Artifacts of Software Reference Architectures: A Case Study

Silverio Martínez-Fernández, Claudia Ayala, Xavier Franch
Universitat Politècnica de Catalunya, BarcelonaTech
Jordi Girona, 1-3, 08034, Barcelona (Spain)
{smartinez, cayala, franch}@essi.upc.edu

Helena Martins Marques
everis
Diagonal, 605, 08028, Barcelona (Spain)
hmartinm@everis.com

ABSTRACT

Context: Software reference architectures (SRA) have emerged as an approach to systematically reuse architectural knowledge and software elements in the development of software systems. Over the last years, research has been conducted to uncover the artifacts that SRAs provide in order to build software systems. However, empirical studies have not focused on providing industrial evidence about such artifacts. **Aim**: This paper investigates which artifacts constitute an SRA, how SRAs are designed, the potential reuse of SRA's artifacts, and how they are used in practice. **Method**: The study consists of a case study made in collaboration with a multinational consulting company that designs SRAs for diverse client organizations. A total of nine European client organizations that use an SRA participated in the study. We analyzed available documentation and contacted 28 practitioners. **Results**: In the nine analyzed projects, we observed that the artifacts that constitute an SRA are mainly software elements, guidelines and documentation. The design and implementation of SRAs are influenced by the reuse of artifacts from previous software system development and experiences, and the reuse of an SRA across different business domains may be possible when they are platform-oriented. Regarding SRAs usage, we observed that conformance checking is seldom performed. **Conclusions**: This study reports artifacts of SRAs as stated by practitioners in order to help software architects and scientists in the inception, design, and application of SRAs.

Categories and Subject Descriptors

D.2.11 [**Software Engineering**]: Software Architectures – domain-specific architectures

General Terms

Design, Experimentation, Standardization.

Keywords

Software reference architecture, software reuse, empirical software engineering, case study

1. INTRODUCTION

"A Reference Architecture (RA) 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" [19]. Devising an RA is complex, and may involve different architecture disciplines. Bass et al. identify three disciplines: software architecture, system architecture and enterprise architecture [5]. In this work, we focus on RAs for software architectures, namely Software Reference Architecture (SRA) [2]. "An SRA is a generic architecture for a class of systems that is used as a foundation for the design of concrete architectures from this class" [2].

Hence, the purpose of SRAs is to serve as guidance for the standardization, development, and evolution of software systems in a cost-effective manner [20][21][29]. A recent Gartner report claims that an SRA "enables greatly increased speed and reduced operational expenses as well as quality improvements due to lowered complexity, greater investment and greater reuse" [36]. Thus, "IT organizations that lack architecture and configuration standards […] have higher costs and less agility that those with enforced standards" [36]. According to their expected benefits [22], SRAs have become widely studied and used [1].

However, several works have stated recently that the perspective on what constitutes an SRA still varies significantly [7][8]. Also, a recent literature review about evidence in software architecture, in which only two papers were about SRAs, shows that there is limited knowledge about current industrial SRA usage practices [32]. As a result, practitioners usually find the current SRA-related literature scarce and abstract [1], limiting the industrial uptake of research results in the field. In this context, we believe that the research community must both clarify the artifacts that constitute an SRA and understand what the current industrial SRA-related practices are to envisage more realistic and effective solutions [15]. This would enable practitioners to fully exploit the benefits of SRA adoption and usage. Thus, our research goal is:

Which are the <u>artifacts</u> that compose an SRA and how such artifacts are <u>designed</u>, <u>reused</u>, and <u>used</u>?

With this goal in mind, we planned and designed an exploratory case study in the context of a multinational consulting company called *everis* in order to understand its SRA practices and gather observations. Despite this study only focuses in the *everis* context, it aims to provide evidence that might contribute to an empirical basis towards further understanding SRA practices in similar contexts (cf. [10][16][17][18][24][28]), and therefore the alignment of research endeavors with real industrial needs.

The paper is organized as follows. Section 2 provides a brief background to SRA infrastructure, design, reuse and use. Section 3 presents the methodology and details of this study. Section 4 presents the results obtained, while Section 5 provides an in-depth discussion of the findings. Section 6 summarizes the conclusions.

2. BACKGROUND
This section provides a brief background on the aspects of SRAs which are the focus of this study: the artifacts that constitute an SRA, and the design, the reuse and the usage of SRAs. In the discussions, this background is compared with our results.

2.1 SRA Infrastructure
The artifacts or constituent parts of SRAs have received little attention [2]. However, a few works in the RA literature (not only limited to the software architecture dimension) describe the artifacts that could be used to build software systems based on an RA (these artifacts are also known as infrastructure [29]). Next, we show how diverse authors state significantly different views about the artifacts or deliverables of an SRA. We also explore artifacts of RAs, to study when they are also present in SRAs.

First, Angelov et al. distinguish *components and connectors*, *interfaces*, *protocols*, *algorithms*, and *policies and guidelines* [1]. They identified these artifacts after analyzing 24 SRAs.

Second, Galster et al. indicate that the basic structure of an RA consists of its *common building blocks* (i.e., common stakeholders, views, model kinds) according to ISO/IEC 42010 [13]. Besides, they note the importance of the *documentation* of these RA building blocks. The authors are based on their own experience.

Third, Nakagawa et al. [29] indicate that an RA 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 RA; and *guidelines*, which indicate how to apply best practices. They studied the literature of RAs, 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 RAs. They indicate as common elements of RAs: *business purpose*, *standards*, *guidance for implementing*, and *roadmap* [8]. The two latter ones are artifacts of an RA infrastructure. Their vision comes from the system architecture discipline.

Fifth, Herold et al. identify the following artifacts in the RA of a German public administration: *reusable components of software*, *operation platform*, *methodology*, *tools*, *blue-line prints* [17].

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

- **Software elements** [29] (i.e., implementation of components and connectors [1][17]).
- **Best practices and guidelines** [29] (i.e., policies [1], guidance for implementing [8], methodology, tools and blue-line prints [17]).
- **General structure** [29] (represented by documentation of common building blocks [13], architectural knowledge and roadmaps [8]).
- **Hardware elements** [29] (i.e., operating platform [17]).
- **Others**: interfaces, protocols, algorithms [1].

2.2 Reuse in SRA Design
As the definition of an RA shows, its artifacts are often harvested from previous projects [19]. Hence, the design of an SRA usually involves reusing the essential of existing software architectures [13]. Reusable software assets are not limited to code [24], 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 [6], and implies the application of proven components and architectural styles that induce specific quality attributes [24].

However, there is little evidence on what is reused in order to design SRA. As a consequence, several problems arise when reusing assets in the design of SRAs: (1) SRAs are usually not designed in a systematic manner with repeatable steps [13]; (2) 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 [12]; and (3) it requires a high degree of communication between people, especially when the knowledge is shared implicitly [18].

2.3 Reuse of SRA Across Business Domains
SRAs are designed to be used in a given domain. SRAs may be designed for two types of domains: "platform-oriented" [29] or technological when they are related to a specific architectural style or technology (e.g., service-oriented architectures [37]); and business-oriented when they address a specific application domain (e.g., automotive [3][10], banking [28], space [24][30]).

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 [25] and private partnerships [3]. Still, there is no evidence when an SRA can be reused across domains.

2.4 SRA Usage
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 [7][17][31]. 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 [7].

Although automatic rule-based conformance checking has been explored for SRAs [7][17], there are not empirical studies that investigate how the aforementioned roles are adopted for SRAs.

3. RESEARCH METHODOLOGY
This study has been planned and designed in the context of *everis*, a multinational consulting company with which the GESSI research group at UPC is currently running a long-term collaboration aimed at supporting innovation dissemination [3] and the improvement of *everis*' SRA-related practices. The goal is to get evidence on the artifacts that compose an SRA, and the role that these artifacts play in the design, reuse and use of SRAs. Thus, *everis* is utterly interested and willing to participate in the study.

3.1 Research Setting
everis offers solutions for big organizations from diverse business domains (e.g., banks, insurance companies, public administration, utilities, and industrial organizations) that need to manage a wide portfolio of complex and business-critical applications. Given the complexity of these resulting applications, which integrate bespoke software with off-the-shelf components, they demand high-quality software architectures. The strategy adopted by *everis* to

reach this quality is to foster the adoption of SRAs in these client organizations as a baseline for developing standardized concrete software architectures for new applications. In summary, *everis* supports its client organizations to design and develop their own SRA, and to build applications on top of such SRA.

There are three types of key stakeholders related to the design and use of SRAs: (1) **software architects** that cooperatively work to figure out an SRA based on the *everis'* reference model (i.e., a platform-oriented design solution from previous SRA projects) to accomplish the desired quality attributes and architecturally-significant requirements of the client organization. We differentiate between a reference model and an SRA because an SRA includes software elements that cooperatively implement the functionality of the reference model [5]; (2) **architecture developers** that are responsible for coding, maintaining, integrating, testing and documenting the software components of the SRA designed by everis for the client organization; and (3) **application builders** that take the SRA reusable components and instantiate them to build concrete software architectures for the client organization's applications. Figure 1 shows how software architects and architecture developers form the **SRA team**, whilst application builders form **concrete software architecture teams** (one for every application) [20]. Both teams compose an SRA project and may combine professionals from *everis* and from the client organization.

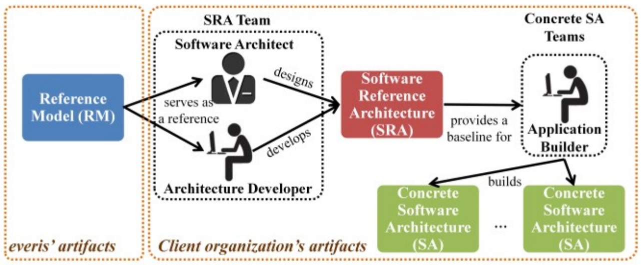

Figure 1. Key stakeholders in *everis*' SRA projects.

In order to design and develop the SRA for their client organizations, SRA teams are suggested to use a corporate reference model that helps them to reuse knowledge and homogenize the conception of SRAs. This is a conceptual model divided into four different views: (a) "execution architecture", which details the software elements that an SRA could provide; (b) "development architecture" which details the need of guidelines to facilitate the development of software systems and documentation about the design; (c) "technical architecture", which generally states the hardware elements and application servers (e.g., WebLogic, Tomcat) for deploying applications, and (d) "architecture of operation", which consists of the procedures to guarantee that applications are working (e.g., monitoring tools).

In this context, and with our research goal in mind, four GESSI members and three *everis* managers considered the study of the *everis*' SRA-related practices in some *everis'* client organizations. We inquired the Research Questions (RQ) below.

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 **RQ1: 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 **RQ2: 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 **RQ3: 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 **RQ4: 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.

3.2 Research Design

In line with the exploratory nature of our RQs, we decided to use a qualitative case study approach based on the analysis of nine SRA projects designed by *everis* for nine different client organizations. Considering several projects in different client organizations allowed us a better interpretation and assessment of the SRA design practices and usage. It would otherwise have been very difficult to interpret certain decisions or influential factors related to the nature of the projects or their context. Baldasasarre et al. state that explorative case study is the most efficient study for innovation dissemination because it involves stakeholders in the innovation process and knowledge diffusion [4].

3.2.1 Sampling and data collection

As stated above, the target population of our study was *everis*' SRA projects. The authors and another 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 roles (i.e., software architects, architecture developers and application builders). Once the projects were selected, the *everis*' managers provided us with their detailed information (explicitly, the so-called SRA project card with summarized description, documentation, and metrics about the invested effort) and contacted with potential participants for agreeing on their participation. Whenever we needed clarification, *everis* managers put us in contact with the corresponding project technical manager or suitable people to handle our questions. Thus, when required, we held some informal meetings with them.

We finally ended up with 28 people from *everis* that participated in the selected projects. In one project we did not manage at the end to contact any application builder, whilst in other two projects, we had two of them (see Table 1). The high response rate shows the interest of *everis* in the study results. Table 1 gives information about the SRA projects for the client organizations.

In line with the objective of each RQ, it was clear that some stakeholders were related with specific RQs. We designed different data collection instruments (available in [23]) based on this assumption, and designed and piloted them following the guidelines stated in [33][34], previous experience performing international surveys [1], and the corresponding literature background.

Table 1. Overview of the selected *everis*' SRA projects.

Id org.	Partici-pants			Main domain	SRA Target	SRA Project Type	Approx. Effort (in hours)
	SA	AD	AB				
A	1	1	1	Industry	Web-based applications to allow vendors updating information about clients in a department store.	Des. P.O.	≈5,000
B	1	1	1	Banking	Multi-platform applications that are fast, satisfy practices of the market and support transaction processing.	Des., Evol. A.D.	≈97,000
C	1	1	1	Banking	Multi-platform applications of a bank.	Des., Evol. P.O.	≈37,000
D	1	1	1	Insurance	Applications that satisfy internal request for proposals.	Des., Evol. P.O.	≈29,000
E	1	1	1	Public sector	Java web applications, with flexible front-end, integration and batch processes.	Des., Evol. P.O.	≈6,500
F	1	1	2	Public sector	Web-based applications for the different departments of a public administration.	Des., Evol. P.O.	≈20,000
G	1	1	0	Public sector	Applications with enhanced reusability and reduced development costs.	Des. P.O.	≈4,500
H	1	1	2	Insurance	Applications integrated with services of an insurance company.	Des. P.O.	≈4,000
I	1	1	1	Public sector	Applications that include the business processes of a utility organization.	Des. P.O.	≈6,500

SA; Software Architect; AD: Architecture Developer; AB: Application Builder; Des: SRA Design; Evol: SRA Evolution; PO: Platform-Oriented SRA; AD: SRA with Application (i.e., business) Domain.

To address RQ1 and RQ2, we approached software architects, as they are responsible of designing the SRAs and to design the artifacts than conform an SRA. Furthermore, as SRA design is a non-deterministic and unsystematic task, the need of further understanding certain decisions and the rationale behind the SRA's artifacts were evident. Hence, we used semi-structured interviews based on an interview guide as a data gathering instrument. Semi-structured interviews provided us with the ability of eliciting details for each of the analyzed projects, whilst inquiring and understanding the particularities of each project, as follow up questions were allowed when required. In general, the interview guide was mostly focused on analyzing design tasks of software architects to create SRA artifacts and their opinions regarding the scope of reuse of the resulting SRA of the analyzed project (i.e., RQ3). Prior to the interview, we requested 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. Each interview took about one hour and was audio-taped and prepared for analysis through the manual transcription of the audio records into text documents (made by an external company and reviewed by the researchers).

Based on the results and insights gained from performing the semi-structured interviews to software architects, we then aimed at understanding the scope of reuse and usage aspects of the SRAs designed by *everis* (i.e., approaching RQ3 and RQ4 respectively). The target participants for gathering such information were architecture developers and application builders as they are the ones that develop or use the SRA designed by the software architects. Given the nature of RQ3 and RQ4 and the fact that it was almost impossible to contact personally architecture developers and application builders (they use to work on different locations based on the client organizations, sometimes located in other cities or even countries), we decided to use online questionnaires to approach them. Thus, two different online questionnaires were designed. Architecture developers and application builders were inquired about the reuse of the artifacts that conform the SRA being analyzed. In addition, application builders were also asked about how the artifacts of the SRA designed by *everis* were used in the context of the each client organization, as they are the ones that use the SRA designed by *everis*' software architects. To partially mitigate the rigidness of questionnaires, we provided room to add any comment or observation to all the questions therein. *everis* managers contacted the corresponding SRA project managers in order to ensure the stakeholders' participation.

3.3 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.

In the case of the interviews, we used an Excel-based template to organize each participant's answer to each question using tables. For doing this, we used the interview transcripts and individual notes taken by different researchers during the interviews. Then, we analyzed the data from two perspectives that lead us to a better understanding and interpretation of the results. First, we analyzed the answers at the project level, in order to understand the specific context of each project and the perspective of its stakeholders. Second, we analyzed the answers at whole by assessing all participants' answers related to each question, in order to categorize their answers using template tables that described (in each column): the name of the category, a detailed description of the cases covered by the category, the participant, and the explicit sentences that support the category. Such categories were then further discussed and analyzed by the entire team in order to better analyze the evidence and improve our understanding until reaching an agreement. As a result, some categories were split, modified, discarded or added to ensure that all answers were well-represented. In the case of online questionnaires, we used tools (e.g., LimeSurvey) to automatically gather and organize the data, and followed the same procedure above to generate categories.

It is important to emphasize that, in line with the qualitative character of our approach, the generated categories were aimed to provide us a way to analyze and report our findings instead of providing a quantitative vision of the *everis* context.

3.4 Limitations of the Study

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

Construct validity. To strengthen this aspect we made sure of performing a rigorous planning of the study and establishing a solid protocol and templates for data collection and data analysis by following guidelines for software engineering [33]. We paid special attention to design our data collection templates in such a way that qualitative details and representative sentences supporting the evidence are always captured and related. Our protocol included specific mitigation actions for evaluation apprehension by ensuring the confidentiality of the answers. In addition, the instruments used to gather data (i.e., the interview guide and the online questionnaires) were designed based on current SRA-related practices in the literature and practice, and then piloted and enhanced to ensure their effectiveness. For instance, we made sure of polishing the instruments with suitable vocabulary that the participants were familiar with. This was particularly relevant in our case as stakeholders used different terms to refer to the same thing. In this way we ensured the effectiveness of our instruments.

Internal validity. We are aware of factors that might affect internal validity such as the selection of the most successful projects as sampling by *everis*' managers. To minimize this, we explained them the importance of having a representative sampling of the SRA projects they perform in order to obtain reliable data. In addition, the facts that we analyzed each project independently and that diverse roles were approached for each project, allow us better interpretation and understanding of contextual information and the SRA–related practices. Otherwise, it would have been very difficult to interpret certain decisions or influential factors related to the nature of the projects or the activities of the stakeholders. Furthermore, 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 for. 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 them during the interviews in order to ask for clarifications if necessary; (2) for the online questionnaires, the respondent needed to answer all the corresponding questions while s/he could complete the questionnaire at any time (so it gives them the possibility of consulting documentation in case s/he needs to remember something), and optional extra space was given in each question to add comments; (3) in all cases we also had the opportunity to contact the participants when we really needed clarification. 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 answers correctness.

Other mitigation strategies were: recording and transcribing all interviews to contribute to 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 results.

External validity. We recognize that our results are tight to the *everis* context and should be interpreted as such. Therefore, the *everis* context has been detailed as much as possible. Nevertheless, the results offer useful evidence that might mostly serve in organizations that have similar context, e.g., Volvo [10], Océ [16], Credit Suisse [28], Dutch e-government [14], and German public administration [17].

We acknowledge that several factors that were not explicitly requested in our study may influence SRA artifacts, such as organizational processes and policies, resources, and cultural issues. But we think that our instruments were designed to identify and inquiry as much as possible in these situations.

We are aware of the importance of conducting similar studies in other organizations. Hence, we foster other researchers and practitioners to use the instruments of this study (available and further described at [23]). We expect that our results strengthen the evidence regarding SRA and encourage others to provide similar evidences that help to mature SRAs research and practice.

4. RESULTS

This section presents the results of the study. They are grouped in four subsections according to the RQs. 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. Besides, these categories are further explained in the annex [23]. 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.

4.1 RQ1: 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 develop to give support to such logic design" [C].

Regarding the produced deliverables, we present the results in terms of the three main types of artifacts below.

4.1.1 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].

4.1.2 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].

4.1.3 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 that: "when we have investigated how to communicate with an environment and consider 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].

4.2 RQ2: What is it Reused by *everis* in Order 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* that 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 and reuse 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 disintegrated in 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-oriented SRA that includes the most popular cross-cutting software elements common in diverse business domains. This corporate platform-oriented 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 RA" [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].

4.3 RQ3: 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 2):

- Platform-oriented SRA. Among these SRAs, we found two situations. On the one hand, platform-oriented 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-oriented SRA that also have some artifacts tied 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 argue the response" [A-AB].

Additionally, since most of participants replied that the SRA of their projects were platform-oriented 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 3. It can be observed that among the most popular ones (in more than 50% of projects), we can find: 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.

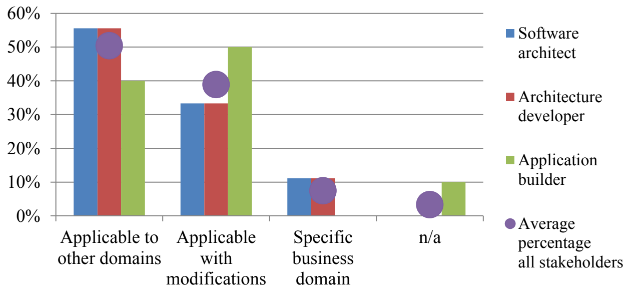

Figure 2. What stakeholders think about SRA reuse in other domains.

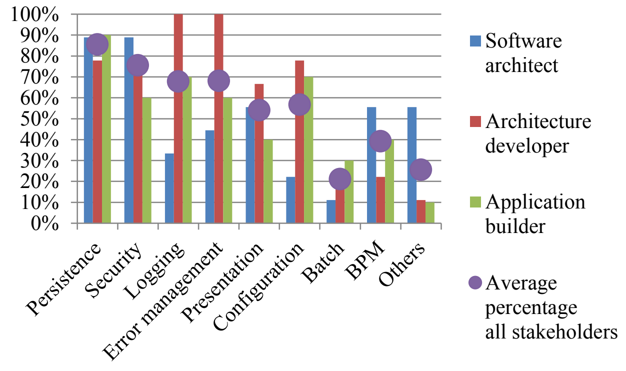

Figure 3. Popularity of SRAs' cross-cutting software elements.

4.4 RQ4: 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].
- 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. DISCUSSION

This section discusses the most relevant observations from the results. Each subsection corresponds to one RQ. 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.

5.1 Software Elements, Guidelines and Documentation are the Main Artifacts of SRAs

In Section 2.1 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. First, an SRA provides software elements (provided as source code or libraries) to be reused for all the applications (see Section 4.1.1). 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 (see Section 4.1.2). 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 (see Section 4.1.3). Next, we analyze the artifacts that were not highlighted by the participants of this study or that surprisingly differs from our results.

5.1.1 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 go down and think about the hardware infrastructure that supports it" [A]; b) firstly design of hardware infrastructure, e.g., "when we come to their environment we found a pre-defined hardware infrastructure, which was a Unix with WebSphere and Java" [D]; c) in parallel, e.g., "we design SRA modules and they had very strong non-functional requirements. At the same time, we needed to be sure that some hardware infrastructure would support it" [B].

5.1.2 SRAs mostly perceived at a technological level

The elaboration of mission, vision and strategy of the organization was not mentioned by the participants [8]. This element is the backbone of enterprise architectures [38]. 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 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.

5.1.3 An SRA could also include other artifacts

Some artifacts reported in the literature, such as algorithms [1], 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 [24][30].

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 [5]). 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. [29] and Herold et al. [17] are very close to these results. 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.

5.2 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-oriented 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 [6][13][24]. In industry, it seems difficult to find Futuristic SRAs (those only based on theoretical architectural patterns instead of experience [1]), being common Practice SRAs [1]. 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 wanted to take as much benefit from them as possible.

5.3 Platform-Oriented 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-oriented 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 business-oriented SRA for many organizations is AUTOSAR [3] that standardizes software development of automotive competing firms.

Although potentially reusable, platform-oriented SRAs are difficult to design in the beginning: "designing reuse SRAs complicates the design phase because you have to identify the pieces that really are reusable whereas the business logic is responsibility of the application developer" [F]. Platform-oriented SRAs are possible because "there are not that many architectural styles. For instance, Microsoft made a compendium of architectures [26]" [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 SRA (A, D, E, G, I) that were categorized as platform-oriented 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 this SRA projects were only in the design phase.

5.3.1 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, for instance, 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 module).

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).

5.4 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 that 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 [1]). 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 upload 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.

6. CONCLUSIONS AND FUTURE WORK

There is a vital need for disseminating empirical evidence to help researchers to assess current research and identify future research areas, and practitioners to choose appropriate methods and techniques that support the software architecture process [11].

A case study was conducted to analyze the artifacts that nine SRAs provide for the development of software systems in large organizations. The main findings were: (1) the identification of artifacts that compose the SRA infrastructure in nine SRAs in industry; (2) uncovering the utility of capturing knowledge from previous experiences; (3) analyzing when SRAs could be reused to other domains; and (4) checking how strictly SRAs are used in practice. These results aim to improve current empirical evidence in SRA research and to help practitioners in the design and use of SRA.

To sum up, the results from this work may help maturing SRA adoption and usage. On the one hand, these results disseminate in *everis* the artifacts of SRAs from previous projects. On the other hand, although these results are tight to the *everis* context, they offer useful evidence for organizations that have similar context. Thus, researchers and practitioners may also use the evidence provided by this paper in similar contexts for the inception, design, and application of SRAs.

As future work, we plan to conduct two more empirical studies to investigate other key assets of SRAs: (a) quality attributes and requirements; and (b) architectural decisions.

7. ACKNOWLEDGMENTS

This work has been supported by the "*Cátedra everis*" collaboration. We would like to thank all participants for their cooperation.

8. REFERENCES

[1] Ameller, D., Ayala, C., Cabot, J., Franch, X. 2012. How do software architects consider non-functional requirements: An exploratory study. In RE 2012, 41-50.

[2] Angelov, S., Grefen, P., and Greefhorst, D. 2012. A framework for analysis and design of software reference architecttures. Information and Software Technology 54(4), 417-431.

[3] AUTOSAR. 2011. Development Partnership AUTOSAR to extend scope of applications to non-automotive areas. http://www.autosar.org/download/media_release/Development%20Partnership%20AUTOSAR%20to%20extend%20scope%20of%20applications%20to%20non-automotive%20areas.pdf [Online; accessed 11-Sep-2013]

[4] Baldassarre, M. T., Caivano, D., & Visaggio, G. 2013. Empirical studies for innovation dissemination: ten years of experience. In EASE 2013, 144-152.

[5] Bass, L., Clements, P., Kazman, R. 2012. Software Architecture in Practice. Addison-Wesley Professional; 3rd edition.

[6] Beyer, H. J., Hein, D., Schitter, C., Knodel, J., Muthig, D., and Naab, M. 2008. Introducing architecture-centric reuse into a small development organization. In ICSR 2008, 1-13.

[7] Buchgeher, G., and Weinreich, R. 2013. Towards continuous reference architecture conformance analysis. In ECSA 2013.

[8] Cloutier, R., Muller, G., et al. 2010. The concept of reference architectures. Systems Engineering 13(1), 14-27

[9] Corbin, J., and Strauss, A. 2008. Basics of qualitative research. Sage Publications.

[10] Eklund, U., Askerdal, Ö. et al. 2005. Experience of introducing reference architectures in the development of automotive electronic systems. In ACM SIGSOFT SE Notes 30(4), 1-6.

[11] Falessi, D., Babar, M. A. et al. (2010). Applying empirical software engineering to software architecture: challenges and lessons learned. ESE Journal, 15(3), 250-276.

[12] Farenhorst, R., and van Vliet, H. 2009. Understanding how to support architects in sharing knowledge. In SHARK@ICSE 2009, 17-24.

[13] Galster, M., and Avgeriou, P. 2011. Empirically-grounded reference architectures: a proposal. In QoSA 2011, 153-158.

[14] Galster, M., Avgeriou, P., & Tofan, D. (2012). Constraints for the design of variability-intensive service-oriented reference architectures–An industrial case study. IST Journal.

[15] Glass, R. L. 1994. The software-research crisis. Software, IEEE, 11(6), 42-47.

[16] Graaf, B., Van Dijk, H., and van Deursen, A. 2005. Evaluating an Embedded Software Reference Architecture—Industrial Experience Report—. In CSMR 2005, 354-363.

[17] Herold, S., Mair, M., Rausch, A., & Schindler, I. 2013. Checking Conformance with Reference Architectures: A Case Study. In EDOC 2013, pp. 71-80.

[18] Irlbeck, M., Bytschkow, D., Hackenberg, G. et al. 2013. Towards a Bottom-Up Development of Reference Architectures for Smart Energy Systems. In SE4SG 2013.

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

[20] Martínez-Fernández, S., Ayala, C., Franch, X., Marques, H. and Ameller, D. 2013. A Framework for Software Reference Architecture Analysis and Review. ESELAW@CIbSE 2013.

[21] Martínez-Fernández, S., Ayala, C. P., Franch, X., and Marques, H. 2013. REARM: A Reuse-Based Economic Model for Software Reference Architectures. In ICSR 2013, 97-112.

[22] Martínez-Fernández, S., Ayala, C., Franch, X., and Martins, H. Benefits and Drawbacks of Reference Architectures. In ECSA 2013, 307-310.

[23] Martínez-Fernández, S. 2013. Materials used in this study. http://www.essi.upc.edu/~smartinez/files/ease14annex.pdf [Online; accessed; 26-Jan-2014]

[24] Mattmann, C. A., Downs, R. R., Marshall, J. J., Most, N. F., et al. 2010. Reuse of software assets for the NASA Earth science decadal survey missions. In IGARSS 2010, 1687-1690.

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

[26] Microsoft. 2009. Microsoft Application Architecture Guide, 2nd Edition. http://msdn.microsoft.com/en-us/library/ff650706.aspx [Online; accessed 30-Apr-2012]

[27] Miles, M. B., and Huberman, A. M. 1994. Qualitative data analysis: An expanded sourcebook. 2nd Edition. SAGE.

[28] Murer, S. 2013. 15 Years of Service Oriented Architecture at Credit Suisse. http://www.sei.cmu.edu/library/assets/presentations/murer-saturn2013.pdf [Online; accessed;09-Sep-2013]

[29] Nakagawa, E. Y. et al. 2012. Ramodel: A reference model for reference architectures. In Joint WICSA 2012, 297-301.

[30] Panunzio, M., & Vardanega, T. 2013. On Software Reference Architectures and Their Application to the Space Domain. In ICSR 2013, 144-159.

[31] Passos, L et al. 2010. Static architecture-conformance checking: An illustrative overview. Software IEEE, 27(5), 82-89.

[32] Qureshi, N., Usman, M., and Ikram, N. 2013. Evidence in software architecture, a systematic literature review. In EASE 2013, 97-106.

[33] Runeson, P., and Höst, M. 2009. Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering 14(2), 131-164.

[34] Runeson, P., Host, M., Rainer, A. and Regnell, B. 2012. Case Study Research in Software Engineering: Guidelines and Examples. Wiley.

[35] Seddon, P., and Scheepers, P. 2011. Towards the improved treatment of generalization of knowledge claims in IS research: drawing general conclusions from samples, EJIS, 1-16.

[36] Scott, D. 2012. Gartner hype cycle for real-time infrastructure 2012. http://www.gartner.com/id=2570016 [Online; accessed;13-Sep-2013]

[37] The Open Group. SOA Reference Architecture Technical Standard. http://www.opengroup.org/soa/source-book/soa_refarch/intro.htm [Online; accessed 22-Jan-2014]

[38] Winter, R., and Fischer, R. 2006. Essential layers, artifacts, and dependencies of enterprise architecture. In EDOCW.

[39] Yin, R. K. 2009. Case Study Research: Design and Methods. SAGE Publications.