A Framework for Software Reference Architecture Analysis and Review

Silverio Martínez-Fernández, Claudia Ayala, Xavier Franch, David Ameller

GESSI Research Group, Universitat Politècnica de Catalunya, Barcelona, Spain
{smartinez,cayala,franch,dameller}@essi.upc.edu

Abstract. Tight time-to-market needs pushes software companies and IT consulting firms (ITCFs) to continuously look for techniques to improve their IT services in general, and the design of software architectures in particular. The use of software reference architectures allows ITCFs reusing architectural knowledge and components in a systematic way. In return, ITCFs face the need to analyze the return on investment in software reference architectures for organizations, and to review these reference architectures in order to ensure their quality and incremental improvement. Little support exists to help ITCFs to face these challenges. In this paper we present an empirical framework aimed to support the analysis and review of software reference architectures and their use in IT projects by harvesting relevant evidence from the wide spectrum of involved stakeholders. Such a framework comes from an action research approach held in an ITCF, and we report the issues found so far.

Keywords: Software architecture, reference architecture, architecture analysis, architecture evaluation, empirical software engineering.

1 Introduction

Nowadays, the size and complexity of information systems, together with critical time-to-market needs, demand new software engineering approaches to design software architectures (SA) [17]. One of these approaches is the use of software reference architectures (RA) that allows to systematically reuse knowledge and components when developing a concrete SA [8][13].

As defined by Bass et al. [3], a reference model (RM) is ‘‘a division of functionality together with data flow between the pieces’’ and an RA is ‘‘a reference model mapped onto software elements (that cooperatively implement the functionality defined in the reference model) and the data flows between them’’.

A more detailed definition of RAs is given by Nakagawa et al. [17]. They define an RA 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 develop-

ment of systems for that domain. All of this must be supported by a unified, unambiguous, and widely understood domain terminology".

In this paper, we use these two RA definitions. We show the relationships among RM, RM-based RA and RA-based concrete SA in Fig. 1. Throughout the paper, we use the term RA to refer to RM-based RA and SA to refer to RA-based concrete SA. Angelov et al. have identified the generic nature of RAs as the main feature that distinguishing them from concrete SAs. Every application has its own and unique SA, which is derived from an RA. This is possible because RAs are abstract enough to allow its usage in differing contexts. [2]

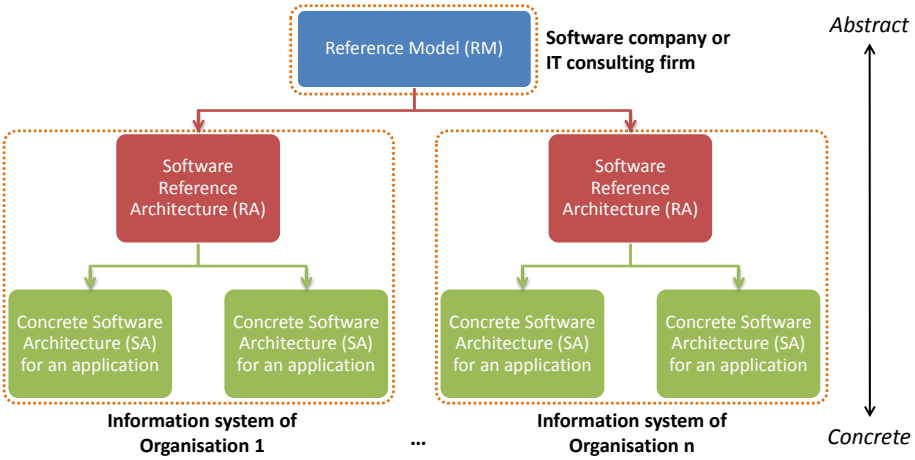

Fig. 1. Relationships among RM, RA and SA.

The motivations behind RAs are: to facilitate reuse, and thereby harvest potential savings through reduced cycle times, cost, risk and increased quality [8]; to help with the evolution of a set of systems that stem from the same RA [13]; and to ensure standardization and interoperability [2]. Due to this, RAs are becoming a key asset of organizations [8].

However, although the adoption of an RA might have plenty of benefits for an organization, it also implies several challenges, such as the need for an initial investment [13] and ensuring its adequacy for the organization's portfolio of applications. Hence, in order to use RAs, software companies and information technology consulting firms (ITCFs) face two fundamental questions:

- Is it worth to invest on the adoption of an RA?
- Once adopted, how the suitability of an RA for deriving concrete SAs for an organization's applications can be ensured?

Currently, organizations lack of support for dealing with these questions. On the one hand, there is a shortage of economic models to precisely evaluate the benefit of architecture projects [6] in order to take informed decisions about adopting an RA in an organization. On the other hand, although there are qualitative evaluation methods for RAs [1][12][14], they do not systematize how these RAs should be evaluated

regarding certain quality attributes (for instance, their capability to satisfy the variability in applications developed from RAs [18]).

In this context, the goal of this research is to devise a framework that supports organizations to deal with the aforementioned questions by providing procedural guidelines for setting up and carrying out empirical studies aimed to extract evidence for: 1) supporting organizations to assess if it is worth to adopt an RA, and 2) ensuring the suitability of an RA for deriving concrete SAs for an organization's applications. It is worth mentioning that this research has its origin in an ongoing action-research initiative among our research group and the Center of Excellence on Software Architectures (ARCHEX) of Everis, a multinational consulting company based in Spain. ARCHEX faced the fundamental questions stated above and the framework proposed in this paper was mainly originated and shaped throughout our involvement for helping ARCHEX to envisage a suitable solution. The idea behind devising such a framework is twofold: to help other organizations dealing with similar problems as ARCHEX; and to improve the guidelines of the framework by the experience gained in each application of the framework in order to consolidate architectural knowledge from the industrial practice.

The paper is structured as follows. In Section 2 we describe the fundamental aspects of RAs that are suggested to be assessed. In Section 3 we describe the empirical studies that compose the framework. In Section 4 we present the context of ITCFs and show how the framework can be applied in the context of an ITCF. In Sections 5 and 6 we present preliminary results of two studies of the framework applied in Everis. In Section 7 we end up with conclusions and future work.

2 Practical Review Criteria for Reference Architectures

A commonly accepted set of criteria to assess RAs does not exist [1][12-14]. Thus, in this section we identify important aspects to assess RAs out of practice and out of the literature. The framework presented in this paper envisages these aspects as a primary input for their further refinement based on the evidence from organizations.

Table 1. Summary of relevant aspects for software reference architecture assessment

	Aspect	Description of the Architectural Aspect
Qualitative	1	Overview: functionalities [1], origin, utility and adaptation [13]
	2	Requirements and quality attributes analysis [1][10][12]
	3	Architectural knowledge and decisions [10][12][17]
	4	Business qualities [1] and architecture competence [4]
	5	Software development methodology [10][17]
	6	Technologies and tools [10][17]
Quantitative	7	Benefits and costs metrics to derive SAs from RAs [10]

In [1], Angelov et al. state that SAs and RAs have to be assessed for the same aspects. For this reason, we started by analyzing some available works on SA assessment [4][10]. However, existing evaluation methods for SAs are not directly applica-

ble to RAs because they do not cover the generic nature of RAs [1]. Therefore, we elaborated further this analysis considering both the specific characteristics of RAs as described in [1][12-13][17] and our own experience in the field. The resulting aspects for assessing RA are detailed below and summarized in Table 1.

Aspect 1 refers to the need of having an overview of the RA. It includes an analysis of its generic functionalities, its domain [1], its origin and motivation, its correctness and utility, and its support for efficient adaptation and instantiation [13]. Since RAs are defined to abstract from certain contextual specifics allowing its usage in differing contexts [2], their support for efficient adaptation and instantiation while deriving concrete SAs is an aspect to assess [13].

Many prominent researchers [1][7][10][12] highlight the importance of quality attributes, as well as architectural decisions for the SA design process and the architectural assessment. These two aspects should also be considered for the RA assessment because, as we said, SAs and RAs have to be assessed for the same aspects [1]. Thus, we considered them as Aspects 2 and 3 respectively. However, since an RA has to address more architectural qualities than an SA (e.g., applicability) [1], this analysis could be wider for RAs in this sense. A list of quality attributes that are strictly determined by SAs is defined in [7]. This list consists of the following ten quality attributes: performance, reliability, availability, security, modifiability, portability, functionality, variability, subsetability and conceptual integrity.

SAs also address business qualities [1] (e.g., cost, time-to-market) that are business goals that affect their competence [4]. It is considered as Aspect 4.

To improve the SA design process, there also exist supportive technologies such as methods, techniques and tools [10][17]. Thus, it is not only important for an RA to collect data to assess its design process, but also its supportive technologies, which are assessed by Aspects 5 and 6.

As stated in [10], a crucial aspect to define the goodness of a SA is related to the ROI. The optimal set of architectural decisions is usually the one that maximizes the ROI. Aspect 7 is intended to quantify benefits and costs of RAs to calculate their ROI.

We recommend gathering evidence about all these aspects, which are summarized in Table 1, while assessing an RA. Existing methods for SA assessment have been previously applied for RA assessment, such as in [1][12][14]. However, none of them cover all the aspects of Table 1, especially Aspect 7. Hence, new approaches to assess RAs considering these aspects altogether are required. This has motivated our work.

These architectural aspects can be divided in two areas of different nature. First, Aspects 1 to 6 are qualitative architectural concerns. Second, Aspect 7 consists of quantitative metrics to calculate the benefits and costs of deriving SAs from RAs.

3 An Empirical Framework to Review Reference Architectures

In this section, we present the ongoing version of our empirical framework. In order to design the framework, we have followed the guidelines for conducting empirical studies in software engineering recommended by Wohlin et al. [21]. It is composed of

an assortment of empirical studies. Each empirical study reviews a subset of the relevant architectural aspects presented in Table 1.

Current economic models [16] and RAs evaluation methods (e.g., [1][12-14]) suggest to gather Aspects 7, and Aspect 1-6 respectively, directly from the organizations. However none of them provides support nor guidelines for doing so. Thus, our framework is aimed to provide support for such gathering process while suggests to apply any of the existing methods to evaluate the economic costs of adopting an RA or its quality based on the empirically obtained data. The selection of the method used in each situation would depend on the organization context [5].

Regarding to Aspect 7, an economic model becomes necessary. The data needed to feed such an economic model depends on the existing value-driven data in the organization (see Sections 3.1 and 5). Such data may be gathered by conducting post-mortem studies that collect real metrics or, when the organization does not have previous experience with RAs, to estimate these metrics is suggested.

In order to gather data to cover Aspects 1-6, our framework suggests conducting surveys (see Sections 3.3 and 6). These studies gather information not only from RA projects, but also from SA projects as they are a direct outcome of the RA usage. This allows analyzing the RA's suitability for producing the SAs of enterprise applications in organizations as well as detecting improvement opportunities.

Fig. 2 summarizes the studies that compose the framework. The studies are classified by their approach to assess the RA (qualitative or quantitative), depending on which question of Section 1 they answer. It is important to note that the studies suggested by our framework are complementary and support each other. Our framework benefits from this combination of studies. For instance, results from a preceding empirical study can be used to corroborate or develop further these results (e.g., results from the survey to check existing value-driven data in organizations indicate the data that is useful for calculating the ROI). For this reason, the suggested studies can be conducted sequentially.

The assortment of studies that has been envisaged for our framework is detailed below. In order to explain each study, a similar same structure as in [10] has been used: context and motivation, objectives, and method.

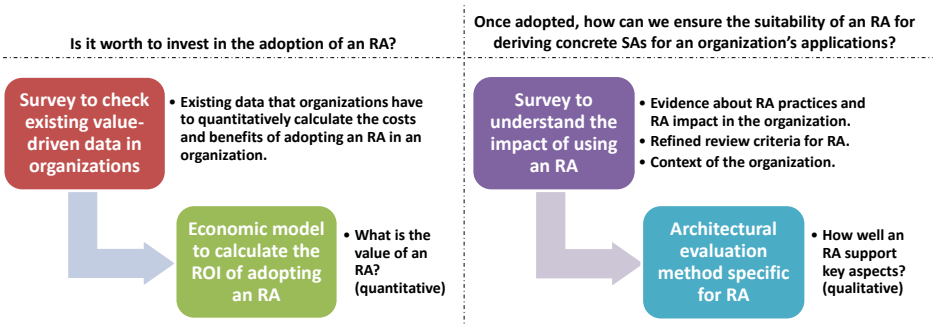

Fig. 2. Empirical studies of the framework to assess RAs.

3.1 Surveys to check existing value-driven data in RA projects

Context. Typically, organizations do not have resources to compare the real cost of creating applications with and without an RA. Thus, alternatives should be envisaged.
Objective. To discover existing data that organizations have to quantitatively calculate the costs and benefits of adopting an RA in an organization.
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 RA projects and application projects.

3.2 Applying an economic model to calculate the ROI of adopting an RA

Context. Before deciding to launch an RA, organizations need to analyze whether undertaking or not the investment. Offering them an economic model that has been used in former projects can help them to make more informed decisions.
Objective. To assess whether it is worth investing in an RA.
Method. Depending on the maturity of the organization, two methodologies can be applied. If the organization does not have an RA, the economic model should be fed with estimated data. Nevertheless, when the organization already has an RA, 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 RA. Some related works explain how to calculate the ROI of a product [11], and software reuse [19]. We suggest using the economic model for RAs presented in [14].

3.3 Surveys to understand the impact of using an RA

Context. To refine the set of review criteria for RAs, it is needed to understand RA's characteristics, as well as its potential benefits and limitations. Assessing previous RA projects is a feasible way to start gaining such an understanding.
Objective. To understand the impact and suitability of an RM for the elaboration of RAs, and of an RA for the creation of SAs. Improvement insights can also be identified from different stakeholders.
Method. Exploratory surveys with personalized questionnaires applied to relevant stakeholders (e.g., architects, developers) to gather their perceptions and needs.

3.4 Applying an architectural evaluation method to prove RA effectiveness

Context. Architecture is the product of the early design phase [7]. RA evaluation is a way to find potential problems before implementing RA modules, and to gain confidence in the RA design provided to SA projects.
Objective. To analyze the RA strengths and weaknesses and to determine which improvements should be incorporated in the RA.
Method. An existing evaluation method specific for RAs such as [1][12-14]. The selection of the method would depend on the organization context [5].

4 Use of the framework in an IT consulting firm

4.1 Context of Information Technology Consulting Firms

Motivation. We are interested in the case in which an ITCF has designed an RA with the purpose of deriving concrete SAs for each application of a client organization. This usually happens when the ITCF is regularly contracted to create or maintain information systems in client organizations. Each information system is built upon the RA and includes many enterprise applications (see Fig. 3).

As Angelov et al. point out, an RA can be designed with an intended scope of a single organization or multiple organizations that share a certain property. Although Fig. 3 shows RAs that are used for the design of concrete SAs in a single organization, there also exist RAs for multiple organizations that share a market domain or a technological domain such as enterprise web applications [2].

The use of RAs allows ITCFs to reuse the architectural knowledge of their RM, and software components (normally associated to particular technologies) for the design of SAs in client organizations. Thus, RAs inherit best practices from previous successful experiences and a certain level of quality. The goal of these RAs provide a baseline that facilitates standardization and interoperability as well as the attainment of business goals during enterprise applications' development and maintenance.

Kinds of projects. There are three kinds of projects with different targets (Fig. 3): 1) RM projects; 2) RA projects; and 3) SA projects.

Stakeholders for RA analysis. Stakeholders need to be clearly defined for RA assessment purposes [1]. The people involved in an RA assessment are the evaluation team, which conducts the empirical studies of the framework, and stakeholders from architectural projects. In the three kinds of projects defined above performed by ITCFs, we consider the following five stakeholders essential for RA assessment: project business manager, project technological manager, software architect, developer, and application builder. 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 kinds of projects [12]. However, there could be more people involved in an architectural evaluation, as Clements et al. indicate in [7]. Below, we describe to which kind of project stakeholders belong and their interests.

RM projects. It is composed of software architects from the ITCF that worked in previous successful RA projects. They are specialized in architectural knowledge management. Their goal is to gather the best practices from previous RA projects' experiences in order to design and/or improve the corporate RM.

RA projects. RA projects involve people from the ITCF 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 the ITCF are responsible for meeting schedule and interface with the project business managers from the client organization.

Software architects (also called as RA managers) usually come from the ITCF, 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 the ITCF and are responsible of coding, maintaining, integrating and testing the RA software components and documenting them.

SA projects. Enterprise application projects can involve people from the client organization and/or subcontracted ITCFs (which may even be different than the RM owner) whose members are usually very familiar with the specific organization domain. The participation of the client organization in RA and SA projects is one possible strategy for ensuring the continuity of their information systems without having much dependency on subcontracted ITCF.

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 RA reusable components and instantiate them to build an application.

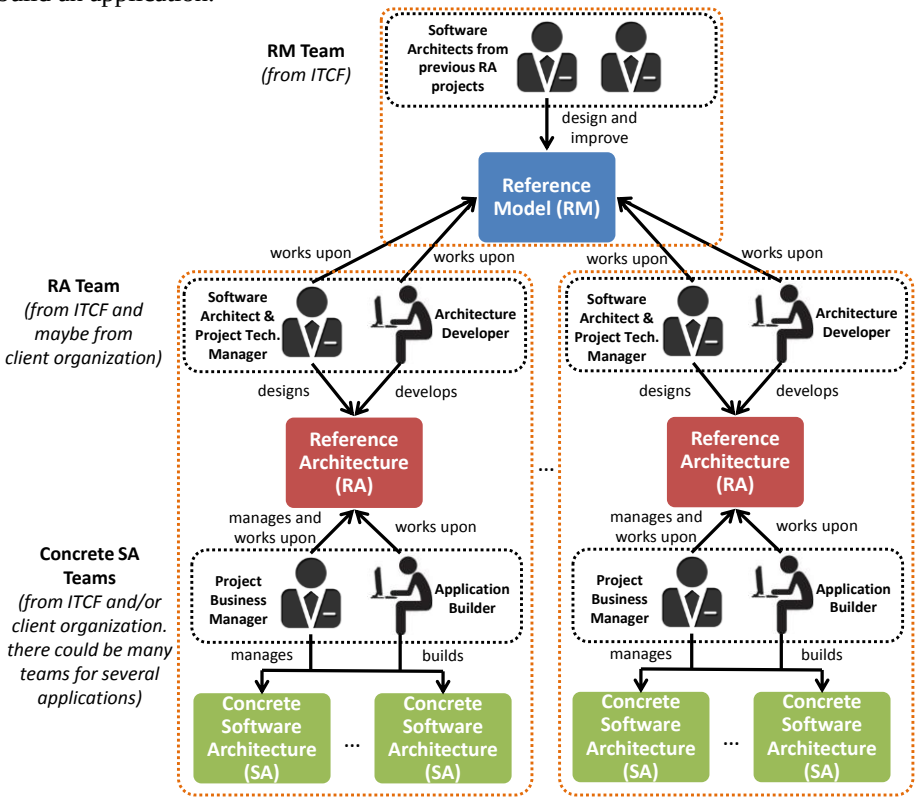

Fig. 3. Relevant stakeholders for the assessment of RM, RA and SA projects.

4.2 Instantiation of the Framework

The presented empirical framework is currently being applied at the Architecture Centre of Excellence of Everis, a multinational ITCF. The main motivation of Everis for conducting the empirical studies is twofold: 1) strategic: providing quantitative evidence to their clients about the potential benefits of applying an RA; 2) technical: identifying strengths and weaknesses of an RA.

Everis fits into the context described in Section 4.1 (e.g., they carry out the three types of projects described there). Following the criteria found in [1], RAs created by Everis can be seen as Practice RAs, since they are defined from the accumulation of practical knowledge (the architectural knowledge of their corporate RM). According to the classification of [2], they are also classical, facilitation architectures designed to be implemented in a single organization. They are classical because their creation is based on experiences, and their aim is to facilitate guidelines for the design of systems, specifically for the information system domain.

All the studies suggested in Section 3 are planned to be conducted to understand and evaluate RAs defined by Everis. In this paper, we present the protocol and preliminary results of the two surveys of the understanding step. Section 5 describes the available value-driven data in projects in order to create or choose an economic model to calculate the ROI of adopting an RA. In Section 6, an excerpt of the survey protocol, which has already been designed and reviewed, is presented. The survey is still in the analysis step. However, the data about the Aspect 4 (business qualities) have already been processed. Preliminary results about this aspect show the benefits and aspects to consider for organizations that adopt RAs.

Table 2 shows how the roles are covered by the different studies in the Everis case.

Table 2. Stakeholders of the Everis case

Project[a]	Business Manager	Technical Manager	Software Architect	Architecture Developer	Application Builder
RM	n/a	n/a	S1, ROI, S2	n/a	n/a
RA	S2, Eva	S1, S2, Eva	ROI, S2, Eva	S2, Eva	S2, Eva
SA	n/a	n/a	n/a	n/a	S1, S2

a. Legend: Survey to study existing data (S1), Economic model for RA's ROI (ROI), Survey to understand RA projects (S2), and RA evaluation (Eva).

5 Survey to check existing value-driven data in projects

5.1 Protocol

Objectives of this study. The objective of this survey is to identify the quantitative information that can be retrieved from past projects in order to perform a cost-benefit analysis. The cost-benefit analysis, which is the *evaluation* step of the framework, needs this kind of data to calculate the ROI of adopting an RA in an organization.

Sampling. A sample of four RA projects and several enterprise application built upon them has been selected.

Approach for data collection. The main perceived economic benefit on the use of RAs are the 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 [16]. We use online questionnaires to ask project technical managers and application builders about existing information that there is about past projects for calculating these cost savings. When the organization does not have any experience in RAs, these data can be estimated.

5.2 Preliminary results: costs and benefits metrics for RAs

In this section we describe the information that typically is available in order to calculate the costs and benefits of adopting an RA. We divide existing information in two categories: effort and software metrics. On the one hand, the invested effort from the tracked activities allows the calculation of the costs of the project. On the other hand, software metrics indicate the benefits that can be found in the source code.

Effort metrics to calculate projects' costs. The effort of the training, development and maintenance activities is usually tracked. The training effort is the total amount of hours invested to the training on the SA teams. These training could be done through workshops, user manuals, wikis, web portals, and/or other training activities. The development effort is the total amount of hours invested in the development of the RA and the SAs of applications. It could be extracted from the spent time for each activity of the development of the projects. The maintenance effort is the total amount of hours invested in the maintenance of the RA and the SAs of applications. Maintenance activities include changes, incidences, support and consults.

Software metrics to calculate benefits in reuse and maintainability. The analysis of the code from RA and SA projects allow to quantify the size of these projects in terms of LOC or function points (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 the reuse of RA modules, we can calculate the benefits of RA by estimating the benefits of reusing them. Poulin defines a model for measuring the benefits of software reuse [19]. Maintenance savings due to a modular design could be calculated with design structured matrices (DSM) [15]. For a detailed explanation about how such metrics can be used in a cost-benefit analysis, the reader is referred to [16].

5.3 Lessons learned.

Architecture improvements are extremely difficult to evaluate in an analytic and quantitative fashion as the efficacy of the business (sales, marketing, and manufacturing) [6]. This is because software development is a naturally low-validity environment and reliable expert intuition can only be acquired in a high-validity environment [9]. In order to evaluate RAs 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. There are tools that allow it. In

order to get the metrics defined in the Section 5.2, JIRA[1] and Redmine[2] 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[3] and Jenkins[4]. We think that adopting good 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.

6 Survey to understand the impact of using an RA

6.1 Protocol.

Objectives of this survey. The purpose of the survey is to understand the impact of using RAs for designing the SAs of the applications of an information system of a client organization. This is a descriptive survey that measures what occurred while using RAs rather than why. The following research questions are important in order to review relevant Aspects 1 to 6 of RAs (defined in Section 2):

1. How is an RA adapted for creating SAs of an organization's applications?
2. What is the state of practice on requirements engineering for RAs?
3. What is the state of practice on architectural design for RAs?
4. How does the adoption of RAs provide observable benefits to the different involved actors?
5. What methodologies are currently being used in RA projects by Everis?
6. Which tools and technologies are currently being used in RAs projects by Everis?

Sampling. The target population of this survey are RA projects and SA projects. A representative sample of these projects in nine organizations in Europe (seven from Spain) has been selected.

Approach for data collection. On the one hand, semi-structured interviews are used for project technological managers, software architects, and client's project business managers. The reason of using interviews is that these roles have higher knowledge than the other roles about the architectural aspects of the Table 1, or another perspective in the case of client's project business managers, so we want to collect as much information as possible from them. Prior to the interviews, questionnaires are delivered to collect personal information about the interviewee and to inform him/her about the interview. On the other hand, online questionnaires are used for RA developers and application builders, since most of their questions are about supportive technologies and their responses can be previously listed, simplifying the data collection process.

This is an excerpt of the survey protocol, the complete version of the protocol is available at http://www.essi.upc.edu/~gessi/papers/eselaw13-survey-protocol.pdf.

---

1 JIRA, http://www.atlassian.com/es/software/jira/overview
2 Redmine, http://www.redmine.org/
3 Sonar, http://www.sonarsource.org/
4 Jenkins, http://jenkins-ci.org/

6.2 Preliminary results: strengths and weaknesses of RAs

In this section we present preliminary results about the *business quality* section of the survey, which are the answer to the fourth research question of the protocol: "How does the adoption of RAs provide observable benefits to the different involved actors?" Below, the benefits and aspects to consider that we found out are reported.
Benefits. The benefits of using an RA for the creation of applications are mainly:

- Increased quality of the enterprise applications.
  - An RA helps to accomplish business needs by improving key quality attributes.
  - An RA helps to improve the business processes of an organization.
  - An RA reuses architectural knowledge of previous successful experiences.
- Reduction of the development time and faster delivery of applications.
  - An RA allows starting developing applications since the first day by following architectural decisions already taken.
  - An RA decreases the development time of applications since the RA's modules that implement needed functionality are reused in the application.
- Increased productivity of application builders.
  - An RA facilitates material and tools for the development, testing and documentation of applications, and for training application builders.
  - An RA generates or automatizes the creation of code in the applications.
  - An RA indicates the guidelines to be followed by the application builders.
  - An RA reduces the complexity of applications' developments because part of the functionality is already resolved in the RA.
  - An RA facilitates the configuration of its modules and the integration with legacy systems or external systems.
- Cost savings in the maintenance of the applications.
  - An RA increases the control over applications through their homogeneity.
  - An RA maintains only once reused services by all applications.
  - An RA allows adding, changing or deleting functionalities by means of a modular design.
  - An RA establishes standards and "de facto" technologies that will be supported in future versions.

Aspects to consider. The adoption of an RA for the creation of applications implies the consideration of the following aspects that eventually can become risks:

- Initial investment. An RA implies an initial investment of time to start developing applications. This initial investment could be decreased in the case of using an RM. Also, it is also recommended to invest in an RA when the organization has a wide portfolio of applications that can be based on the RA.
- Additional learning curve. An RA implies an additional training for their own tools and modules, even if its technologies are standard or "de facto" already known by the application builder.
- Dependency on the RA. Applications depend on the reused modules of the RA. If it is necessary to make changes in a reused module of the RA or to add a new func-

tionality, application builders have to wait for the RA developers to include it in the RA for all the applications.
- Limited flexibility of the applications. The use of an RA implies following its guidelines during the application development and adopting its architectural design. If business needs require a different kind of application, the RA would limit the flexibility of that application.

6.3 Lessons learned

During the pilot of the survey, we learnt the following lessons about its design:

- The same term could have slightly different meaning in the academia and in the industry (for instance, the term "enterprise architecture" is sometimes use in the industry to mean "software reference architecture for a single organization").
- Questions that deal with several variables disconcert the interviewee and make the analysis more difficult. It is 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.
- In online questionnaires, it is recommendable to allow the interviewee to write any comments or clarifications in some field and also include a "n/a" option when necessary. Besides, a previous button is useful to make changes in previous questions.

7 Conclusions and Future Work

Driving empirical studies is becoming one of the main sources of communication between practitioners and the academia. The main contribution of this ongoing work intends to be the formulation of a framework to conduct empirical studies for supporting decision making and assessment related to RAs. It consists of a list of relevant aspects for RAs assessment, and an assortment of four complementary empirical studies that allow understanding and evaluating these aspects.
It is a practical framework that can be adapted to the specific context of software companies and ITCFs. Consequently, organizations that apply the framework could benefit from a common reference framework to review RAs.

The framework is being applied in Everis. This allows getting feedback for assessing its effectiveness and gathering industrial evidence. Preliminary results of this application indicate the importance of good practices like time tracking, continuous feedback, test-driven development and continuous integration in order to quantitatively evaluate RAs. Another result is that the adoption of an RA implies cost savings in the development and maintenance of applications.

Future work spreads into two directions. In terms of validation, we are also conducting the evaluation step of the framework in Everis. With respect to this first version of the framework, we aim to extend it considering Wohlin's improvement step (see Section 3) in order to build preliminary guidelines for improving RAs in ITCFs.

References

1. S. Angelov, J.J.M. Trienekens, P. Grefen, “Towards a Method for the Evaluation of Reference Architectures: Experiences from a Case,” ECSA 2008.
2. S. Angelov, P. Grefen, D. Greefhorst, “A Framework for Analysis and Design of Software Reference Architectures,” Information and Software Technology, 54(4), 2012.
3. L. Bass, P. Clements, R. Kazman, “Software Architecture in Practice,” Addison-Wesley Professional, 2003.
4. L. Bass, P. Clements, R. Kazman, M. Klein, “Evaluating the Software Architecture Competence of Organisations,” ECSA 2008.
5. L. Bass, R.L. Nord, “Understanding the Context of Architecture Evaluation Methods,” WICSA/ECSA, 2012
6. J. Carriere, R. Kazman, I. Ozkaya, “A cost-benefit framework for making architectural decisions in a business context,” ICSE 2010.
7. P. Clements, R. Kazman, M. Klein, “Evaluating Software Architectures: Methods and Case Studies,” Addison-Wesley Professional, 2001
8. R. Cloutier et al., “The Concept of Reference Architectures,” Systems Engineering, 13(1), 2010.
9. H. Erdogmus, J. Favaro, “The Value Proposition for Agility–A Dual Perspective,” 2012. Available at: http://www.infoq.com/presentations/Agility-Value
10. D. Falessi, M. Ali Babar, G. Cantone, P. Kruchten, “Applying Empirical Software Engineering to Software Architecture: Challenges and Lessons Learned,” Empirical Software Engineering, 15(3), 2010.
11. Forrester Research, “Forrester‟s Total Economic Impact (TEI)”, available online at: www.forrester.com/TEI
12. B.P. Gallaguer, “Using the Architecture Tradeoff Analysis MethodSM to Evaluate a Reference Architecture: A Case Study,” DTIC Document, 2000.
13. M. Galster, P. Avgeriou, “Empirically-grounded Reference Architectures: A Proposal,” QoSA+ISARCS, 2011.
14. B. Graaf, H. van Dijk, A. van Deursen, “Evaluating an embedded software reference architecture,” CSMR, 2005.
15. A. MacCormack, J. Rusnak, and C. Baldwin, “Exploring the Duality between Product and Organizational Architectures: A Test of the Mirroring Hypothesis”, Harvard Business School Working Paper, 2008, available at: http://hbswk.hbs.edu/item/5894.html.
16. S. Martínez-Fernández, C. Ayala, and X. Franch. "A Reuse-Based Economic Model for Software Reference Architectures". Technical Report ESSI-TR-12-6, November 2012, available at: http://hdl.handle.net/2117/16970
17. E. Y. Nakagawa, P. O. Antonino, M. Becker, “Reference Architecture and Product Line Architecture: A Subtle But Critical Difference,” ECSA 2011.
18. E. Y. Nakagawa, “Reference Architectures and Variability: Current Status & Future Perspectives,” VARSE@ECSA, 2012.
19. J.S. Poulin, “Measuring Software Reuse,” Reading, MA: Addison-Wesley, 1997.
20. P. Runeson, M. Höst, “Guidelines for Conducting and Reporting Case Study Research in Software Enginnering,” ESEM 2008.
21. C. Wohlin, M. Höst, K. Henningsson, “Empirical Research Methods in Software Engineering,” ESERNET 2003.