A Reuse-Based Economic Model for Software Reference Architectures

Silverio Martínez-Fernández[1], Claudia Ayala[1],
Xavier Franch[1]

[1] Software Engineering for Information Systems Group
Universitat Politècnica de Catalunya (GESSI-UPC)
Barcelona, Spain
{smartinez, cayala, franch}@essi.upc.edu

November 2012

Report ESSI-TR-12-6

Departament d'Enginyeria de Serveis i Sistemes d'Informació

A Reuse-Based Economic Model for Software Reference Architectures

Silverio Martínez-Fernández, Claudia Ayala, Xavier Franch
Universitat Politècnica de Catalunya (UPC)
Barcelona, Spain

***Abstract*—The growing size and complexity of software systems, together with critical time-to-market needs, demand new software engineering approaches for software development. To remain competitive, organizations are challenged to make informed and feasible value-driven design decisions in order to ensure the quality of the 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 opening a path in this direction by presenting a pragmatic preliminary economic model to perform cost-benefit analysis on the adoption of software reference architectures as 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 incorporated as our research progresses.**

***Index Terms*—Software architecture, reference architecture, architecture evaluation, cost-benefit analysis, quality attributes.**

I. Introduction and Motivation

Nowadays, the size and complexity of software systems, together with critical time-to-market needs, demand new software engineering (SE) approaches to software development. One of these approaches is the use of software reference architectures (RA), which are becoming widely studied and adopted in SE research and practice [3]. An RA encompasses the knowledge about how to design concrete software architectures (SA) of systems of a given application domain; therefore, it must address the business rules, architectural styles, best practices of software development, and the software elements that support development of systems [18].

The motivation behind RAs is to systematically reuse knowledge and software elements when developing concrete SA for new systems, to help with the evolution of a set of systems that stem from the same RA, or to ensure standardization and interoperability. 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 [11].

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 [19]. In particular, 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" [6]. Thus, the adoption of RAs is usually made without evaluating their economic impact.

The goal of this paper is to set a new research direction on RA evaluation attracting both: researchers for the need to formulate accurate models, and practitioners for the opportunity of making more informed decision-making about whether to make 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 initiative among our research group and the Center of Excellence on Software Architectures (ARCHEX) of Everis, a multinational consulting company based in Spain. ARCHEX 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, but the current results are already triggering change in some development processes in the organization (e.g., bug reporting and management). 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.

II. BACKGROUND

Current research on RA evaluation [2][10][12] has little support to analyze the cost and benefits of RAs based on economics. However, there exist a few SA evaluation methods that drive the decision-making process during SA review and design. In this direction, Moore et al. presented CBAM [17], Ozkaya et al. proposed an economic valuation of architectural patterns [21], and Ivanovic et al. quantified the customer value that stems from quality improvements [13]. Although these SA methods are useful for prioritizing architectural strategies that bring higher value, they can be applied only in a restricted way for RAs adoption. RAs involve fundamental assumptions that these methods do not reflect, especially when it comes to defining potential economic benefits and the payback time.

Another inspiring area is Software Product Lines (SPL). A survey by Ali et al. [1] summarized twelve economic models for SPL. Among them, we may remark SIMPLE [7]. SIMPLE comprises a set of cost and benefit functions that 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.

On the other hand, there exist several approaches that propose metrics for estimating cost savings in software development and maintenance. For instance, Poulin considered the reuse of assets in developing individual applications, and the potential costs savings that it implies to the development and maintenance [22]. 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 [6]. MacCormack et al. extracted coupling metrics from an architecture DSM view for inferring the likelihood of change propagation and, hence, future maintenance costs [15]. Baldwin et al. presented a gene-ric expression for evaluating the option to redesign a module also based on DSMs [4] that is used in [24]. In addition, the use of technical debt (either architecture-focused [20] or code-based [14]) is a way to measure unexpected rework costs due to expediting the delivery of stakeholder value in short.

Despite the existence of this inspiring body of work from the SPL, software reuse and modular design areas, the proposed models are not directly applicable to RAs. Although SPL and RAs have similarities (both have reuse as their core strategy) they have also important differences. RAs deal with the range of knowledge of an application domain, providing standardized solutions for such a broad domain, while SPL approaches are more specialized, focusing sometimes on a specific subset of the systems of a domain and providing standardized solutions for a smaller family of systems [18].

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

III. An Economic Model for Reference Architectures

A. Method for formulating the model

An RA cost-benefit analysis should be based on giving an economic value to its activities. This can be done in three steps:

1) Identify the costs and benefits stemming from the use of an RA: It is necessary to identify the RA quality attributes that bring more benefit to the development and maintenance of systems, and the costs of constructing these systems [17]. These attributes may vary depending on the characteristics of RA [3]. It is crucial to involve relevant stakeholders to ensure the trustworthiness of the collected information [23].

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.

3) Add the costs and benefits calculated in the second step to the formula for calculating the ROI:

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

where the *benefits* are the improvements of system quality attributes, and the *costs* are the expenses in constructing the systems and the RA. Eq. 1 has been proposed by Boehm [5].

B. Example of application

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

For the first step, we conducted a survey involving project managers, architects and developers of 9 organizations in Europe (7 from Spain) [16]. 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. These cost savings could be seen as the improvement of reusability and maintainability quality attributes. Thus, we decided to focus our cost-benefit analysis over these particular dimensions.

For the second step, 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. This helped us to prioritize the operationalization of these aspects into the model.

We started the formulation of metrics by adopting Poulin's method for measuring code reuse [22]. With the help of the propagation cost metric [15], we also consider necessary changes to reusable elements (which are not considered by Poulin's method) and, therefore, maintainability. For this purpose, we use six cost-benefit factors, which we classify into development and maintenance:

a) Benefits and costs of development:

- 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.
- ADC (Additional Development Costs). It is the *cost* of developing an RA for a particular organization.

b) Benefits and costs of maintenance:

- SCA (Service Cost Avoidance). It is the *benefit* from modifying the reused code only once.
- AMC (Additional Maintenance Costs). It is the *cost* of fixing bugs in the (reusable) RA modules.
- AEC (Additional Evolution Costs). It is the *cost* of changing or adding functionalities to the RA modules.

Given a number *n* of applications built in top of the RA, and a number *m* of RA modules added or changed as it evolves, the benefits and costs of adopting an RA are defined as:

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

$$\text{Costs} = ADC + AMC + \sum_{i=1}^{n} UDC_i + \sum_{j=1}^{m} AEC_j \quad (3)$$

Table I shows the formulas to calculate the six factors as well as parameters that are needed for these calculations.

As final step, we can use the benefits of Eq. 2 and costs of Eq. 3 into the Eq. 1 in order to calculate the ROI.

TABLE I. PARAMETERS OF THE ECONOMIC MODEL TO CALCULATE THE ROI OF ADOPTING A REFERENCE ARCHITECTURE IN AN ORGANIZATION

Parameter	Description (adapted for the RA context)	Value in the study
RCR [22]	Relative Cost of Reuse: effort that it takes to reuse a component without modification versus writing it new one-at-a-time	0.064
RCWR [22]	Relative Cost of Writing for Reuse: effort that it takes to write a reusable component versus writing it for one-time use only	2.468
ER [22]	Error Rate: the historical error rate in new software developed by your organization, in errors per thousand lines of code	2.015 errors/kLOC
EC [22]	Error Cost: your organization's historical cost to fix errors after releasing new software to the customer, in euros per error	1,036.38 €/error
NMSI	New Module Source Instruction: the LOC that the changed or new module has, which can be the average of previous ones	1,526 LOC/module
PC [15]	Propagation Cost: the percentage of code affected in the RA when performing evolutions (i.e., changing modules)	9.7 %
CPL [22]	Cost per LOC: the historical cost to develop a LOC of new software in your organization	2.5 €
USI	Unique Source Instructions: the amount of unique software (i.e., not reused) that was written or modified for an application	2,885
RSI [22]	Reused Source Instructions: it is the total LOC of the RA's modules that are reused in an application	8,364
TSI [22]	Total Source Instructions: it is the total LOC of the RA that can be reused	41,189 LOC
DCA [22]	Development Cost Avoidance: the benefits from reusing RA's modules → DCA = RSI * (1-RCR) * CPL	19,579 €
ADC [22]	Additional Development Costs: the costs to develop the RA → ADC = (RCWR-1) * TSI * CPL	151,213 €
UDC	Additional Unique Development Costs: the costs to develop the unique part of an application → UDC = USI*CPL	7,212 €
SCA [22]	Service Cost Avoidance: the benefits from maintaining only once RA's modules → SCA = RSI * ER * EC	17,467 €
AMC	Additional Maintenance Costs: the cost of fixing bugs in the (reusable) RA modules → AMC = TSI * ER * EC	86,019 €
AEC	Additional Evolution Costs: the costs of changing or adding a new functionality and maintaining it to the RA → AEC = evolution development + evolution maintenance + propagation = (NMSI*CPL) +(NMSI*ER*EC) +(TSI*CPL*PC)	17,028 €

IV. PRELIMINARY VALIDATION

To assess the feasability of the economic model, we con-ducted a retrospective analysis of a particular case, in which we calculated the costs and benefits (and hence the ROI) of adopting an RA in the IT department of a public organization. This adoption was driven by Everis so that we applied the implementation of the model described in Section III.B.

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 3 parts. First, a post-mortem analysis in which our challenge was to extract metrics (see Table I) from already collected data. We calculated:

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

The above costs were accurately computed because this organization keeps track of project activities with their invested time. To identify the benefits, we interviewed application developers to estimate the costs of developing applications without using the RA. When there was no data, we asked the involved stakeholders as a light-weight solution [23].

Second, it was necessary to estimate the rest of factors:

- AMC, 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 AMC.
- AEC, 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.

Third, 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, after some additional interviews, we have 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 AMC is computed in the first year.
- A module is evolved (with new functionality) or added to the RA every year starting the second year.

Under these assumptions, the costs and benefits for the future can be calculated as shown in Table II. Table III summarizes the total costs and benefits for these 3 years. As we can see in Table III, this organization will realize a ROI within 2 years through gains in systematic reuse and application maintainability. The ROI after 3 years is 78%.

TABLE II. DESIGN OF THE BUSINESS CASE

	Year 1	Year 2	Year 3
Total benefi	4*(DCA+SCA)	8*(DCA+SCA)	8*(DCA+SCA)
Total cost	ADC+AMC+4*UDC	8*UDC+AEC	8*UDC+AEC

TABLE III. RESULTS OF THE BUSINESS CASE

	Year 1	Year 2	Year 3	Total
Total benefit	148,190 €	296,379 €	296,379 €	740,948 €
Total cost	266,083 €	74,729 €	74,729 €	415,540 €
Net cash flow	-117,894 €	221,651 €	221,651 €	325,408 €
Cumulative cash flow .	-117,894 €	103,757 €	325,408 €	

Data collection in this exemplar case was supported by the adoption of good practices with tool support: activity tracking with JIRA[1] (keeping track of activities and their invested time); test-driven development with Sonar[2] that informed about basic software metrics and percentages of tests compliance; and continuous integration with Jenkins[3]. The adoption of these practices and tools to collect data is the basis for moving software development from its usual low-validity environment, to a high-validity environment. As stated by Erdogmus and Favaro [9], high-validity environments allow acquiring reliable expert intuition and performing an accurate cost-benefit analysis, in this case of RAs.

V. CONCLUSIONS & NEXT STEPS

Architecture improvements are extremely difficult to evaluate in an analytic and quantitative way, contrary to the business efficacy (sales, marketing, and manufacturing) [6]. 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 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 [8].

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 point is the integration of different metrics that complement each other evaluating several RA-relevant aspects. On the other hand, a potential weakness of this approach is that it does not look at risks.

As future work, we plan to enrich the economic model with more metrics (like technical debt) to analyze more quality attributes, and coping with metrics risk. Also, we want to use homogeneity metrics to discover the RA modules that bring more benefit to an organization [7].

[1] http://www.atlassian.com/es/software/jira/overview
[2] http://www.sonarsource.org/
[3] http://jenkins-ci.org/

REFERENCES

[1] M. Ali, M. Ali Babar, K. Schmid, "A Comparative Survey of Economic Models for Software Product Lines," SEAA 2009.

[2] S. Angelov, J.J.M. Trienekens, P. Grefen, "Towards a Method for the Evaluation of Reference Architectures: Experiences from a Case," ECSA, 2008.

[3] S. Angelov, P. Grefen, D. Greefhorst, "A framework for analysis and design of software reference architectures," Information and Software Technology, vol. 54, 2012.

[4] C.Y. Baldwin, K.B. Clark, "Design Rules: Volume 1, The Power of Modularity," Cambridge, MA: MIT Press, 2000.

[5] B. Boehm, "Value-based software engineering: Seven key elements and ethical considerations," Value-Based Software Engineering, 2006.

[6] J. Carriere, R. Kazman, I. Ozkaya, "A cost-benefit framework for making architectural decisions in a business context," ICSE 2010.

[7] P.C. Clements, J.D. McGregor, and S.G. Cohen, "The Structured Intuitive Model for Product Line Economics (SIMPLE)," Technical Report CMU/SEI, 2005.

[8] H. Erdogmus, J. Favaro, W. Strigel, "Return on Investment," IEEE Software, 2004.

[9] H. Erdogmus, J. Favaro, "The Value Proposition for Agility–A Dual Perspective," 2012. Available at: http://www.infoq.com/presentations/Agility-Value

[10] B.P. Gallagher, "Using the architecture tradeoff analysis method to evaluate a reference architecture: A case study," Technical Report CMU/SEI, 2000.

[11] M. Galster, P. Avgeriou, "Empirically-grounded reference architectures: a proposal," QoSA/ISARCS, 2011.

[12] B. Graaf, H. van Dijk, A. van Deursen, "Evaluating an embedded software reference architecture," CSMR, 2005.

[13] A. Ivanovic, P. America, "Customer Value in Architecture Decision Making," ECSA, 2010.

[14] J. Letouzey, "The SQALE Method for Evaluating Technical Debt," MTD@ICSE, 2012.

[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, http://hbswk.hbs.edu/item/5894.html.

[16] S. Martínez-Fernández, D. Ameller, C. Ayala, X. Franch and X. Terradellas, "Conducting Empirical Studies on Reference Architectures in IT Consulting Firms," UPC, 2012.

[17] M. Moore, R. Kazman, M. Klein, J. Asundi, "Quantifying the Value of Architecture Design Decisions: Lessons from the Field," ICSE, 2003.

[18] E.Y. Nakagawa, P.O. Antonino, M. Becker, "Reference architecture and product line architecture," ECSA, 2011.

[19] E.Y. Nakagawa, "Reference Architectures and Variability: Current Status & Future Perspectives," VARSE/ECSA, 2012.

[20] R.L. Nord, I. Ozkaya P. Kruchten, M. Gonzalez-Rojas, "In Search of a Metric for Managing Architectural Technical Debt", ECSA, 2012.

[21] I. Ozkaya, R. Kazman, M. Klein, "Quality-Attribute Based Economic Valuation of Architectural Patterns," ESC, 2007.

[22] J.S. Poulin, "Measuring Software Reuse," Reading, MA: Addison-Wesley, 1997.

[23] R. van Solingen, "Measuring the ROI of Software Process Improvement," IEEE Software, 2004.

[24] K.J. Sullivan, W. Griswold, Y. Cai, B. Hallen, "The structure and value of modularity in software design," ESEC/FSE 2001.