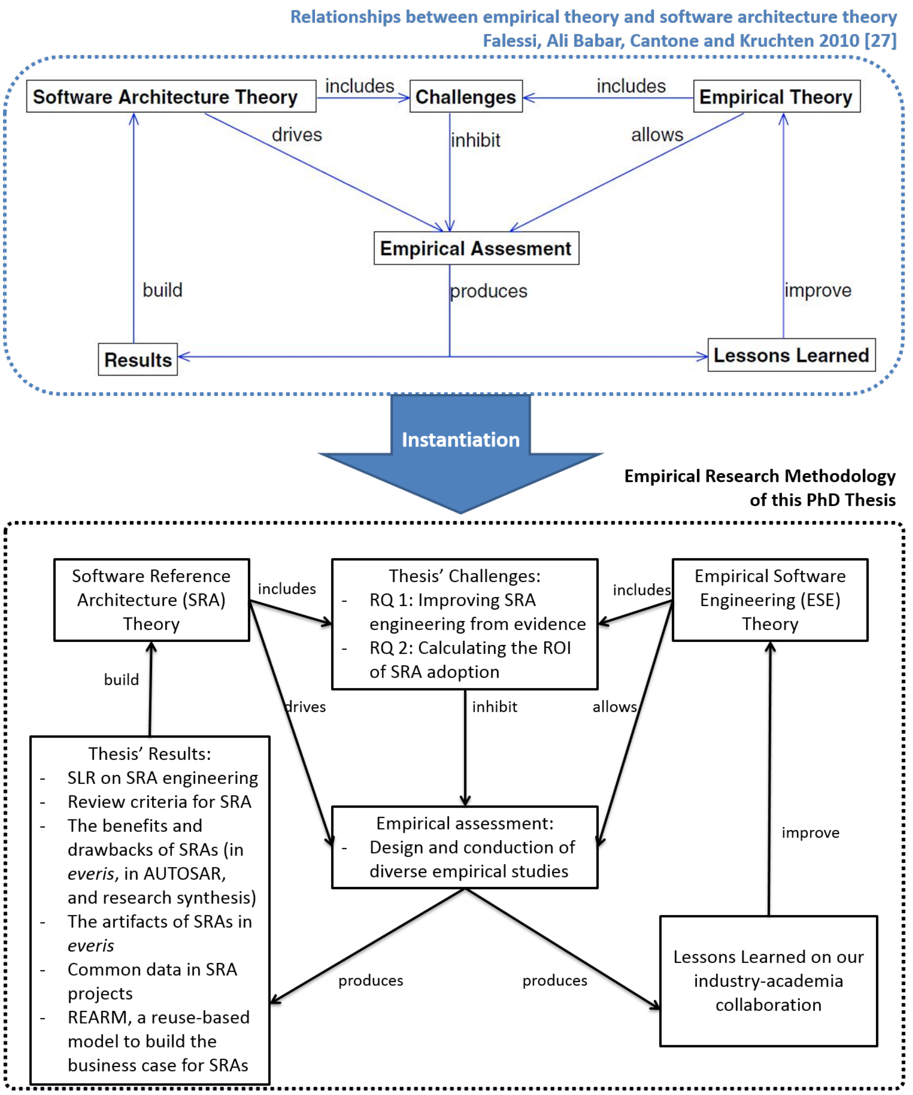
Relationships between empirical theory and software architecture theory
Falessi, Ali Babar, Cantone and Kruchten 2010 [27]
Software Architecture Theory
includes
Challenges
includes
Empirical Theory
drives
inhibit
allows
Empirical Assesment
build
produces
improve
Results
Lessons Learned
Instantiation
Empirical Research Methodology of this PhD Thesis
Software Reference Architecture (SRA) Theory
includes
Thesis' Challenges:
- RQ 1: Improving SRA engineering from evidence
- RQ 2: Calculating the ROI of SRA adoption
includes
Empirical Software Engineering (ESE) Theory
build
drives
inhibit
allows
Thesis' Results:
- SLR on SRA engineering
- Review criteria for SRA
- The benefits and drawbacks of SRAs (in everis, in AUTOSAR, and research synthesis)
- The artifacts of SRAs in everis
- Common data in SRA projects
- REARM, a reuse-based model to build the business case for SRAs
Empirical assessment:
- Design and conduction of diverse empirical studies
improve
produces
produces
Lessons Learned on our industry-academia collaboration

Figure 14.1: Novel contributions of this PhD thesis to the SRA theory and the ESE theory.

5.3, are: common software elements (i.e., software components) aimed to be reused for all the applications; guidelines for the homogeneous development of application; and, documentation that describe the logical solution to create a set of applications.

- **Study of the benefits and drawbacks of SRAs in *everis*' client organizations**. The main benefits of SRAs in *everis*, presented in Chapter 5.4, are reduced development and maintenance costs, and easier development. The main drawback in the *everis* context is the extra learning curve to master in SRA-based developments.
- **Study of the benefits and drawbacks of SRAs for AUTOSAR practitioners**. The survey done to AUTOSAR practitioners, presented in Chapter 7, indicated that the most popular benefits of AUTOSAR are standardization, reuse and interoperability whereas its most important drawbacks are complexity, initial investment and the learning curve.
- **Consolidating a theory of the benefits and drawbacks of SRAs in industry**. After analyzing the benefits and drawbacks of SRAs in the contexts of *everis* and AUTOSAR, we synthesized such empirical evidence with other three empirical studies of other researchers. This synthesis, presented in Chapter 8, showed that five SRA benefits considerably increased their belief value after aggregation: interoperability of software systems, reduced development costs, improved communication among stakeholders, reduced risk, and reduced time-to-market. Also, one drawback of SRAs increased its belief value: the required learning curve for developers. These aggregated results consolidated knowledge and confidence on the studied SRA effects (i.e., benefits and drawbacks).
- **Study of the common available value-driven data in SRA projects in *everis***. The survey to identify the common data in SRA projects, presented in Chapter 9, revealed the following available data: effort metrics to calculate SRA projects' costs, and software metrics to calculate benefits in reuse and maintainability.
- **Calculation of the ROI of adopting an SRA in an *everis* client organization**. We performed a cost-benefit analysis of an SRA adoption in a public administration using REARM, presented in Chapter 10, and it showed that the SRA payed off after creating 7 small applications. With medium to large applications, this number could be reduced to 2 applications.

14.2.2 Contributions to the ESE Theory

In this subsection, we report the challenges that we have faced and dealt with in the "Cátedra everis-UPC". Our approach to collect such data was a focus group, which was presented in Chapter 13. Among the most important challenges and lessons learned are: identification of the goal of the collaboration, fluent and direct communication, contacting and involving best-in-class employees, industrial experience of researchers, creating internal reports presenting results that are not intended to end up as a scientific publication, understanding the real context in industry, adequately presentation of results to upper management, identifying realistic sources of data, validation of results, facing incomplete information, devising repeatable techniques and results, and presenting results to practitioners.

14.2.3 Overall Contributions

As a final conclusion, the most valuable outcome of this PhD thesis as a whole is the formulation of guidelines to conduct empirical studies for both supporting gathering evidence and building a business case for SRAs, and how their application in our industry-academia collaboration has contributed to the SRA and ESE theories.

14.3 Future Work

We divide the future work in three parts, corresponding to the three first parts of this document:

First, regarding the state-of-the-art, we plan to make a mapping study about SRAs published in the literature by using the search string of our SLR.

Second, with respect to RQ 1, we plan to gather empirical evidence of more relevant aspects using the guidelines (e.g., requirements) and consolidating such evidence.

Third, regarding RQ 2, future versions of REARM could include more aspects, which we elicited in the feedback sessions of Chapter 12.

References

[1] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice (2nd Edition)*. Addison-Wesley Professional, 2003.

[2] E. Y. Nakagawa, P. O. Antonino, and M. Becker, “Reference architecture and product line architecture: A subtle but critical difference,” in *5th European Conference on Software Architecture*, ser. LNCS, vol. 6903, 2011, pp. 207–211.

[3] R. Cloutier, G. Muller, D. Verma, R. Nilchiani, E. Hole, and M. Bone, “The concept of reference architectures,” *Systems Engineering*, vol. 13, no. 1, pp. 14–27, 2010.

[4] S. Angelov, P. Grefen, and D. Greefhorst, “A framework for analysis and design of software reference architectures,” *Information and Software Technology*, vol. 54, no. 4, pp. 417–431, 2012.

[5] S. Martínez-Fernández, C. Ayala, X. Franch, and H. Martins, “REARM: A Reuse-Based Economic Model for Software Reference Architectures,” in *13th International Conference on Software Reuse (ICSR)*, ser. LNCS, vol. 7925, 2013, pp. 97–112.

[6] G. Muller and P. van de Laar, “Researching reference architectures,” in *Views on Evolvability of Embedded Systems*, ser. Embedded Systems. Springer Netherlands, 2011, pp. 107–119.

[7] D. L. Parnas, “On the design and development of program families,” *IEEE Transactions on Software Engineering*, vol. SE-2, no. 1, pp. 1–9, 1976.

[8] D. Scott, “Gartner hype cycle for real-time infrastructure,” 2012. [Online]. Available: https://www.gartner.com/doc/2098715/hype-cycle-realtime-infrastructure-

[9] M. Galster, "Software Reference Architectures: Related Architectural Concepts and Challenges," in *1st International Workshop on Exploring Component-based Techniques for Constructing Reference Architectures (CobRA)*, 2015, pp. 5–8.

[10] M. Galster and P. Avgeriou, "Empirically-grounded reference architectures: a proposal," in *ACM SIGSOFT conference on Quality of Software Architectures (QoSA)*, 2011, pp. 153–158.

[11] R. Haesevoets, D. Weyns, and T. Holvoet, "Architecture-centric support for adaptive service collaborations," *ACM Transactions on Software Engineering and Methodology*, vol. 23, no. 1, pp. 1–40, 2014.

[12] R. Behjati, S. Nejati, and L. C. Briand, "Architecture-Level Configuration of Large-Scale Embedded Software Systems," *ACM Transactions on Software Engineering and Methodology*, vol. 23, no. 3, pp. 1–43, 2014.

[13] E. Y. Nakagawa, F. Oquendo, and M. Becker, "RAModel: A Reference Model for Reference Architectures," in *Joint Working IEEE/IFIP Conference on Software Architecture (WICSA) and European Conference on Software Architecture (ECSA)*, 2012, pp. 297–301.

[14] The Open Group, "SOA Reference Architecture," 2011. [Online]. Available: https://www2.opengroup.org/ogsys/jsp/publications/PublicationDetails.jsp?publicationid=12490

[15] C. Ballard, C. Compert, T. Jesionowski, I. Milman, B. Plants, B. Rosen, and H. Smith, "IBM Redbooks | Information Governance Principles and Practices for a Big Data Landscape," 2014. [Online]. Available: http://www.redbooks.ibm.com/abstracts/sg248165.html?Open

[16] A. Grosskurth and M. Godfrey, "A reference architecture for Web browsers," in *21st IEEE International Conference on Software Maintenance (ICSM)*, 2005, pp. 661–664.

[17] L. Bueno, R. Oliveira, and E. Y. Nakagawa, "A Service-Oriented Reference Architecture for Software Testing Tools," in *European Conference on Software Architecture (ECSA)*, 2011, pp. 405–421.

[18] AUTOSAR, "AUTOSAR – The Worldwide Automotive Standard for E/E Systems," *ATZextra worldwide*, vol. 18, no. 9, pp. 5–12, Oct. 2013.

[19] NASA, "ESDS Reference Architecture for the Decadal Survey Era," 2012. [Online]. Available: https://earthdata.nasa.gov/sites/default/files/field/document/ESDSReferenceArchitecturev1.1.pdf

[20] S. Mazzini, J. Favaro, and T. Vardanega, "Cross-Domain Reuse : Lessons Learned in a Multi-project Trajectory," in *International Conference on Software Reuse (ICSR)*, 2013, pp. 113–126.

[21] S. Brand, "Accenture Has Business-Outcome-Driven EA Capabilities, but Doesn't Automatically Begin With This Approach," 2014. [Online]. Available: https://www.gartner.com/doc/2651816/accenture-businessoutcomedriven-ea-capabilities-doesnt

[22] S. Angelov, J. Trienekens, and R. Kusters, "Software Reference Architectures - Exploring Their Usage and Design in Practice," in *European Conference on Software Architecture (ECSA)*, ser. LNCS, vol. 7957, 2013, pp. 17–24.

[23] S. Angelov, J. Trienekens, and P. Grefen, "Towards a method for the evaluation of reference architectures: Experiences from a case," in *European Conference on Software Architecture (ECSA)*, ser. LNCS, vol. 5292, 2008, pp. 225–240.

[24] N. Qureshi, M. Usman, and N. Ikram, "Evidence in software architecture, a systematic literature review," in *17th International Conference on Evaluation and Assessment in Software Engineering (EASE)*, 2013, pp. 97–106.

[25] D. J. Reifer, *Making the Software Business Case: Improvement by the Numbers*. Addison-Wesley Professional, 2001.

[26] S. Martínez-Fernández, C. Ayala, X. Franch, and D. Ameller, "A Framework for Software Reference Architecture Analysis and Review," in *Memorias del X Workshop Latinoamericano de Ingeniería en Software Experimental (ESELAW) - ISBN 978-9974-8379-3-5*, 2013, pp. 89–102.

[27] D. Falessi, M. A. Babar, G. Cantone, and P. Kruchten, "Applying empirical software engineering to software architecture: challenges and lessons learned," *Empirical Software Engineering*, vol. 15, no. 3, pp. 250–276, 2010.

[28] M. Brydon-Miller, D. Greenwood, and P. Maguire, "Why Action Research?" *Action Research*, vol. 1, no. 1, pp. 9–28, 2003.

[29] S. Martínez-Fernández, "A Framework for Software Reference Architecture Analysis and Review. PhD Proposal," UPC BarcelonaTech, 2013.

[30] S. Martínez-Fernández, "Towards Supporting the Adoption of Software Reference Architectures: An Empirically-Grounded Framework," in *11th International Doctoral Symposium on Empirical Software Engineering (IDoESE)*, 2013. [Online]. Available: http://umbc.edu/eseiw2013/idoese/pdf/eseiw2013_IDoESE_180.pdf

[31] S. Martínez-Fernández, C. P. Ayala, X. Franch, H. Martins Marques, and D. Ameller, "Towards Guidelines for Building a Business Case and Gathering Evidence of Software Reference Architectures in Industry," *Journal of Software Engineering Research and Development (JSERD)*, vol. 2, no. 7, Aug. 2014.

[32] S. Martínez-Fernández, L. B. Ruas de Oliveira, C. P. Ayala, X. Franch, and E. Y. Nakagawa, "Planning a Systematic Review on Business Case for Reference Architectures," Poster Session from Component-Based Software Engineering and Software Architecture federated conference (CompArch), 2014. [Online]. Available: http://www.essi.upc.edu/~smartinez/wp-content/papercite-data/pdf/martinez-fernandez2014planning.pdf

[33] S. Martínez-Fernández, J. Bisbal, and X. Franch, "Accuracy Assessment of Forecasting Services (poster)," 1st European Business Intelligence Summer School (eBISS), 2011. [Online]. Available: http://cs.ulb.ac.be/conferences/ebiss2011/files/martinez.pdf

[34] S. Martínez-Fernández, J. Bisbal, and X. Franch, "QuPreSS: A Service-Oriented Framework for Predictive Services Quality Assessment," in *7th International Conference on Knowledge Management in Organizations: Service and Cloud Computing (KMO)*, ser. Advances in Intelligent Systems and Computing, vol. 172, 2013, pp. 525–536.

[35] S. Martínez-Fernández, X. Franch, and J. Bisbal, "Verifying Predictive Services' Quality with Mercury," in *4th International Workshop on Academic Software Development Tools and Techniques (WASDeTT)*, 2013.

[36] S. Martínez-Fernández, X. Franch, and J. Bisbal, "Mercury: Using the QuPreSS Reference Model to Evaluate Predictive Services," *Science of Computer Programming*, 2016.

[37] S. Martínez-Fernández, C. Ayala, X. Franch, and H. Marques, "Benefits and Drawbacks of Reference Architectures," in *7th European Conference on Software Architecture (ECSA)*, ser. LNCS, vol. 7957, 2013, pp. 307–310.

[38] S. Martínez-Fernández, C. Ayala, X. Franch, and H. M. Marques, "Artifacts of Software Reference Architectures: A Case Study," in *18th International Conference on Evaluation and Assessment in Software Engineering (EASE)*, 2014, pp. 42:1–42:10.

[39] S. Martínez-Fernández, D. Ameller, C. Ayala Martínez, X. Franch, and X. Terradellas Fernandez, "Conducting empirical studies on reference architectures in IT consulting firms," ESSI-TR-12-2, Tech. Rep., 2012.

[40] S. Martínez-Fernández, C. P. Ayala, X. Franch, and E. Y. Nakagawa, "A Survey on the Benefits and Drawbacks of AUTOSAR," in *1st International Workshop on Automotive Software Architecture (WASA)*, 2015, pp. 19–26.

[41] S. Martínez-Fernández, P. S. Medeiros Dos Santos, C. Ayala, X. Franch, and G. H. Travassos, "Aggregating Empirical Evidence about the Benefits and Drawbacks of Software Reference Architectures," in *ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2015, pp. 154–163.

[42] S. Martínez-Fernández, C. Ayala Martínez, and X. Franch, "A reuse-based economic model for software reference architectures," ESSI-TR-12-6, Tech. Rep., 2012.

[43] S. Martínez-Fernández and H. M. Marques, "Practical Experiences in Designing and Conducting Empirical Studies in Industry-Academia Collaboration," in *2nd International Workshop on Conducting Empirical Studies in Industry (CESI)*, 2014, pp. 15–20.

[44] D. E. Perry and A. L. Wolf, "Foundations for the study of software architecture," *ACM SIGSOFT Software Engineering Notes*, vol. 17, no. 4, pp. 40–52, 1992.

[45] P. Kruchten, *The rational unified process: an introduction*. Addison-Wesley Professional, 2004.

[46] E. R. Poort and H. Van Vliet, "RCDA: Architecting as a risk-and cost management discipline," *Journal of Systems and Software*, vol. 85, no. 9, pp. 1995–2013, 2012.

[47] I. Averdunk, "Tivoli reference architectures," 2012. [Online]. Available: https://www.ibm.com/developerworks/mydeveloperworks/blogs/tivoli-ra/entry/welcome_to_the_tivoli_reference_architecture_blog12?lang=es

[48] A. Wilson, D. M. Lindholm, and C. LASP, "Towards a Domain Specific Software Architecture for Scientific Data Distribution," in *AGU Fall Meeting Abstracts*, vol. 1, 2011, p. 1609.

[49] Microsoft, "Microsoft Industry Reference Architecture for Banking (MIRA-B)," 2012. [Online]. Available: http://www.microsoft.com/en-us/news/download/presskits/msfinancial/docs/MIRAB.pdf

[50] B. Graaf, H. Van Dijk, and A. van Deursen, "Evaluating an Embedded Software Reference Architecture – Industrial Experience Report," in *9th European Conference on Software Maintenance and Reengineering (CSMR)*, 2005, pp. 354–363.

[51] L. R. Welch, M. W. Masters, L. A. Madden, D. T. Marlow, P. M. Irey IV, P. V. Werme, and B. A. Shirazi, "A distributed system reference architecture for adaptive QoS and resource management," in *10th Symposium on Parallel and Distributed Processing*, ser. LNCS, vol. 1586, 1999, pp. 1316–1326.

[52] T. J. Williams, "The purdue enterprise reference architecture," *Computers in industry*, vol. 24, no. 2, pp. 141–158, 1994.

[53] P. Clements, R. Kazman, and M. Klein, *Evaluating software architectures*. Addison-Wesley Reading, 2001.

[54] I. C. Society, "IEEE STANDARD 1471-2000 - IEEE Recommended Practice for Architectural Description for Software-Intensive Systems," 2000.

[55] M. Galster, P. Avgeriou, and D. Tofan, "Constraints for the design of variability-intensive service-oriented reference architectures–An industrial case study," *Information and Software Technology*, vol. 55, no. 2, pp. 428–441, 2013.

[56] K. Pohl, G. Böckle, and F. J. van der Linden, *Software product line engineering: foundations, principles, and techniques*. Springer, 2005.

[57] P. Clements and L. Northrop, *Software product lines*. Addison-Wesley Boston, 2002.

[58] U. Eklund, N. Jonsson, J. Bosch, and A. Eriksson, "A reference architecture template for software-intensive embedded systems," in *WICSA/ECSA Companion Volume*, 2012, pp. 104–111.

[59] B. P. Gallagher, "Using the Architecture Tradeoff Analysis Method to Evaluate a Reference Architecture: A Case Study," CMU/SEI-2000-TN-007, Tech. Rep., 2000.

[60] E. Y. Nakagawa, M. Guessi, J. C. Maldonado, D. Feitosa, and F. Oquendo, "Consolidating a Process for the Design, Representation, and Evaluation of Reference Architectures," in *IEEE/IFIP Conference on Software Architecture (WICSA)*, 2014, pp. 143–152.

[61] R. Weinreich and G. Buchgeher, "Automatic Reference Architecture Conformance Checking for SOA-Based Software Systems," in *IEEE/IFIP Conference on Software Architecture (WICSA)*, 2014, pp. 95–104.

[62] M. Guessi, L. de Oliveira, and E. Nakagawa, "Representation of Reference Architectures: A Systematic Review," in *International Conference on Software Engineering & Knowledge Engineering (SEKE)*, 2011, pp. 782–785.

[63] V. Zani, D. Feitosa, and E. Nakagawa, "Current State of Reference Architectures in the Context of Agile Methodologies." in *International Conference on Software Engineering & Knowledge Engineering (SEKE)*, 2011, pp. 590–595.

[64] L. Bueno Ruas de Oliveira, K. Felizardo, and E. Y. Nakawaga, "Reference models and reference architectures based on service-oriented architecture: a systematic review," in *European Conference on Software Architecture (ECSA)*, 2010, pp. 360–367.

[65] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Keele University and University of Durham EBSE-2007-01, Tech. Rep., 2007. [Online]. Available: https://www.cs.auckland.ac.nz/~norsaremah/2007GuidelinesforperformingSLRinSEv2.3.pdf

[66] O. Dieste and A. G. Padua, "Developing Search Strategies for Detecting Relevant Experiments for Systematic Reviews," in *International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2007, pp. 215–224.

[67] T. Dyba, T. Dingsoyr, and G. K. Hanssen, “Applying Systematic Reviews to Diverse Study Types: An Experience Report,” in *International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2007, pp. 225–234.

[68] C. Wohlin, P. Runeson, P. A. da Mota Silveira Neto, E. Engström, I. do Carmo Machado, and E. S. de Almeida, “On the reliability of mapping studies in software engineering,” *Journal of Systems and Software*, vol. 86, no. 10, pp. 2594–2610, 2013.

[69] J. Cohen, “Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit,” *Pychol Bull*, vol. 70, no. 4, pp. 213–220, 1968.

[70] S. Herold and M. Mair, “Checking Conformance with Reference Architectures: A Case Study,” in *17th IEEE International Enterprise Distributed Object Computing Conference (EDOC)*, 2013, pp. 71–80.

[71] C. Mattmann and R. Downs, “Reuse of software assets for the NASA Earth science decadal survey missions,” in *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 2010, pp. 1687–1690.

[72] H. J. Beyer, D. Hein, C. Schitter, J. Knodel, D. Muthig, and M. Naab, “Introducing architecture-centric reuse into a small development organization,” in *International Conference on Software Reuse (ICSR)*, ser. LNCS, vol. 5030, 2008, pp. 1–13.

[73] R. Farenhorst and H. van Vliet, “Understanding How to Support Architects in Sharing Knowledge,” in *Workshop on Sharing and Reusing Architectural Knowledge (SHARK)*, 2009, pp. 17–24.

[74] M. Irlbeck and D. Bytschkow, “Towards a bottom-up development of reference architectures for smart energy systems,” in *2nd International Workshop on Software Engineering Challenges for the Smart Grid (SE4SG)*, 2013, pp. 9–16.

[75] AUTOSAR, “Development Partnership AUTOSAR to extend scope of applications to non-automotive areas,” 2011. [Online]. Available: http://www.autosar.org/fileadmin/files/media_release/Development_Partnership_AUTOSAR_to_extend_scope_of_applications_to_non-automotive_areas_EN.pdf

[76] G. Buchgeher and R. Weinreich, "Towards continuous reference architecture conformance analysis," in *7th European Conference on Software Architecture (ECSA)*, 2013, pp. 332–335.

[77] L. Passos, R. Terra, M. T. Valente, R. Diniz, and N. Mendonça, "Static Architecture-Conformance Checking: An Illustrative Overview," *IEEE Software*, vol. 27, no. 5, pp. 82–89, 2010.

[78] L. Dobrica and E. Ovaska, "Analysis of a cross-domain reference architecture using change scenarios," in *5th European Conference on Software Architecture (ECSA) Companion Volume*, 2011, pp. 10:1—-10:9.

[79] P. Reed, "Reference Architecture: The best of best practices," Sep. 2002. [Online]. Available: http://www.ibm.com/developerworks/rational/library/2774.html

[80] M. Henning, "The Rise and Fall of CORBA," *ACM Queue*, vol. 4, no. 5, pp. 28–34, 2006.

[81] L. M. Northrop and P. C. Clements, "A framework for software product line practice, version 5.0." [Online]. Available: http://www.sei.cmu.edu/productlines/frame_report/index.html

[82] B. W. Boehm, "Value-based software engineering: Seven key elements and ethical considerations," in *Value-Based Software Engineering*. Springer, 2006, pp. 109–132.

[83] M. S. Ali, M. A. Babar, and K. Schmid, "A comparative survey of economic models for software product lines," in *35th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2009, pp. 275–278.

[84] Forrester, "Total Economic Impact (TEI)," 2013. [Online]. Available: https://www.forrester.com/marketing/product/consulting/tei.html

[85] P. Clements, J. D. McGregor, and S. G. Cohen, "The Structured Intuitive Model for Product Line Economics (SIMPLE)," CMU/SEI-2005-TR-003, Tech. Rep., 2005.

[86] D. Ganesan, D. Muthig, and K. Yoshimura, "Predicting return-on-investment for product line generations," in *10th International Software Product Line Conference (SPLC)*, 2006, pp. 13–24.

[87] J. S. Poulin, *Measuring software reuse: principles, practices, and economic models*. Addison–Wesley, 1997.

[88] B. Boehm, A. W. Brown, R. Madachy, and Y. Yang, "A software product line life cycle cost estimation model," in *International Symposium on Empirical Software Engineering (ISESE)*, 2004, pp. 156–164.

[89] K. Schmid, "An initial model of product line economics," in *4th International Workshop Software Product-Family Engineering (PFE)*, ser. LNCS, 2002, pp. 38–50.

[90] J. P. Nóbrega, E. Almeida, and S. R. L. Meira, "Income: Integrated cost model for product line engineering," in *34th Euromicro Conference Software Engineering and Advanced Applications (SEAA)*, 2008, pp. 27–34.

[91] W. Frakes and C. Terry, "Software reuse: metrics and models," *ACM Computing Surveys (CSUR)*, vol. 28, no. 2, pp. 415–435, 1996.

[92] A. Mili, S. F. Chmiel, R. Gottumukkala, and L. Zhang, "An integrated cost model for software reuse," in *International Conference on Software Engineering (ICSE)*, 2000, pp. 157–166.

[93] R. Kazman, J. Asundi, and M. Klien, "Making architecture design decisions: An economic approach," CMU/SEI-2002-TR-035, ESC-TR-2002-035, Tech. Rep., 2002.

[94] I. Ozkaya, R. Kazman, and M. Klein, "Quality-attribute based economic valuation of architectural patterns," in *1st International Workshop on the Economics of Software and Computation (ESC)*, 2007, pp. 5–5.

[95] D. Falessi, P. Kruchten, and G. Cantone, "Issues in applying empirical software engineering to software architecture," in *1st European Conference on Software Architecture (ECSA)*, ser. LNCS, 2007, vol. 4758, pp. 257–262.

[96] J. Carriere, R. Kazman, and I. Ozkaya, "A cost-benefit framework for making architectural decisions in a business context," in *32nd International Conference on Software Engineering (ICSE)*, vol. 2, 2010, pp. 149–157.

[97] A. MacCormack, C. Baldwin, and J. Rusnak, "Exploring the duality between product and organizational architectures: A test of the "mirroring" hypothesis," *Research Policy*, vol. 41, no. 8, pp. 1309–1324, 2012.

[98] C. Y. Baldwin and K. B. Clark, *Design rules: The power of modularity*. The MIT Press, 2000, vol. 1.

[99] R. L. Nord, I. Ozkaya, P. Kruchten, and M. Gonzalez-Rojas, "In search of a metric for managing architectural technical debt," in *Joint Working IEEE/IFIP Conference on Software Architecture (WICSA) and European Conference on Software Architecture (ECSA)*, 2012, pp. 91–100.

[100] J. Letouzey, "The SQALE method for evaluating Technical Debt," in *3rd International Workshop on Managing Technical Debt (MTD)*, 2012, pp. 31–36.

[101] M. Khurum, T. Gorschek, and K. Petersson, "Systematic review of solutions proposed for product line economics," in *12th International Conference Software Product Lines (SPLC)*, vol. 2, 2008, pp. 277–284.

[102] L. Bass, P. Clements, R. Kazman, and M. Klein, "Evaluating the software architecture competence of organizations," in *7th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, 2008, pp. 249–252.

[103] S. Montagud, S. Abrahão, and E. Insfran, "A systematic review of quality attributes and measures for software product lines," *Software Quality Journal*, vol. 20, no. 3-4, pp. 425–486, 2011.

[104] F. van der Linden, J. Bosch, E. Kamsties, K. Känsälä, L. Krzanik, and H. Obbink, "Software product family evaluation," in *Software Product-Family Engineering*, ser. LNCS. Springer Berlin Heidelberg, 2004, vol. 3014, pp. 352–369.

[105] S. Deelstra, M. Sinnema, and J. Bosch, "Product derivation in software product families: a case study," *Journal of Systems and Software*, vol. 74, no. 2, pp. 173–194, 2005.

[106] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering*, vol. 14, no. 2, pp. 131–164, 2009.

[107] P. Runeson, M. Host, A. Rainer, and B. Regnell, *Case Study Research in Software Engineering: Guidelines and Examples*. Wiley, 2012.

[108] U. Eklund, Ö. Askerdal, J. Granholm, A. Alminger, and J. Axelsson, "Experience of introducing reference architectures in the development

of automotive electronic systems," *ACM SIGSOFT Software Engineering Notes*, vol. 30, no. 4, pp. 1–6, 2005.

[109] S. Murer and C. Hagen, "Fifteen Years of Service-Oriented Architecture at Credit Suisse," *IEEE Software*, vol. 31, no. 6, pp. 9–15, 2014.

[110] P. B. Seddon and R. Scheepers, "Towards the improved treatment of generalization of knowledge claims in IS research: drawing general conclusions from samples," *European Journal of Information Systems*, vol. 21, no. 1, pp. 6–21, 2011.

[111] R. Winter and R. Fischer, "Essential Layers, Artifacts, and Dependencies of Enterprise Architecture," in *10th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW)*, 2006, p. 30.

[112] M. Panunzio and T. Vardanega, "On Software Reference Architectures and Their Application to the Space Domain," in *International Conference on Software Reuse (ICSR)*, 2013, pp. 144–159.

[113] Microsoft Patterns & Practices Team, *Microsoft Application Architecture Guide*. Microsoft, 2009. [Online]. Available: http://www.microsoft.com/en-us/download/details.aspx?id=16236

[114] J. García-Alonso, J. B. Olmeda, and J. M. Murillo, "Java para Aplicaciones Corporativas de la Administración," in *Jornadas de Ingeniería del Software y Bases de Datos (JISBD)*, E. Teniente and S. Abrahão, Eds., 2010, pp. 263–266.

[115] G. Reichart and M. Haneberg, "Key Drivers for a Future System Architecture in Vehicles," in *Convergence International Congress & Exposition On Transportation Electronics*, 2004.

[116] AUTOSAR, "AUTomotive Open System ARchitecture: AUTOSAR's site." [Online]. Available: http://www.autosar.org/

[117] U. Eklund and J. Bosch, "Architecture for Embedded Open Software Ecosystems," *Journal of Systems and Software*, vol. 92, pp. 128–142, 2014.

[118] T. Lethbridge, S. Sim, and J. Singer, "Studying Software Engineers: Data Collection Techniques for Software Field Studies," *Empirical Software Engineering*, vol. 10, no. 3, pp. 311–341, 2005. [Online]. Available: http://link.springer.com/article/10.1007/s10664-005-1290-x

[119] S. Fürst, "AUTOSAR – A Worldwide Standard is on the Road," in *14th International VDI Congress Electronic Systems for Vehicles*, 2009. [Online]. Available: http://www.win.tue.nl/~mvdbrand/courses/sse/0910/AUTOSAR.pdf

[120] Springer, ""AUTOSAR has Become Mature and Accepted"," *ATZextra worldwide*, vol. 18, no. 9, pp. 13–15, 2013.

[121] G. Weiß, "Future vehicle software architectures - fraunhofer esk." [Online]. Available: http://www.esk.fraunhofer.de/en/projects/adaptives_bordnetz.html

[122] D. Durisic, M. Staron, M. Tichy, and J. Hansson, "Evolution of Long-Term Industrial Meta-Models – An Automotive Case Study of AUTOSAR," in *40th EUROMICRO Conference on Software Engineering and Advanced Applications*, 2014, pp. 141–148.

[123] G. M. K. Selim, S. Wang, J. R. Cordy, and J. Dingel, "Model transformations for migrating legacy deployment models in the automotive industry," *Software & Systems Modeling*, vol. 14, no. 1, pp. 365–381, 2013.

[124] G. Krdzalic and A. Driss, "Software Architecture Without Autosar," *Auto Tech Review*, vol. 3, no. 4, pp. 28–31, 2014.

[125] W. Mueller, "Engineering Standards beyond AUTOSAR," in *6th AUTOSAR Open Conference*, Munich, 2013. [Online]. Available: http://www.autosar.org/fileadmin/files/events/2013-11-13-6th-autosar-open/AUTOSAR_Engineering_Standards_Uni_Paderborn_Mueller.pdf

[126] C. Wohlin, M. Höst, and K. Henningsson, "Empirical research methods in software engineering," in *Empirical Methods and Studies in Software Engineering (ESERNET)*, ser. LNCS, vol. 2765, 2003, pp. 7–23.

[127] M. Ciolkowski, O. Laitenberger, S. Vegas, and S. Biffl, "Practical experiences in the design and conduct of surveys in empirical software engineering," in *Empirical Methods and Studies in Software Engineering (ESERNET)*, ser. LNCS, vol. 2765. Springer, 2003, pp. 104–128.

[128] J.-F. Salessy, "Overview on AUTOSAR Development," in *6th AUTOSAR Open Conference*, Munich, 2013. [Online]. Available: http://www.autosar.org/fileadmin/files/events/2013-11-13-6th-autosar-open/AUTOSAR_Keynote_PSA_Salessy.pdf

[129] D. Ameller, M. Galster, P. Avgeriou, and X. Franch, "A survey on quality attributes in service-based systems," *Software Quality Journal*, pp. 1–29, 2015.

[130] P. S. Medeiros Dos Santos and G. H. Travassos, "On the representation and aggregation of evidence in software engineering: A theory and belief-based perspective," *Electronic Notes in Theoretical Computer Science*, vol. 292, pp. 95–118, 2013.

[131] D. S. Cruzes and T. Dybå, "Synthesizing evidence in software engineering research," in *ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, no. 1, 2010.

[132] P. S. Medeiros Dos Santos, I. Nascimento, and G. H. Travassos, "A Computational Infrastructure for Research Synthesis in Software Engineering," in *Workshop en Ingeniería del Software Experimental (ESELAW)*, 2015, pp. 309–322.

[133] D. I. K. Sjøberg, T. Dybå, B. C. D. Anda, and J. E. Hannay, "Building theories in software engineering," in *Guide to advanced empirical software engineering*. Springer, 2008, pp. 312–336.

[134] C. F. Auerbach and L. B. Silverstein, *Qualitative data: An introduction to coding and analysis. Qualitative studies in psychology.* New York University Press, 2003.

[135] J. Mahoney, "A Tale of Two Cultures: Contrasting Quantitative and Qualitative Research," *Political Analysis*, vol. 14, no. 3, pp. 227–249, 2006.

[136] G. Shafer, *A Mathematical Theory of Evidence*. Princeton University Press, 1976. [Online]. Available: http://press.princeton.edu/titles/2439.html#reviews

[137] D. Atkins, D. Best, P. A. Briss, M. Eccles, Y. Falck-Ytter, S. Flottorp, G. H. Guyatt, R. T. Harbour, M. C. Haugh, D. Henry, S. Hill, R. Jaeschke, G. Leng, A. Liberati, N. Magrini, J. Mason, P. Middleton, J. Mrukowicz, D. O'Connell, A. D. Oxman, B. Phillips, H. J. Schünemann, T. T.-T. Edejer, H. Varonen, G. E. Vist, J. W. Williams, and S. Zaza, "Grading quality of evidence and strength of recommendations." *BMJ (Clinical research ed.)*, vol. 328, no. 7454, p. 1490, 2004.

[138] R. J. Wieringa, *Design Science Methodology for Information Systems and Software Engineering*. Springer, 2014.

[139] Atlassian, "JIRA: Issue and Project Tracking Software." [Online]. Available: https://www.atlassian.com/es/software/jira

[140] SonarSource, "Continuous Inspection of Code Quality." [Online]. Available: http://www.sonarsource.com/

[141] H. Erdogmus and J. Favaro, "The value proposition for agility–a dual perspective," 2012. [Online]. Available: http://www.infoq.com/presentations/Agility-Value

[142] Redmine, "A flexible project management web application." [Online]. Available: http://www.redmine.org/

[143] Jenkins, "An extendable open source continuous integration server." [Online]. Available: http://jenkins-ci.org/

[144] R. Van Solingen, "Measuring the ROI of software process improvement," *IEEE Software*, vol. 21, no. 3, pp. 32–38, 2004.

[145] J. Poulin, "The economics of product line development," *International Journal of Applied Software Technology*, vol. 3, pp. 15–28, 1997.

[146] A. Sandberg, L. Pareto, and T. Arts, "Agile collaborative research: Action principles for industry-academia collaboration," *IEEE Software*, vol. 28, no. 4, pp. 74–83, 2011.

[147] P. Runeson, "It takes two to tango–an experience report on industry–academia collaboration," in *IEEE 5th International Conference on Software Testing, Verification and Validation (ICST)*, 2012, pp. 872–877.

[148] M. Baldassarre, D. Caivano, and G. Visaggio, "Empirical studies for innovation dissemination: Ten years of experience," in *ACM International Conference on Evaluation and Assessment in Software Engineering (EASE)*, 2013, pp. 144–152.

[149] C. Wohlin, A. Aurum, L. Angelis, L. Phillips, Y. Dittrich, T. Gorschek, H. Grahn, K. Henningsson, S. Kagstrom, G. Low *et al.*, "The success factors powering industry-academia collaboration," *IEEE software*, vol. 29, no. 2, pp. 67–73, 2012.

[150] C. Wohlin, “Empirical software engineering research with industry: Top 10 challenges,” in *1st International Workshop on Conducting Empirical Studies in Industry (CESI)*, 2013, pp. 43–46.

[151] T. Gorschek, C. Wohlin, P. Carre, and S. Larsson, “A Model for Technology Transfer in Practice,” *IEEE Software*, vol. 23, no. 6, pp. 88–95, 2006.

[152] M. A. Babar, L. Bass, and I. Gorton, “Factors influencing industrial practices of software architecture evaluation: an empirical investigation,” in *3rd International Conference on Quality of Software Architectures (QoSA)*, ser. LNCS, 2007, vol. 4880, pp. 90–107.

[153] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*. Springer, 2012.

[154] W. C. Booth, G. G. Colomb, and J. M. Williams, *The craft of research*. University of Chicago Press, 2003.

Appendix A

Glossary

Empirical Software Engineering (ESE) Empirical research is a research using empirical evidence, i.e., gaining knowledge by means of direct and indirect observation or experience. Empirical software engineering focuses on how empirical studies and experiments in particular fit into the software engineering context [153].

Enterprise architecture Enterprise architecture is a discipline for proactively and holistically leading enterprise responses to disruptive forces by identifying and analyzing the execution of change toward desired business vision and outcomes. Enterprise architecture delivers value by presenting business and IT leaders with signature-ready recommendations for adjusting policies and projects to achieve target business outcomes that capitalize on relevant business disruptions. EA is used to steer decision making toward the evolution of the future state architecture[1].

Information Technology (IT) Information technology is the application of computers and telecommunications equipment to store, retrieve, transmit and manipulate data, often in the context of a business or other enterprise[2].

REARM (REference ARchitecture Model) REARM is a reuse-based economic model for SRAs. It can be used to perform cost-benefit analysis on the adoption of SRAs as a key asset for optimizing architectural decision-making.

[1]`http://www.gartner.com/it-glossary/enterprise-architecture-ea/`
[2]`https://en.wikipedia.org/wiki/Information_technology`

Reference architecture A Reference Architecture provides a prescriptive way (a template solution) for an architecture for a particular domain [47].

Research Question (RQ) Specifying the research question is the methodological point of departure of scholarly research in both the natural and social sciences. The research will answer the question posed. At an undergraduate level, the answer to the research question is the thesis statement. The answer to a research question will help address a "research problem" which is a problem "readers think is worth solving" [154].

Return-on-Investment (ROI) Measure of how much profit an investment earns computed by dividing net income by the assets used to generate it [25].

Systematic Literature Review (SLR) A Systematic Literature Review (SLR) is a formalized and repeatable process to document relevant knowledge on a specific subject area for interpreting all available research [65].

Software architecture The software architecture of a system is the set of structures needed to reason about the system, which comprise software elements, relations among them, and properties of both [1].

Software Engineering (SE) Software Engineering means application of a systematic, disciplined, quantifiable approach to development, operation and maintenance of software [153].

Software Reference Architecture (SRA) An architecture that encompasses the knowledge about how to design concrete architectures of systems of a given application [or technological] domain; therefore, it must address the business rules, architectural styles (sometimes also defined as architectural patterns that address quality attributes in the reference architecture), best practices of software development (for instance, architectural decisions, domain constraints, legislation, and standards), and the software elements that support development of systems for that domain. All of this must be supported by a unified, unambiguous, and widely understood domain terminology [2].

SRA acquistion/adoption SRA acquisiton or adoption is the strategic decision of an organization (taken by managers and software architects) to base the software development and maintenance of a family of software systems in a corporate SRA.

System architecture A system's architecture is a representation of a system in which there is a mapping of functionality onto hardware and software components, a mapping of the software architecture onto the hardware architecture, and a concern for the human interaction with these components [1].

Appendix B

Included Studies in the Systematic Review

Next, the list of included studies in the SLR of Chapter 3 is presented. The included studies are labeled as [*Si*], where *S* indicates that it is a primary study of the SLR, and *i* indicates its number.

List of Included Studies in the SLR of Chapter 3:

S1 Angelov, S., Grefen, P., Greefhorst, D.: A framework for analysis and design of software reference architectures. *Information and Software Technology* 54(4), 417-431 (2011)

S2 Angelov, S., Grefen, P., Greefhorst, D.: A classification of software reference architectures: Analyzing their success and effectiveness. WICSA/ECSA 2009, pp. 141-150. IEEE (2009)

S3 Angelov, S., Trienekens, J., Kusters, R.: Software reference architectures - Exploring their usage and design in practice. ECSA 2013. LNCS, vol. 7957, pp. 17-24. Springer (2013)

S4 Angelov, S., Trienekens, J., Grefen, P.: Towards a method for the evaluation of reference architectures: Experiences from a case. ECSA 2008. LNCS, vol. 5292, pp. 225-240. Springer (2008)

S5 Angelov, S., Hilliard, R.: Towards an Improved Stakeholder Management for Software Reference Architectures. ECSA 2014. LNCS, vol. 8627, pp. 90-97. Springer (2014)

S6 Astekin, M., Sozer, H.: Utilizing Clone Detection for Domain Analysis of Simulation Systems. WICSA/ECSA 2012, pp. 287-291. IEEE (2012)

S7 Buchgeher, G., Weinreich, R.: Towards continuous reference architecture conformance analysis. ECSA 2013. LNCS, vol. 7957, pp. 332-335. Springer (2013)

S8 Bueno, L.B.R., Romero Felizardo, K., Feitosa, D., Nakagawa, E.Y.: Reference models and reference architectures based on service-oriented architecture: A systematic review. ECSA 2010. LNCS, vol. 6285, pp. 360-367. Springer (2010)

S9 Cloutier, R., Muller, G., Verma, D., Nilchiani, R., Hole, E., Bone, M.: The concept of reference architectures. *Systems Engineering Journal* 13(1), 14-27 (2010)

S10 Deiters, C., Dohrmann, P., Herold, S., Rausch, A: Rule-Based Architectural Compliance Checks for Enterprise Architecture Management. EDOC 2009, pp. 183-192. IEEE (2009)

S11 Dillon, T.S., Wu, C., Chang, E.: Reference Architectural Styles for Service-Oriented Computing. IFIP NPC 2007. LNCS, vol. 4672, pp. 543-555. Springer (2007)

S12 Nguyen, D.N., Usbeck, K., Mongan, W.M., Cannon, C.T., Lass, R.N., Salvage, J., Regli, W.C., Mayk, I., Urness, T.: A Methodology for Developing an Agent Systems Reference Architecture. AOSE 2010. LNCS, vol. 6788, pp. 177-188. Springer (2011)

S13 Eklund, U., Jonsson, N., Eriksson, A., Bosch, J.: A reference architecture template for software-intensive embedded systems. WICSA/ECSA Companion Volume 2012, pp. 104-111. ACM (2012)

S14 Galster, M., Avgeriou, P.: Empirically-grounded reference architectures: A proposal. QoSA@CompArch 2011, pp. 153-158. ACM (2011)

S15 Graaf, B., Van Dijk, H., Van Deursen, A.: Evaluating an embedded software reference architecture. Industrial experience report. CSMR 2005, pp. 354-363. IEEE (2005)

S16 Guessi, M., Oliveira, L.B.R., Nakagawa, E.Y.: Representation of reference architectures: A Systematic Review. SEKE 2011, pp. 782-785. (2011)

S17 Guessi, M., Oquendo, F., Nakagawa, E.Y.: Variability viewpoint to describe reference architectures. VARSA@WICSA 2014, pp. 14:1-14:6. ACM (2014)

S18 Haft, M., Humm, B., Siedersleben, J.: The Architect's Dilemma - Will Reference Architectures Help? QoSA-SOQUA 2005. LNCS, vol. 3712, pp. 106-122. Springer (2005)

S19 Hassan, A.E., Holt, R.C.: A reference architecture for Web servers. Working Conference on Reverse Engineering, pp. 150-159. IEEE (2000)

S20 Heitmann, B., Kinsella, S., Hayes, C., Decker, S.: Implementing Semantic Web applications: Reference architecture and challenges. Workshop on Semantic Web Enabled Software Engineering (2009)

S21 Herold, S., Mair, M., Rausch, A., Schindler, I.: Checking conformance with reference architectures: A case study. EDOC 2013, pp. 71-80. IEEE (2013)

S22 Irlbeck, M., Bytschkow, D., Hackenberg, G., Koutsoumpas, V.: Towards a bottom-up development of reference architectures for smart energy systems. SE4SG@ICSE 2013, pp. 9-16. IEEE (2013)

S23 Kang, K.C., Kim, S., Lee, J., Kim, K., Shin, E., Huh, M.: FORM: A feature-oriented reuse method with domain-specific reference architectures. *Annals of Software Engineering* 5:143 (1998)

S24 Lind, K., Heldal, R.: Automotive system development using reference architectures. SEW 2012, pp. 42-51. IEEE (2012)

S25 Martínez-Fernández, S., Ayala, C., Franch, X. and Marques, H.M., Ameller, D.: A framework for software reference architecture analysis and review. ESELAW@CIbSE 2013, pp. 89-102 (2013)

S26 Martínez-Fernández, S., Ayala, C.P., Franch, X., Marques, H.M.: REARM: A reuse-based economic model for software reference architectures. ICSR 2013. LNCS, vol. 7925, pp. 97-112. Springer (2013)

S27 Martínez-Fernández, S., Ayala, C.P., Franch, X., Marques, H.M.: Benefits and drawbacks of reference architectures. ECSA 2013. LNCS, vol. 7957, pp. 307-310. Springer (2013)

S28 Martínez-Fernández, S., Ayala, C.P., Franch, X., Marques, H.M.: Artifacts of Software Reference Architecture: A Case Study. EASE 2014, pp. 42:1-42:10. ACM (2014)

S29 Martínez-Fernández, S., Ayala, C.P., Franch, X., Marques, H.M.: Towards Guidelines for Building a Business Case and Gathering Evidence of Software Reference Architectures in Industry. *Journal of Software Engineering Research and Development* 2(7) (2014)

S30 Meldal, S., Luckham, D.C.: NSA's MISSI Reference Architecture – Moving from Prose to Precise Specifications. RTSE 1997. LNCS, vol. 1526, pp, 293-329. Springer (1998)

S31 Miksovic, C., Zimmermann, O.: Architecturally Significant Requirements, Reference Architecture, and Metamodel for Knowledge Management in Information Technology Services. WICSA 2011, pp. 270-279. IEEE (2011)

S32 Muller, G.: Right sizing reference architectures; How to provide specific guidance with limited information. INCOSE 2008 (2008)

S33 Muller, G., van de Laar, P.: Researching Reference Architectures. In: van der Laar, P., Punter, T. (eds.) *Views on Evolvability of Embedded Systems*, pp. 107-119. Springer (2011)

S34 Nakagawa, E.Y., Antonino, P.O., Becker, M.: Reference architecture and product line architecture: A subtle but critical difference. ECSA 2011. LNCS, vol. 6903, pp. 207-211. Springer (2011)

S35 Nakagawa, E.Y., Barbosa, E.F., Maldonado, J.C.: Exploring ontologies to support the establishment of reference architectures: An example on software testing. WICSA/ECSA 2009, pp. 249-252. IEEE (2009)

S36 Nakagawa, E.Y., Becker, M., Maldonado, J.C.: A knowledge-based framework for reference architectures. SAC 2011, pp. 1197-1202. ACM (2011)

S37 Nakagawa, E.Y., Guessi, M., Maldonado, J.C., Feitosa, D., Oquendo, F.: Consolidating a Process for the Design, Representation, and Evaluation of Reference Architectures. WICSA 2014, pp. 143-152. IEEE (2014)

S38 Nakagawa, E.Y., Maldonado, J.C.: Reference architecture knowledge representation: An experience. SHARK@ICSE 2008, pp. 51-54. ACM (2008)

S39 Nakagawa, E.Y., Maldonado, J.C.: Towards the Open Source Reference Architectures. SBCARS 2011, pp. 61-70. IEEE (2011)

S40 Nakagawa, E.Y., Oliveira, L.B.R.: Using systematic review to elicit requirements of reference architectures. WER@CIbSE (2011)

S41 Nakagawa, E.Y., Oquendo, F., Becker, M.: RAModel: A Reference Model for Reference Architectures. WICSA/ECSA 2012, pp. 297-301. IEEE (2012)

S42 Panunzio, M., Vardanega, T.: On software reference architectures and their application to the space domain. ICSR 2013. LNCS, vol. 7925, pp. 144-159. Springer (2013)

S43 Ramirez-Cadena, M.J.: Low Cost Educational Technology Based on Open System Reference Architecture for Engineering Courses. BASYS 2002, pp. 525-532. Kluwer (2002)

S44 Regli, W.C., Mayk, I, Cannon, C.T., Kopena, J.B., Lass, R.N., Mongan, W.M., Nguyen, D.N., Salvage, J.K., Sultanik, E.A, Usbeck, K.: Development and Specification of a Reference Architecture for Agent-Based Systems. *IEEE Transactions on Systems, Man, and Cybernetics Systems* 44(2), 146-161 (2014)

S45 Rittenbach, T., Satake, H., Redding, E., Perry, K., Thawani, M., Dietrich, C., Thandee, R.: GRA model driven design process. MILCOM 2010, pp. 1151-1156. IEEE (2010)

S46 Rittenbach, T., Kovarik Jr., V.J., Krause-Aiguier, R., Stewart, C.: Complex terminal systems design: Minimizing time to deployment. MILCOM 2010, pp. 656-661. IEEE (2010)

S47 Tekinerdogan, B., Özturk, K., Dogru, A.: Modeling and Reasoning about Design Alternatives of Software as a Service Architectures. WICSA 2011, pp. 312-319. IEEE (2011)

S48 Weinreich, R., Buchgeher, G.: Automatic reference architecture conformance checking for SOA-Based software systems. WICSA 2014, pp. 95-104. IEEE (2014)

S49 Wu, R.R., Zhang, Y.Y.: A CAPP framework and its methodology. *The International Journal of Advanced Manufacturing Technology* 14(4), 255-260 (1998)

S50 Zhou, N., Zhang, L.J.: Analytic Architecture Assessment in SOA Solution Design and its Engineering Application. ICWS 2009, pp. 807-814. IEEE (2009)

S51 Zhu, L. and Staples, M. and Jeffery, R.: Scaling up software architecture evaluation processes. ICSP 2008. LNCS, vol. 5007, pp. 112-122. Springer (2008)

S52 Zimmermann, O., Kopp, P., Pappe, S.: Architectural Knowledge in an SOA Infrastructure Reference Architecture. In: Ali Babar, M., Dingsøyr, T., Lago, P., van Vliet, H. (eds.) *Software Architecture Knowledge Management*. Springer (2009)

Appendix C

Materials for Gathering Evidence of SRAs

In this appendix, we present the set of relevant aspects for SRAs engineering, and a template survey to gather evidence of such aspects from different stakeholders.

C.1 Relevant Aspects of SRAs

Table C.1: Summary of relevant aspects for SRA engineering.

Aspect	Description of the SRA Aspect
1	Overview and classification of an SRA
2	Requirements and quality attributes analysis
3	Architectural knowledge and decisions
4	Supportive technologies
5	Business qualities and architecture competence

C.2 Template Survey for Gathering Evidence of SRAs

The template survey is divided in two parts: questions about personal data, project, and experience; and, questions about the relevant aspects of SRAs.

For each question, we indicate for which stakeholders they are interesting: software architects (SA), architecture developers (AD), application builders (AB), or project business leaders (PBL).

C.2.1 Questions about Personal Data, Project, and Experience

- Personal data
  - Name and surname *(SA, AD, AB, PBL)*
  - E-mail *(SA, AD, AB, PBL)*
  - Phone *(SA, AD, AB, PBL)*
  - Level of education *(SA, AD, AB, PBL)*
  - Education area *(SA, AD, AB, PBL)*
  - Certificates *(SA, AD, AB)*
- About the SRA and concrete architecture projects
  - Describe briefly the SRA project *(SA, AD)*
  - Describe briefly the concrete architecture project based on the SRA *(AB)*
  - Name of the SRA of the project *(SA, AD, AB, PBL)*
  - Role(s) in the project *(SA, AD, AB)*
  - Number and role of participants *(SA)*
  - With how many participants did you interact during the development of the software? *(AD, AB)*
  - SRA project initial development phase duration (if it applies) *(SA, AD, PBL)*
  - SRA project maintenance duration (if it applies) *(SA, AD, PBL)*
  - How many participants of the project had experience in SRAs? *(SA)*
  - Did you have previous experience in SRAs before the project? *(SA, AD)*

  - Did you have previous experience in developing SRA-based applications before this project? *(AB)*
  - Total effort of the SRA project (people/month) *(SA)*

- Experience in the organization
  - Job position in the organization (when you participated in this project) *(SA, AD, AB)*
  - Years in this job position (when you participated in this project) *(SA, AD, AB)*
  - Years in the organization (when you participated in this project) *(SA, AD, AB)*
  - Experience in project management (when you participated in this project) *(SA)*

C.2.2 Questions about the Relevant Aspects of SRAs

- Aspect 1: Overview and classification of an SRA
  - What do you understand by SRA? *(SA)*
  - Could you give a short description of the functionalities of the SRA project? *(SA)*
  - Which was the problematic that motivated the SRA project? *(SA)*
  - What was the objective of the SRA project? *(SA)*
  - How was the relationship among the stakeholders during the design and implementation of the SRA? *(SA)*
  - How was the contact among the stakeholders after the first release of the SRA? *(SA)*
  - Does the SRA take into account the organization's business processes? *(SA)*
  - What artifacts or deliverables have been produced in the SRA project? How did they help? *(SA)*
  - Is the SRA general or specific for a domain? Could it be used for other domains? *(SA, AD, AB)*
  - During the design of the SRA, have you reused some existing architectural knowledge or software element? *(SA)*

  - Does your SRA offer reusable modules for transversal services? *(SA, AD, AB)*
  - Are SRA-based applications aligned with business needs? *(PBL)*
  - Is the quality of the SRA deliverables good? *(PBL)*
  - Has the integration been performed easily? *(PBL)*

- Aspect 2: Requirements and quality attributes analysis
  - Which were the main functional requirements of the SRA? *(SA)*
  - Which were the main non-functional requirements (i.e., quality attributes) of the SRA? Could you give an example important for this SRA project? *(SA)*
  - Which were the main constraints of the SRA? Could you give an example important for this SRA project? *(SA)*
  - How were requirements elicited in this SRA project? *(SA)*
  - How were requirements documented in this SRA project? *(SA)*
  - How were requirements validated in this SRA project? *(SA)*
  - Who defined the requirements? *(PBL)*
  - Can you give an example of requirement? *(PBL)*
  - Did you follow some pattern for defining the requirements? *(PBL)*
  - Were requirements expressed in sufficient detail to discern their satisfaction? *(PBL)*
  - Were the requirements met successfully? *(PBL)*

- Aspect 3: Architectural knowledge and decisions
  - How did you design the SRA? *(SA)*
  - How much freedom did you have while taking architectural decisions? *(SA)*
  - Did you have to use some technology (constraint)? *(SA)*
  - What was chosen before in this project: SRA design or technological framework? *(SA)*
  - Could you give us an example of an architectural decision and its relation to quality attributes of this SRA project? *(SA)*

  - Were architectural decisions documented? Were these architectural decisions predefined? *(SA)*
  - Do you have a global vision of the SRA? *(AD, AB)*

- Aspect 4: Supportive technologies
  - About the software development methodology
    - Describe the used methodology and processes for this SRA project. Explain their stages. *(SA, AD, AB)*
    - Which practices or methods were followed in this SRA project in relation to testing? *(SA, AD, AB)*
    - Besides the possible documentation of requirements and architectural decisions, which documentation was done in this SRA project? *(SA)*
    - Which documentation was done in this SRA project? *(AD, AB)*
    - Why have you used the software development methodology, the testing methods, and the documentation aforementioned? *(AD, AB)*
    - Which input documentation did you receive to start coding? *(AD, AB)*
    - What liberty grade (restrictions about technologies, libraries, way of coding...) have you had while coding? *(AD, AB)*
    - Have you used best practices? *(AD, AB)*
  - About technologies and tools
    - Which integrated development environments (IDEs) were used in this SRA project? *(SA, AD, AB)*
    - What tools were used for project management? *(SA)*
    - How was continuous integration performed? Have you used some tool for that? *(SA, AD, AB)*
    - Does your SRA include a monitoring tool for applications? *(SA, AD, AB)*
    - Have you used any tool to generate code automatically? *(SA, AD, AB)*
    - Do you think that any development or coding task can be done automatically (totally or partially)? *(SA, AD, AB)*
    - Which programming languages did you use in this project? *(AD, AB)*

    - Which technologies were used for presentation in this project? *(AD, AB)*
    - Which technologies were used for the development of services in this project? *(AD, AB)*
    - Which technologies were used for the development of business processes in this project? *(AD, AB)*
    - Which technologies were used for interoperability and integration in this project? *(AD, AB)*
    - Which technologies were used for the management of data in this project? *(AD, AB)*
    - Which database management systems were used in this project? *(AD, AB)*
    - Which application servers were used in this project? *(AD, AB)*
    - Why have you used the aforementioned tools and technologies? *(AD, AB)*
    - Have you used any other important tool in this project? *(SA, AD, AB)*
    - Do you consider that the usage of some technologies and tools have caused any limitation? *(AD, AB)*

- Aspect 5: Business qualities and architecture competence
  - Architecture competence of the organization that use the SRA
    - To develop new SRA-based applications, is recommended any development methodology? *(SA)*
    - To develop new SRA-based applications, is recommended any method in relation to testing? *(SA)*
    - To develop new SRA-based applications, are recommended good documentation practices? *(SA)*
    - What are the benefits of using the SRA? *(SA, AD, AB)*
    - What are the drawbacks of using the SRA? *(SA, AD, AB)*
    - How was conducted the training for the organization in order to use the SRA? *(SA, AD, AB)*
    - Did the use of the SRA cause any organizational change in the organization? *(SA)*
    - How does the use of the SRA reduce the time-to-market of SRA-based applications in the organization? *(SA)*

    * What types of non-functional requirements are reinforced because of using the SRA in applications? *(SA, AD, AB)*
    * Which other benefits or problems developers might experience while using the SRA? *(SA)*
    * To sum up, what conclusions do you draw from the facilities provided by the SRA for the organization? *(SA, AD, AB)*
    * Does the SRA make easier the collaboration between IT and business teams? *(PBL)*
    * Has the development time been reduced because of using a reference model (comparing to other SRA projects)? *(PBL)*
    * Has the SRA improved the quality of the applications? *(PBL)*
    * Has SRA-based applications' price been reduced because of using a reference model (comparing to other SRA projects)? *(PBL)*
    * How many incidences have you experienced with the applications? *(PBL)*
    * Which is the most relevant benefit from using an SRA? *(PBL)*
    * Would you like to change something in future versions of the SRA? *(PBL)*
    * Indicate your overall satisfaction with the SRA and the functioning of this project *(PBL)*
    * Indicate if you have met all expectations regarding the use of the SRA (e.g., time-to-market, cost ...) *(PBL)*
    * Do you consider successful the use of the SRA? Would you use it again in the future? *(PBL)*
  - Architecture competence of the SRA vendor (if any)
    * Do you think that a common repository for all SRA for reusing services would be useful for the vendor organization? *(AD)*
    * Is the SRA based on a reference model or any other existing architectural knowledge and software components in the vendor organization? *(SA, AD)*
    * What do you think should be replaced, included or updated in prospective versions of the SRA? *(SA, AD, AB)*
    * To sum up, what conclusions do you draw from the facilities provided by the SRA for the vendor organization? *(SA, AD)*

Appendix D

Materials for Building the Business Case for SRAs

In this appendix, we present the questions for software architects to check existing value-driven data in SRA projects, and the materials of the REARM economic model.

D.1 Questions to Check Existing Value-Driven Data in SRA Projects

- Questions about data available for the SRA:
  - Is it possible to access to a code quality management tool (e.g., Sonar) used in the SRA project? If not, is it possible to install Sonar to take metrics from the source code of the SRA?
  - Would it be possible to estimate the degree of reuse in each of the modules of the SRA with respect to the reference model? (optional)
  - Did the SRA stakeholders track the time spent on each task in the development of the SRA? If so, with which granularity?
  - Is there data about personnel and time invested in maintaining the SRA? If so, with which granularity?
  - Would it be possible to give an estimate of the additional training time that an application builder needs to use the SRA?

  - Would it be possible to specify a standard hourly rate of performing tasks on the SRA?

- Questions about data available for the applications based on the SRA:
  - How many applications are (or will be) based on the SRA? Make a list of the applications with a contact person.
  - Do you have an overview of the development of applications based on the SRA? If so, go on. If not, we will contact the person who you indicated in the previous answer.
  - For which of the above applications is it possible to know which SRA modules have been reused and the degree of reuse?
  - Is additional effort needed to reuse SRA modules?
  - Can you estimate how long it would take to develop SRA modules instead of reusing them?
  - Is it possible to access to a code quality management tool (e.g., Sonar) used for the SRA-based applications? If not, is it possible to install Sonar to take metrics from the source code of the applications?
  - Please, indicate the generic characteristics (e.g., reuse percentage of SRA modules, size of applications) of three ideal types of SRA-based applications with low, medium and high complexity.
  - Did the SRA stakeholders track the time spent on each task in the development of the SRA-based applications? If so, with which granularity?
  - Is there data about personnel and time invested in maintaining the SRA-based applications? If so, with which granularity?
  - Would it be possible to specify a standard hourly rate of performing tasks on the SRA-based applications?
  - Have you done any comparison between the costs and the benefits between SRA-based applications and ad-hoc applications?
  - Currently, are there indicators or metrics to evaluate the improvement of the quality attributes in the applications because of the SRA usage?

- Questions for adding comments and propose metrics to calculate the ROI of the SRA:
  - In addition to the information discussed above, do you think that there is other available information to evaluate how the SRA affects the applications development?
  - Do you think that other metric, not mentioned above, could be useful to calculate the ROI of building applications based on the SRA?
  - Our economic model to calculate the ROI of an SRA is based on the reuse and maintenance of code. Do you think there are other quality attributes or important factors for evaluating an SRA?
  - Before sending the survey, would you like to add any comments that may help to understand the context of your answers?

D.2 Materials of REARM

REARM consists of five steps:

1. Extracting ten basic parameters to feed the cost-benefit factors.
2. Calculating cost-benefit factors to calculate the ROI of adopting an SRA in an organization.
3. Extracting business case parameters.
4. Calculating the ROI: useful scenarios for the organization and its decision making.
5. Consideration of unquantifiable benefits, and uncertainties and risk.

Next, we summarize the support materials to execute such steps.

D.2.1 Ten Basic Parameters to Feed the Cost-Benefit Factors

Table D.1: Basic parameters in order to feed the factors of REARM.

	Description of the parameters (adapted for the SRA context)
RCR	*Relative Cost of Reuse*: effort that it takes to reuse a component without modification versus writing it new one-at-a-time [87]
RCWR	*Relative Cost of Writing for Reuse*: effort that it takes to write a reusable component versus writing it for one-time use only [87]
ER	*Error Rate*: the historical error rate in new software developed by your organization, in errors per thousand lines of code [87]
EC	*Error Cost*: your organization's historical cost to fix errors after releasing new software to the customer, in euros per error [87]
NMSI	*New Module Source Instruction*: the LOC that the changed or new module has, which can be the average of previous ones
PC	*Propagation Cost*: the percentage of code affected in the SRA when performing evolutions (i.e., changing modules) [97]
CPKL	*Cost per KLOC*: the historical cost to develop a KLOC of new software in your organization [87]
USI	*Unique Source Instructions*: the amount of unique software (i.e., not reused) that was written or modified for an application
RSI	*Reused Source Instructions*: it is the total LOC of the SRA's modules that are reused in an application. It supports variability. In other words, reuse of SRA might not be complete but partial, since different applications can reused different SRA's modules. Therefore RSI depend on each application [87].
TSI	*Total Source Instructions*: it is the total LOC of the SRA that can be reused [87].

D.2.2 Cost-Benefit Factors to Calculate the ROI of Adopting an SRA in an Organization

Table D.2: Cost-benefit factors to calculate the ROI of adopting an SRA in an organization.

	Description of the cost-benefit factors (adapted for the SRA context)
DCA	*Development Cost Avoidance*: the benefits from reusing SRA's modules [87] DCA = RSI * (1-RCR) * CPKL
CSWD	*Common Software Development Costs*: the costs to develop the SRA [145] CSWD = RCWR * TSI * CPKL
UDC	*Unique Development Costs*: the costs to develop the unique part of an application UDC = USI*CPKL
SCA	*Service Cost Avoidance*: benefits from maintaining only once SRA's modules [87] SCA = RSI * ER * EC
CSWS	*Common Software Maintenance Costs*: cost of fixing bugs in reusable modules [145] CSWS = TSI * ER * EC
CSWE	*Common Software Evolution Costs*: the costs of changing or adding a new functionality and maintaining it to the SRA CSWE = evolution development + evolution maintenance + propagation = (NMSI*RCWR*CPKL)+(NMSI*ER*EC)+(TSI*CPKL*PC)

D.2.3 Business Case Parameters

- Business Case Parameters:
  - Years of the SRA program.
  - Number and size of SRA-based applications.
  - Number of evolved SRA modules.
  - Hourly rate for software architects and for application builders.

D.2.4 Calculating the ROI: Useful Scenarios for the Organization and its Decision Making

We suggest using these two scenarios to make a business case for calculating the ROI of building an SRA vs. building the applications independently:

- ROI calculation: Is it worth to invest on the adoption of an SRA?
- Payback analysis: How many instantiations are necessary before savings pay off for the up-front investment?

For both of them, we need to calculate the ROI:

$$ROI = \frac{Benefits - Costs}{Costs} \tag{D.1}$$

Putting everything together, given a number n of applications built in top of the SRA, and a number m of SRA modules changed as it evolves, the benefits and costs of adopting an SRA are defined as:

$$Benefits = \sum_{i=1}^{n}(DCA_i + SCA_i) \tag{D.2}$$

$$Costs = CSWD + CSWS + \sum_{i=1}^{n} UDC_i + \sum_{j=1}^{m} CSWE_j \tag{D.3}$$

Putting everything together to calculate the ROI:

$$ROI = \frac{[\sum_{i=1}^{n}(DCA_i + SCA_i)] - [CSWD + CSWS + \sum_{i=1}^{n} UDC_i + \sum_{j=1}^{m} CSWE_j]}{CSWD + CSWS + \sum_{i=1}^{n} UDC_i + \sum_{j=1}^{m} CSWE_j} \tag{D.4}$$

Table D.3 shows an example of business case and how to calculate the cost and benefits for three years since the SRA adoption. The parameters n_1, n_2, n_3 indicate the number of applications developed per year respectively, and m the number of evolved modules.

Table D.3: Example of design of a business case with the cost-benefit factors of the model.

	Year 1	**Year 2**	**Year 3**
Total benefit	n_1*(DCA+SCA)	n_2*(DCA+SCA)	n_3*(DCA+SCA)
Total cost	CSWD+ n_1*UDC+CSWS*$^1/_5$	n_2*UDC+ CSWS*$^2/_5$+m*CSWE	n_3*UDC+ CSWS*$^2/_5$+m*CSWE

D.2.5 Adding Unquantifiable Benefits, Uncertainties and Risks

As Boehm points out [82], two additional factors may be important in business case analysis: unquantifiable benefits, and uncertainties and risk.

First, the economic model that we propose promotes benefits in reusability and maintainability. However, other quality attributes, such as security, could be as relevant as those for this analysis, even when they may be difficult to quantify. These other benefits should also been taken into account when adopting and SRA. Unquantifiable benefits are also considered as “flexibility” in TEI [84], the economic model of Forrester.

Second, to adjust cost and benefits to risk, they can be multiplied by percentages that generally increase the costs and reduce the benefits (assuming the worst case). For instance, TEI proposes to multiple costs by values that range from 98% to 150% and benefits by values between 50% and 110%.