REARM: A Reuse-Based Economic Model for Software Reference Architectures

Silverio Martínez-Fernández[1], Claudia Ayala[1], Xavier Franch[1], Helena Martins Marques[2]

[1] GESSI Research Group, Universitat Politècnica de Catalunya, Barcelona, Spain
{smartinez,cayala,franch}@essi.upc.edu
[2] *everis*, Barcelona, Spain
hmartinm@everis.com

Abstract. To remain competitive, organizations are challenged to make informed and feasible value-driven design decisions in order to ensure the quality of their software systems. However, there is a lack of support for evaluating the economic impact of these decisions with regard to software reference architectures. This damages the communication among architects and management, which can result in poor decisions. This paper aims at ameliorating this problem by presenting a pragmatic preliminary economic model to perform cost-benefit analysis on the adoption of software reference architectures as a key asset for optimizing architectural decision-making. The model is based on existing value-based metrics and economics-driven models used in other areas. A preliminary validation based on a retrospective study showed the ability of the model to support a cost-benefit analysis presented to the management of an IT consulting company. This validation involved a cost-benefit analysis related to reuse and maintenance; other qualities will be integrated as our research progresses.

Keywords: Software architecture, reference architecture, economic model, architecture evaluation, cost-benefit analysis, quality attributes.

1 Introduction and motivation

Nowadays, the size and complexity of software systems, together with critical time-to-market needs, demand new software engineering approaches to software development. One of these approaches is the use of software reference architectures (RA), which are becoming widely studied and adopted in research and practice [3][19].

As defined by Bass et al. [5], 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". An RA encompasses the knowledge about how to design concrete software architectures (SA) of systems of a given domain; it must address the business rules, architectural styles, best practices of software development, and the software elements that support development of systems [28].

The motivations behind RAs are: to systematically reuse knowledge and software elements when developing concrete SA for new systems and thereby harvest potential savings through reduced cycle times, cost, risk and increased quality [11]; to help with the evolution of a set of systems that stem from the same RA [18]; and to ensure standardization and interoperability [3].

However, although the adoption of an RA might have plenty of benefits for an organization, it also implies several challenges, among them the need for an initial investment [18]. Hence, in order to use RAs, organizations face a fundamental question: "Is it worth to invest on the adoption of an RA?"

Thus, organizations need to ensure the feasibility of adopting an RA by assessing their goals, the resources they can invest and the expected benefits. In spite of this need, there is a lack of research methods for economics-driven RA evaluation [29]. Besides, there is a shortage of economic models to "precisely evaluate the benefit of 'architecture projects' - those that aim to improve one or more quality attributes of a system" [8]. Thus, the adoption of RAs is usually made without evaluating their economic impact. To make informed decisions, it becomes necessary to make a business case in order to know how many instantiations (i.e., applications) are necessary before savings pay off for the up-front investment in building an RA.

The goal of this paper is to present a pragmatic preliminary economic model to perform cost-benefit analysis on the adoption of RAs as a key asset for optimizing architectural decision-making (referred to as REARM, REference ARchitecture Model). This goal is of interest for researchers for the need of formulating accurate models and practitioners for the opportunity of making more informed decision-making about whether to implement the strategic move to RA adoption. Due to the aforementioned lack of research in this specific area, we have aimed at adopting and adapting existing results in related areas, from classical software reuse to product line engineering.

It is worth mentioning that the paper has its origin in an ongoing action-research [36] initiative among our research group and *everis*, a multinational consulting company based in Spain. The architecture group of *everis* experienced the inability to calculate the return on investment (ROI) derived from RAs that they create for organizations. The model stemming from this collaboration is currently under formative evaluation [36], but results so far are already triggering change in some development processes in the organization (e.g., bug reporting). As part of the collaboration, we had the chance to provide an initial validation of the economic model. It comprises a retrospective evaluation of an RA created by *everis* for the IT department of a public administration center in Spain.

2 Background and related work

Current research on RA evaluation consists of analysis methods [2][17][21] that involve the analysis of risks, non-risks, benefits and trade-offs. Although they facilitate the analysis of those aspects based on the most important and critical scenarios, they have little support to analyze the cost and benefits of RAs based on economics.

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

Up to our knowledge, there is no specific economic model for estimating whether it is worth or not to invest in an RA for an organization. Due to the lack of research in this specific area, we have aimed at adopting and adapting existing results in related areas: economic models for software product lines (SPL), cost-benefit analysis methods for SAs, and more generic metrics about cost savings.

Economic models for software product lines and software reuse. The terms RA and product line architecture (PLA) are sometimes used indistinctly inside the SPL engineering context, in which the term RA is used to refer to “a core architecture that captures the high-level design for the applications of the SPL” [33, p. 124] or “just one asset, albeit an important one, in the SPL’s asset base” [9, p. 12].

However, out of the SPL context, RA and PLA are considered different types of artifacts [3][12][19][28]. In Fig. 1 we show the main similarities and differences:

- PLAs are RAs whereas not all RAs are PLAs [3], i.e. PLAs are one type of RAs [19]. PLAs are just one asset of SPL [9, p. 12].
- RAs are more generic and abstract than PLAs that are more complete architectures [3][19]. Hence, “RAs can be designed with an intended scope of a single organization or multiple organizations that share a certain property” [3] whereas PLAs are produced for a single organization [19].
- RAs provide standardized solutions for a broader domain (i.e., “spectrum of systems in a technology or application domain” [19]) whereas PLAs provide standardized solutions for a smaller subset of the software systems of a domain [28] (i.e., “group of systems that are part of a product line” [19]). Therefore, PLAs give a coherent and more congruent view of the products in a project (i.e., possible to track the status of) [12] whereas by means of RAs it is more difficult to obtain congruence [3], since they can only provide guidelines for applications’ development.
- PLAs specifically address points of variability and more formal specification in order to ensure clear and precise behavior specifications at well-specified extension points [3]. In contrast, RAs have less focus on capturing variation points [3][12][28]. Although variability is not typically addressed by RAs in a systematic manner, it is also a key fact for RAs [18], and it can be treated as a quality attribute, rather than explicitly as ‘features’ and ‘decisions’ [18].
- RAs include “the reuse of knowledge about software development in a given domain, in particular with regard to architectural design” [28] and dictate the patterns and principles to implement, i.e. “what the design should be” [12]. Conversely, PLAs specifically indicate deviations, i.e. “what the design is” [12].
- RAs include architectural knowledge and the instantiation of this architectural knowledge (i.e., reference model) into software elements [5]. In this sense, both RAs and PLAs are “a superset, a tool box, with every possible architecture element described, which can be used in the design of a product architecture” [12].

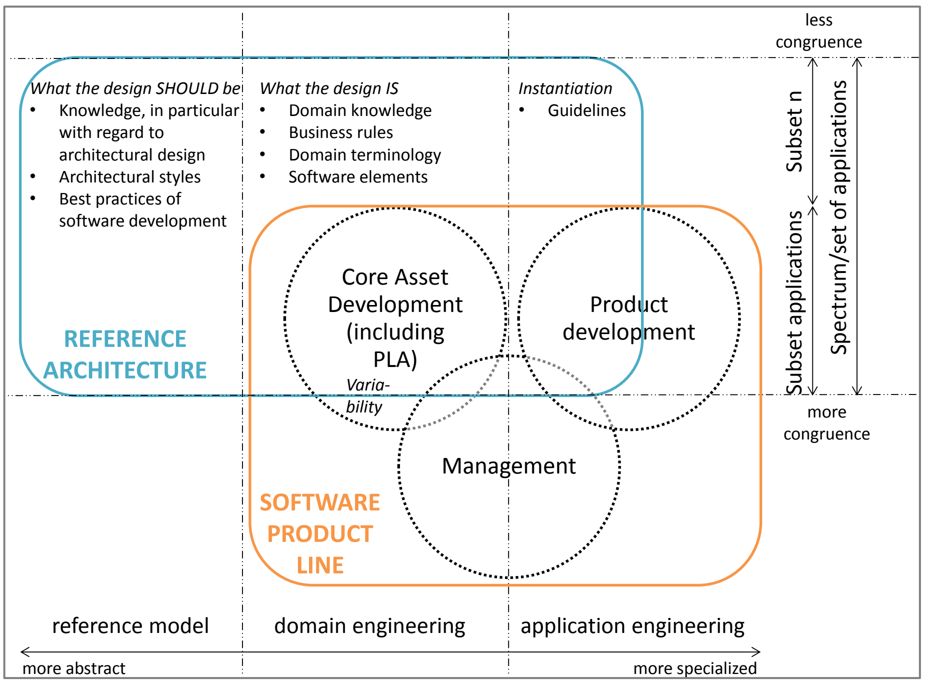

Fig. 1. Similarities and differences between RAs, PLAs, and SPLs.

Although we also consider that RA and PLA are different, some perceived benefits of RA (e.g., cost saving from reusing software elements) and cost-benefit factors (e.g., common software costs, unique development costs) are applicable to both, since both have reuse as their core strategy. For this reason, we studied the applicability of some economic models originally conceived for SPL to RAs. Below, we summarize our results with respect to cost and benefit factors. To see more models, the reader is referred to [1], in which Ali et al. surveyed twelve economic models for SPL, and to [16][27] in which the authors surveyed economic models for software reuse.

Cost and Benefit Factors of Economic Models. SIMPLE [10], Poulin's model [34], and COPLIMO [7] are some of the most widespread economic models for SPLs.

SIMPLE [10] comprises a set of seven cost factors:

- C_{org}, upfront investments to establish a SPL infrastructure.
- C_{cab}, the cost to build reusable assets of the SPL.
- C_{unique}, the cost to develop unique parts of products in a SPL.
- C_{reuse}, the cost of reusing reusable assets in a product inside the SPL.
- C_{cabu}, the cost to evolve the core asset in a SPL.
- C_{prod}, the cost to build a product in a stand-alone fashion.
- C_{evo}, the cost to evolve a product in a stand-alone fashion.

These cost factors and benefit functions can be used to construct equations that can answer a number of questions such as whether the SPL approach is the best option for development and what is the ROI for this approach. Ganesan et al. extended SIMPLE by considering infrastructure degeneration over time [20].

On the other hand, Poulin [34] and Boehm et al. [7] base their reuse-based models in two parameters: RCR and RCWR.

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

For those cases in which there are difficulties to obtain historical data of building and evolving products in a stand-alone fashion (C_{prod}, C_{evo}), we consider more adequate the use of RCR and RCWR (see Section 4.1, step 2).

Finally, we must note two models (Schmid [37], InCoME [30]) that integrate cost and investment models in different layers, which make them more comprehensive.

Value of software architecture design decisions. There exist a few economics-based SA analysis methods that drive the decision-making process during SA review and design. In this direction, CBAM [22] is a useful method for prioritizing architectural decisions that bring higher value. In addition, Ozkaya et al. proposed an economic valuation of architectural patterns [32].

These approaches help to find the optimal set of decisions that maximizes the ROI [15]. They pursue to solve the same problem of this paper, but their scope is broader and general for any kind of SA decision and do not reflect fundamental characteristics of adopting an RA. Therefore, their applicability for studying the ROI of RA adoption would require more effort, since specific cost-benefit factors for architecture-centric reuse are not considered. Hence, they are not the most convenient approaches for making the business case of adopting an RA and calculating its payback time.

Generic software metrics. There exist several approaches that propose metrics for estimating cost savings in software development and maintenance. Metrics as dependency structure matrices (DSM) have been applied to assist architecture-level analysis, such as value of modular designs, and they have proven to be particularly insightful for validating the future value of architecting modular systems [8]. MacCormack et al. extracted coupling metrics from an architecture DSM view for inferring the likelihood of change propagation and, hence, future maintenance costs [25]. Baldwin et al. presented a generic expression for evaluating the option to redesign a module also based on DSMs [4].

In addition, the concept of technical debt (either architecture-focused [31] or code-based [24]) is a way to measure unexpected rework costs due to expediting the delivery of stakeholder value in short.

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

- Validation in industry. “Very few [economic models for SPL] actually have been used as a basis for further development or adopted in industry” [23]. Thus, “there is a clear need for many more empirical studies to validate existing models” [1].
- Easy adoption of models in industry by identifying realistic metrics to collect and report. “It is difficult for the practitioners to evaluate the usability and usefulness of a proposed solution [economic model for SPL] for application in industry” [23]. Not guidelines exist to fully operationalize the models in practice [37].

Economics-driven SA analysis methods do not specifically aim at making an investment analysis of the adoption of an architecture-centric program. RA adoption is a subarea inside their generic decision-making context.

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

This state of the art drove us to the formulation of an economic model for RAs, which is currently on its formative stage. The formulation of the model aims to:

- Adapt cost and benefit factors from SPL models that are easy-to-apply by industry. The goal is to provide guidelines to fully operationalize the model in practice.
- Fill the gap of RA economics inside the SA decision-making context.
- Look for generic software metrics that can quantify new cost and benefit factors.

3 Industrial context

The architecture group of *everis* is an initiative to manage architectural knowledge, best practices and lessons learned from previous experiences; and to provide efficient solutions to a better cost, flexibility and agility to the demands of client organizations.

This architecture group offers solutions for big businesses (e.g., banks, insurance companies, public administration and service, and industrial organizations) that offer a wide spectrum of services to their clients. Often, already existing commercial packages are not completely aligned with the business needs of these organizations, thereby requiring custom development and maintenance of applications. In this scenario, *everis* foster the use RAs for managing a wide spectrum of applications.

The architecture group of *everis* experienced the inability to calculate the ROI derived from RAs that they create for organizations. The purpose of our research is to create a method for extracting costs and benefits of RAs based on data that they were already collected.

4 An economic model for reference architectures

4.1 Method for formulating the economic model

An RA cost-benefit analysis should be based on giving an economic value to its activities. We designed our economic model through the three following steps:

1. **Identify the costs and benefits stemming from the use of an RA.** Although cost modeling is already a mature field within software engineering, benefits have traditionally been far more elusive to quantify [8]. For this reason, it is necessary to identify the RA quality attributes that bring more benefit to the development and maintenance of applications, and the costs of constructing these applications [22]. These attributes may vary depending on the architecturally-significant requirements coming from the applications based on the RA. It is crucial to involve relevant stakeholders to ensure the trustworthiness of the collected information [38].

   The outputs of this step are, therefore, the costs factors of adopting an RA and the list of quality attributes in which the RA brings more benefit.
2. **Adopt metrics that quantify the costs and benefits identified in the first step in order to convert them into a monetary value.** The metrics to quantify them may vary depending on the data available in the organization involved.

   The output of this step is providing guidelines for collecting simple metrics that make possible to calculate the cost and benefits factors in practice.
3. **Make the business case for the adoption of the RA.** Add the costs and benefits calculated in the second step to the formula for calculating the ROI (proposed by Boehm [6]), where the benefits are the improvements of applications quality attributes, and the costs are the expenses in constructing the systems and the RA.

   The output of this step is a business case that captures the reasoning for adopting an RA. The RA business case analysis involves determining the relative financial costs, benefits, and ROI across its life-cycle.

$$\text{ROI} = \frac{\text{Benefits - Costs}}{\text{Costs}} \tag{1}$$

4.2 Execution of the method for formulating the economic model

The action-research collaboration with *everis* provided us the opportunity of implementing this general-purpose method in a particular case.

Step 1. We conducted a survey involving project managers, architects and developers of 9 organizations in Europe (7 from Spain) [26]. The survey pointed out that the main perceived economic benefits on the use of RAs were: (1) an increased value from the improvement of quality attributes, since their reused architectural knowledge is incrementally improved with previous successful experiences from its application domain; (2) cost savings in the development and maintenance of systems due to the reuse of software elements and the adoption of best practices of software development that increase the productivity of developers. Therefore, RAs bring most of the benefit because of the improvement of reusability and maintainability quality attributes. One of the reasons why RAs were adopted in these organizations is that the most important architecturally-significant requirement was reusability. Thus, we decided to focus our cost-benefit analysis over reusability and maintainability.

We found that some of the potential metrics to be used were not as pragmatic as the organization needed. In other words, the organization should have been invested extra time which was not an option. Furthermore, we faced the problem that some of the required data to apply the proposed metrics was not previously registered by the organization. Thus, we stressed the emphasis on formulating a practical model that incrementally deals with diverse cost-benefit aspects.

We identified six cost-benefit factors for RA adoption. We started the formulation of factors by adopting Poulin's method for measuring code reuse [34][35]. We adapted Poulin's model because it has been applied in industry, offers parameters to operationalize it, and we could feed it with available data in *everis* (see Step 2 below). We adopted its benefit factors (DCA, SCA) published in [35]. Conversely, we consider more appropriate for RAs to adopt the cost factors defined for SPL (CSW_{dev_costs}, $CSW_{service\_costs}$) in [35], instead of the additional development costs [34].

To complete the model we add the unique development costs of applications. Also, with the help of the propagation cost metric [25], we also consider necessary changes to reusable elements (which are not considered by Poulin's method) and, therefore, evolution. These two new factors include parameters to operationalize them.

The former three factors are for development and the latter ones for maintenance:

- DCA (Development Cost Avoidance). It is the *benefit* from reusing RA's software modules in applications compared to building the applications independently.
- UDC (Unique Development Costs). It is the *cost* to develop the unique parts of an application that are not already implemented in the modules of the RA. UDC is equivalent to $C_{reuse}+C_{unique}$.
- CSWD (Common Software Development costs). It is the *cost* of the initial investment, i.e., developing an RA. CSWD is equivalent to $C_{org}+C_{cab}$.
- SCA (Service Cost Avoidance). It is the *benefit* of modifying reused code once.
- CSWS (Common Software Service costs). It is the *cost* of fixing bugs in the (reusable) RA modules. CSWS calculates the cost of changes due to bugs in C_{cabu}.
- CSWE (Common Software Evolution costs). It is the *cost* of changing or adding functionalities to the RA modules. CSWE calculates the cost of evolutions in C_{cabu}. Therefore, CSWS+CSWE are equivalent to C_{cabu}.

Putting everything together, given a number *n* of applications built in top of the RA, and a number *m* of RA modules changed as it evolves, the benefits and costs of adopting an RA are defined as:

$$\text{Benefits}=\sum_{i=1}^{n}(DCA_i+SCA_i) \quad (2)$$

$$\text{Costs}=CSWD+CSWS+\sum_{i=1}^{n} UDC_i+\sum_{j=1}^{m} CSWE_j \quad (3)$$

Step 2. We divide the second step in two activities: checking the data available in practice and guide the information extraction from this data.

Data commonly available in practice that should be collected. The data that typically is available in order to calculate the aforementioned costs and benefits are effort and software metrics [26]. It allows converting cost-benefit factors into a monetary value.

On the one hand, the invested effort from the tracked activities allows the calculation of costs. We distinguish between three types of activities: training, development and maintenance. JIRA[1] and Redmine[2] are tools that support keeping track of activities and their invested time. Keeping track of activities is common in practice for project management and auditing. Activity tracking is also known as tickets [8].

On the other hand, software metrics help to analyze the benefits that can be found in the source code. For example, since the cost of applications' development is lower because of the reuse of RA, we estimate the cost avoidance of reusing its LOC. Sonar[3] offers tool support for obtaining general software metrics such as LOC, dependencies between modules, technical debt [24], and percentages of tests and rules compliance.

Software development is a naturally low-validity environment and reliable expert intuition can only be acquired in a high-validity environment. As stated by Erdogmus and Favaro [14], the adoption of practices like time tracking and tools to collect data is the basis for moving software development from its usual low-validity environment, to a high-validity environment.

We experienced difficulties collecting historical data (as in [20]), especially for the "before" state of adopting an RA. We noted that C_{prod} and C_{evo} were seldom available since the "before" state did not exist. For this reason, we use RCR and RCWR.

Using commonly available data in practice to quantify the costs and benefits. In Table 1, we present ten basic parameters that are required for calculating the six cost-benefit factors of the Step 1. Table 2 shows the formulas to calculate this six cost-benefit factors as well as parameters that are needed for these calculations.

Table 1. Basic parameters in order to feed the factors of Table 2.

	Description of the parameters (adapted for the RA context)
RCR	Relative Cost of Reuse: effort that it takes to reuse a component without modification versus writing it new one-at-a-time [34]
RCWR	Relative Cost of Writing for Reuse: effort that it takes to write a reusable component versus writing it for one-time use only [34]
ER	Error Rate: the historical error rate in new software developed by your organization, in errors per thousand lines of code [34]
EC	Error Cost: your organization's historical cost to fix errors after releasing new software to the customer, in euros per error [34]
NMSI	New Module Source Instruction: the LOC that the changed or new module has, which can be the average of previous ones
PC	Propagation Cost: the percentage of code affected in the RA when performing evolutions (i.e., changing modules) [25]
CPKL	Cost per KLOC: the historical cost to develop a KLOC of new software in your organization [34]

1 JIRA, http://www.atlassian.com/es/software/jira/overview

2 Redmine, http://www.redmine.org/

3 Sonar, http://www.sonarsource.org/

USI	Unique Source Instructions: the amount of unique software (i.e., not reused) that was written or modified for an application
RSI	Reused Source Instructions: it is the total LOC of the RA's modules that are reused in an application. It supports variability. In other words, reuse of RA might not be complete but partial, since different applications can reused different RA's modules. Therefore RSI depend on each application [34].
TSI	Total Source Instructions: it is the total LOC of the RA that can be reused [34].

Table 2. Cost-benefit factors to calculate the ROI of adopting an RA in an organization.

	Description of the cost-benefit factors (adapted for the RA context)
DCA	Development Cost Avoidance: the benefits from reusing RA's modules [34] DCA = RSI * (1-RCR) * CPKL
CSWD	Common Software Development costs: the costs to develop the RA [35] CSWD = RCWR * TSI * CPKL
UDC	Unique Development Costs: the costs to develop the unique part of an application UDC = USI*CPKL
SCA	Service Cost Avoidance: benefits from maintaining only once RA's modules [34] SCA = RSI * ER * EC
CSWS	Common Software Maintenance costs: cost of fixing bugs in reusable modules [35] CSWS = TSI * ER * EC
CSWE	Common Software Evolution costs: the costs of changing or adding a new functionality and maintaining it to the RA CSWE = evolution development + evolution maintenance + propagation = (NMSI*RCWR*CPKL)+(NMSI*ER*EC)+(TSI*CPKL*PC)

Step 3. As final step, we can use calculated factors in order to calculate the ROI:

$$\text{ROI} = \frac{[\sum_{i=1}^{n}(\text{DCA}_i+\text{SCA}_i)]-[\text{CSWD}+\text{CSWS}+\sum_{i=1}^{n}\text{UDC}_i+\sum_{j=1}^{m}\text{CSWE}_j]}{\text{CSWD}+\text{CSWS}+\sum_{i=1}^{n}\text{UDC}_i+\sum_{j=1}^{m}\text{CSWE}_j} \qquad (4)$$

We also suggest using these cost-benefit factors to make a business case for calculating the ROI of building an RA vs. building the applications independently. Table 3 shows an example of business case and how to calculate the cost and benefits for three years since the RA adoption. The parameters n_1, n_2, n_3 indicate the number of applications developed per year respectively, and m the number of evolved modules.

Table 3. Example of design of a business case with the cost-benefit factors of the model

	Year 1	**Year 2**	**Year 3**
Total benefit	n_1*(DCA+SCA)	n_2*(DCA+SCA)	n_3*(DCA+SCA)
Total cost	CSWD+ n_1*UDC+CSWS*$^1/_5$	n_2*UDC+ CSWS*$^2/_5$+m*CSWE	n_3*UDC+ CSWS*$^2/_5$+m*CSWE

As Boehm points out [6], two additional factors may be important in business case analysis: unquantifiable benefits, and uncertainties and risk.

First, the economic model that we propose promotes benefits in reusability and maintainability. However, other quality attributes, such as security, could be as relevant as those for this analysis, even when they may be difficult to quantify. This other benefits should also been taken into account when adopting and RA. Unquantifiable benefits are also considered as “flexibility” in TEI[4], the economic model of Forrester.

Second, to adjust cost and benefits to risk, they can be multiplied by percentages that generally increase the costs and reduce the benefits (assuming the worst case). For instance, TEI propose to multiple costs by values that range from 98% to 150% and benefits by values between 50% and 110%.

5 Preliminary validation

To assess the feasibility of the economic model, we conducted a retrospective analysis of a particular case. We calculated the costs and benefits (and hence the ROI) of an RA adoption driven by *everis* in the IT department of a public organization.

By the time we performed the validation, the public organization had already: (1) adopted an RA, (2) created an application using the RA –which we consider “exemplar” application–, and (3) fixed errors discovered in the RA software elements that were reused by the application.

The validation consisted of 4 parts. First, a post-mortem analysis in which our challenge was to extract the parameters of Table 1 from already collected data. The values that we got are shown in Table 4.

Table 4. Values of the basic parameters in the study.

RCR	RCWR	ER	EC	NMSI	PC	CPKL	USI	RSI	TSI
0,064	1,243	2,879 err./kLOC	7,02 hours/err.	1.526 LOC/module	9,7 %	75,22 hours/kLOC	2.885 LOC	8.364 LOC	41.189 LOC[*]

[*] In TSI, 9.231 LOC were refactored from previous project. So, 31.958 were new.

Recommended values for RCR range from 0,03 and 0,25, and for RCWR from 1 to 2,2 [34]. Therefore, with the values that we got in the study, we can see that both RCR and RCWR are low for RAs. A low RCR could show the trend of moving the complexity to the architecture in order to simplify the development of applications. We can also see this trend comparing the code of the RA software elements with the code of applications. RA code present higher values for complexity metrics such as coupling and cohesion. A reason why RCWR is low could be that RA architectural knowledge speeds up the development.

Second, with the data of Table 4, we had real data to calculate (see Table 5):

- CSWD, the RA initial investment, which lasted 6 months.
- DCA, the benefit of reusing RA code in the exemplar application development.
- SCA, the cost from fixing the errors of the reused code in the exemplar application.
- UDC, the cost of developing the application.

[4] Total Economic Impact, http://www.forrester.com/marketing/product/consulting/tei.html

The above costs were accurately computed because *everis* keeps track of activities with their invested time. Third, it was necessary to estimate the rest of factors:

- CSWS, the cost of fixing all bugs in RA code. Since we knew the SCA for the exemplar application and the percentage of reuse, we calculated the error rate and error cost, which we used to estimate CSWS.
- CSWE, the cost of: (1) changing or developing a module with new functionality, (2) fixing its bugs, (3) making changes in the rest of the RA to integrate it.

Table 5. Values of the cost-benefit factors in the study[*].

DCA	CSWD	UDC	SCA	CSWS	CSWE
589 hours	**2.988 hours**	**217 hours**	**169 hours**	*832 hours*	*474 hours*

[*] Values in **bold** are real data. Values in *italic* are estimated.

Fourth, we made the business case analysis with two different scenarios:

Scenario 1. *Is it worth to invest on the adoption of an RA?* We constructed a business case for 3 years starting when the organization decided to adopt the RA, in order to calculate the ROI. For the first 8 months of those 3 years, we have real data about the RA development and the exemplar RA-based application. To estimate the costs and benefits for the rest of these 3 years, we conducted some additional interviews to the involved stakeholders. Stakeholders were carefully selected according to their knowledge and experience to increase the degree of confidence on the data gathered. After these interviews, we made the following assumptions:

- Future applications will have similar characteristics and complexity as the exemplar one.
- The public organization will develop 8 applications per year. Since the RA creation lasted 6 months, the first year they will develop just 4 applications.
- The totality of CSWS is computed proportionally starting the seventh month.
- A module is evolved (with new functionality) or added to the RA every year since the second year.

Under these assumptions, the costs and benefits in hours for the future can be calculated as shown in Table 3. They can be converted into a monetary value by multiplying them by an hourly rate. Assuming a rate of 30€ per hour for an application developer (which affects to DCA, SCA, UDC) and a rate of 40€ per hour for a developer and maintainer of the architecture (which affects to CSWD, CSWM, CSWE), Fig. 2 summarizes financial results for first three years of the RA. This organization will realize a ROI within 2 years through gains in systematic reuse.

Scenario 2. *How many instantiations (i.e., applications) are necessary before savings pay off for the up-front investment in building an RA?* In this scenario we calculated how many applications need to be build based on the RA to have a positive ROI. Fig. 3 shows the ROI due to developing and maintaining applications based on an RA rather than in a stand-alone fashion.

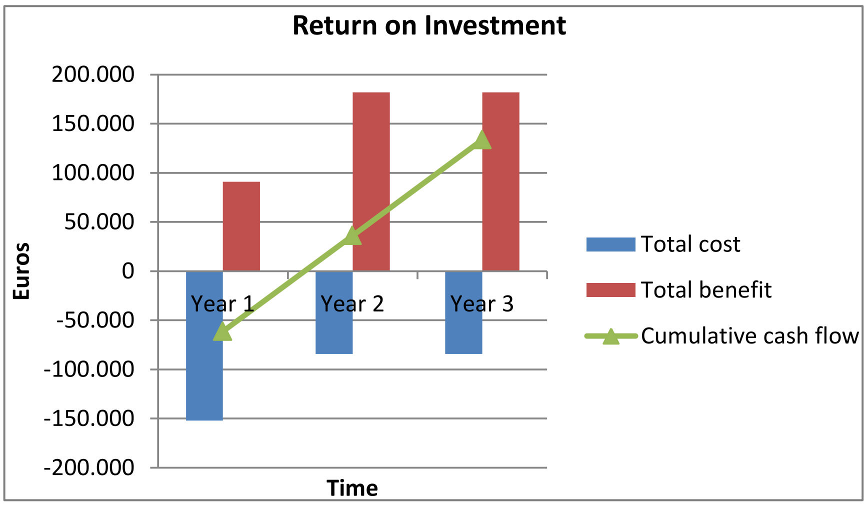

Fig. 2. Summary financial results.

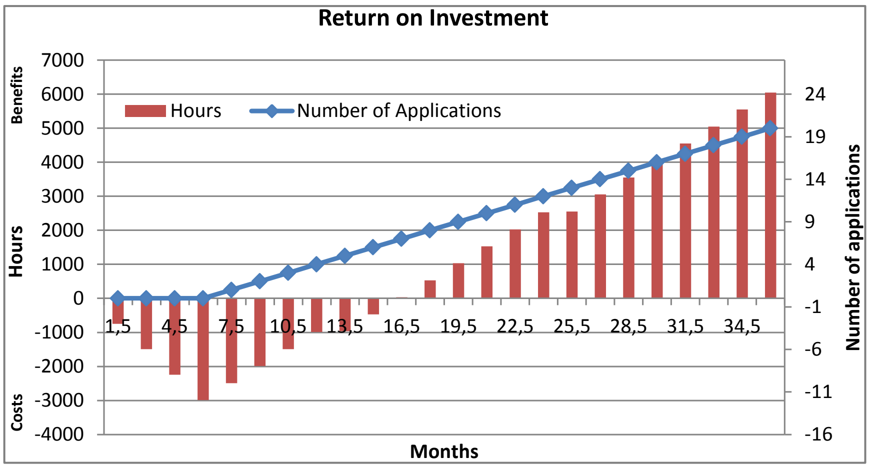

Fig. 3. ROI of developing and maintaining RA-based applications vs. stand-alone fashion.

As Fig. 3 shows, after building 7 applications, savings pay off for the up-front investment in the RA. It must be noted that the exemplar application is small and only 20% of the RA is being reused (RSI/TSI). On the other hand, the application has a high reuse percentage of 74% (RSI/USI+RSI). The higher these percentages are (likely in medium to large applications), the greater the benefit from the RA is.

Moreover, applications are introduced into the market earlier from the seventh month on. This is due to the effort avoidance of 589 hours (DCA) of reusing the RA.

To sum up, this study illustrates the potential way in which an organization can evaluate the value of RA adoption. We calculated a three-year ROI of 42% with a payback period of 16,5 months and 7 applications.

6 Discussion

Once we applied the economic model and calculated the ROI, a last question remains: How accurate are these calculations and the obtained quantitative data? If the economic model is applied with existing data (as we have done in Section 5), the calculation of the ROI reaches a high degree of correctness, since the data that feeds the model is trustworthy. The metrics coming from code analysis (e.g., size in LOC) do not reflect any error. Also, we saw that time tracking is reliable. During data collection we found invested time in activities in two different sources: JIRA, which is optionally used by the project team and keeps the invested time of the project's activities; and a mandatory corporate financial tool, which is used by the financial department. This data differ in 8,75%, being lower internally time tracking of the project. The reason could be that JIRA does not include other activities out of the scope of the project like traveling. To adjust the calculations to this risk, we have always considered the worst case (i.e., greater costs).

Contrary, when the economic model is used to predict the ROI of a completely new RA adoption in an organization, there is not real data since it does not exist yet. In this case, the accuracy totally depends on expert intuition and historical data. Historical data can be scarce in small and medium organizations; especially considering that reuse of architectures is still a research area in progress. In addition, historical data must be continuously updated, since some values of effort-related parameters (such as RCR) are expected to decrease each time a developer instantiates the RA.

As a final remark, the construction of an economic model from the data available in software companies is yet-another-instance of research question which needs to balance soundness with applicability. The awareness of this problem by the software engineering community is increasing and even dedicated events are being organized (See CESI 2013 @ ICSE, http://www.essi.upc.edu/~franch/cesi2013/).

7 Conclusions & next steps

Architecture improvements are extremely difficult to evaluate in an analytic and quantitative way, contrary to business efficacy (sales, marketing, and manufacturing) [8]. Methods and models for changing this state of the practice are demanded.

This paper has opened the path on the area of using economic models for RA assessment. We think that this area has a significant impact not just for researchers but also for practitioners in software development and organization's executives. We presented REARM, an economic model to translate measured or estimated data (i.e., metrics) into monetary terms (i.e., cost-benefit analysis). Then, we use them as the basis for analyzing the economic value of an RA (i.e., valuation) that is adapted by an organization in the pursuit of its business strategies. Thus, our work aligns with Erdogmus et al. vision on economic activities in software industry, that fall into 4 levels: metrics, cost-benefit analysis, valuation and business strategy [13].

We have conducted a preliminary validation to calculate the ROI of adopting an RA in a real organization. This organization will realize a return on their investment within two years through gains in systematic reuse and applications maintainability. The method presented is generic enough to be used when other quality attributes are prioritized by relevant stakeholders. The presented economic model allows quantifying the value that an RA of Type 2 or 4 (those designed with an intended scope of a single organization) [3] brings to an organization. Its strongest points are:

- It translates RA costs and benefits into monetary values, which can be considered an innovative approach in RA research and practice.
- The integration of different metrics from existing models that complement each other evaluating several RA-relevant aspects.
- It provides guidelines for easily collecting and reporting data for practitioners, and for using it to make a business case.
- The model has been applied in a public organization and validated with real data.

On the other hand, potential weaknesses of this approach are:

- It does not consider RA's software elements degeneration over time [20].
- The risk increases when neither real nor historical data are available.

As future work, we plan to enrich the economic model by: (1) adding more metrics (such as technical debt [24], degeneration over time [20], risk metrics, homogeneity metrics [10]), and (2) validating it for bigger applications and in more organizations.

Acknowledgements. This work has been supported by "Cátedra *everis*" and the Spanish project TIN2010-19130-C02-00. We would also like to thank all participants of the data collection process for their kindly cooperation.

References

1. Ali, M., Babar, M., Schmid, K.: A comparative survey of economic models for software product lines. In: Software Engineering and Advanced Applications, 2009. pp. 275-278.
2. Angelov, S., Trienekens, J., Grefen, P.: Towards a method for the evaluation of reference architectures: Experiences from a case. ECSA, 2008.
3. Angelov, S., Grefen, P., Greefhorst, D.: A framework for analysis and design of software reference architectures. Information and Software Technology 54(4), 417-431, 2012.
4. Baldwin, C.Y., Clark, K.B.: Design Rules: The Power of Modularity. MIT Press, 1999.
5. Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice. Addison-W, 2003.
6. Biffl, S., Aurum, A., Boehm, B., Erdogmus, H., Grünbacher, P.: Value-Based Software Engineering. Springer-Verlag, 2005.
7. Boehm, B., Brown, A., Madachy, R., Yang, Y.: A software product line life cycle cost estimation model. In: Empirical Software Engineering, 2004. pp. 156-164.
8. Carriere, J., Kazman, R., Ozkaya, I.: A cost-benefit framework for making architectural decisions in a business context. In: ICSE. vol. 2, pp. 149-157, 2010.
9. Clements, P., Northrop, L.: Software Product Lines. Addison-Wesley, 2002.
10. Clements, P., McGregor, J., Cohen, S.: The structured intuitive model for product line economics (SIMPLE). Tech. rep., DTIC Document, 2005.

11. Cloutier, R., Muller, G., Verma, D., Nilchiani, R., Hole, E., Bone, M.: The concept of reference architectures. Systems Engineering 13(1), 14-27, 2010.
12. Eklund, U., Jonsson, N., Bosch, J., Eriksson, A.: A reference architecture template for software-intensive embedded systems. In: WICSA/ECSA, 2012.
13. Erdogmus, H., Favaro, J., Strigel, W.: Return on investment. IEEE Software 21(3), 2004.
14. Erdogmus, H., Favaro, J.: The Value Proposition for Agility–A Dual Perspective, 2012. http://www.infoq.com/presentations/Agility-Value
15. Falessi, D., Kruchten, P., Cantone, G.: Issues in applying empirical software engineering to software architecture. In: LNCS, vol. 4758, pp. 257-262, 2007.
16. Frakes, W., Terry, C.: Software reuse: metrics and models. Comput. Surv. 28(2), 1996.
17. Gallagher, B.P.: Using the architecture tradeoff analysis method to evaluate a reference architecture: A case study. Technical Report CMU/SEI, 2000.
18. Galster, M., Avgeriou, P.: Empirically-grounded reference architectures: a proposal. In: Proceedings of the joint ACM SIGSOFT conference QoSA/ISARCS, 2011.
19. Galster, M., Avgeriou, P., Tofan, D.: Constraints for the design of variability intensive service-oriented reference architectures an industrial case study. IST 55(2), 428-441, 2013.
20. Ganesan, D., Muthig, D., Yoshimura, K.: Predicting return-on-investment for product line generations. In: SPLC, pp. 13-24, 2006.
21. Graaf, B., van Dijk, H., van Deursen, A.: Evaluating an embedded software reference architecture. In: CSMR, pp. 354-363, 2005.
22. Kazman, R., Asundi, J., Klien, M.: Making architecture design decisions: An economic approach. Tech. rep., DTIC Document, 2002.
23. Khurum, M., Gorschek, T., Petersson, K.: Systematic review of solutions proposed for product line economics. Decision Support Software Intensive Product Management, 2008.
24. Letouzey, J.: The sqale method for evaluating technical debt. In: MTD@ICSE, 2012.
25. MacCormack, A., Rusnak, J., Baldwin, C.: Exploring the duality between product and organizational architectures. Harvard Business School Research Paper (08-039), 2011.
26. Martínez-Fernández, S., Ayala, C., Franch, X., Ameller, D.: A Framework for Software Reference Architecture Analysis and Review. ESELAW@CIbSE, 2013, (in press).
27. Mili, A., Chmiel, S., Gottumukkala, R., Zhang, L.: An integrated cost model for software reuse. In: ICSE, pp. 157-166, 2000.
28. Nakagawa, E., Oliveira Antonino, P., Becker, M.: Reference architecture and product line architecture: A subtle but critical difference. ECSA, pp. 207-211, 2011.
29. Nakagawa, E.: Reference architectures and variability: current status and future perspectives. In: Proceedings of the WICSA/ECSA, pp. 159-162, 2012.
30. Nóbrega, J., Almeida, E., Meira, S.: Income: Integrated cost model for product line engineering. In: SEAA, pp. 27-34, 2008.
31. Nord, R., Ozkaya, I., Kruchten, P., Gonzalez-Rojas, M.: In search of a metric for managing architectural technical debt. In: WICSA/ECSA, pp. 91-100, 2012.
32. Ozkaya, I., Kazman, R., Klein, M.: Quality-attribute based economic valuation of architectural patterns. In: ESC, 2007.
33. Pohl, K., Bockle, G., Van Der Linden, F.: Software product line engineering, vol. 10, 2005
34. Poulin, J.: Measuring Software Reuse. Reading, MA: Addison-Wesley, 1997.
35. Poulin, J.: The economics of product line development. International Journal of Applied Software Technology 3, 15-28, 1997.
36. Robson, C.: Real world research, vol. 2. Blackwell Oxford, 2002
37. Schmid, K.: An initial model of product line economics. SPFE pp. 198-201, 2002.
38. van Solingen, R.: Measuring the ROI of software process improvement. Software, IEEE 21(3), 32-38, 2004.