Economics of architectural investments in industrial practice

Ana Ivanović and Pierre America
Philips Research, Eindhoven, The Netherlands
{ana.ivanovic, pierrre.america}@philips.com

Abstract[1]

Architectural investments are large, lengthy, and risky. A decision to invest in architecture is often postponed because of difficulties to estimate its economic value and to communicate its long-term benefits.

This paper proposes a method to support a decision process of architectural investments on an economic basis in industrial practice. The method combines qualitative Real Options analysis to design an architectural investment decision process and the Net Present Value to quantify the economic value of investing in architecture. The proposed framework can be used by practitioners to make economically sound decisions instead of relaying on gut feeling.

The paper demonstrates the framework by valuating an investment in phasing out legacy software in a medical imaging product line.

1. Introduction

Philips Healthcare develops and markets a wide range of software-intensive product lines. The size of the software is typically several million lines of code, with a similar amount of test code.

Investing in an entirely new architecture in such systems is avoided because this involves an enormous amount of work and risk. However, the organization has to take a decision to improve quality attributes to remain competitive in the market. Software restructuring is becoming an established practice to improve the quality of the software while maintaining the external behavior of the code and requirements stable. We were asked to provide an approach to support a decision process on investments in architecture improvements tailored to architects' practice.

Extensive research has been done in estimating economic value of software investments [8]. Scenario-based methods [5, 9] and Real Options [1, 4, 6, 7] approaches are commonly used for assessing the value of architecture. With some exceptions, the current approaches apply complex mathematical formalisms or require numerous stakeholders for architectural valuation that make them rarely applied in an industrial setting. We propose an approach, based on the Real Options way of thinking and Net Present Value, to explain and estimate economic benefits of investments in architecture tailored to architects' practice. In the following sections we describe our method, apply it on a case study, and propose further method improvements.

2. Method

Real Options and Net Present Value. Real Options is an established economic approach for valuation investments under uncertainties [3]. We apply the Real Options way of thinking defining an architectural investment as investment that gives a right, but not an obligation to implement new features with less development effort and shorter time-to-market. Architectural investment involves at least two investment decisions:

1. *The decision to invest in architecture (buying the option).* This involves architecture implementation, i.e. writing or modifying software following the guidelines of the architecture.
2. *The decision to invest in deploying the architecture (exercising the option).* This involves developing new features, installing the software on systems, offering new services, or selling systems with this software.

Figure 1 shows a simplified decision tree with two decision points: Invest in architecture and Implement feature. In practice regardless of our investment in architecture there may or may not be request to implement a feature. The feature should be implemented

[1] This work has been carried out as a part of the Darwin project at Philips Healthcare under the responsibility of the Embedded Systems Institute. This project is partially supported by the Dutch Ministry of Economic Affairs under the BSIK program.

when its *Market Value - Dev Cost* > *0*. In the case that we did not invest in the architecture *Dev Cost old* would be higher, *Time to market* longer (assuming fixed organization resources) resulting in a lower revenue. Net Present Value determines when the architectural investment will pay off. The architecture will pay off when the present value of the cash flow facilitated by the new architecture is greater than the cash flow facilitated keeping the existing (old) architecture. We identify four parameters to estimate Net Present Value: cost, time, market value, and uncertainty.

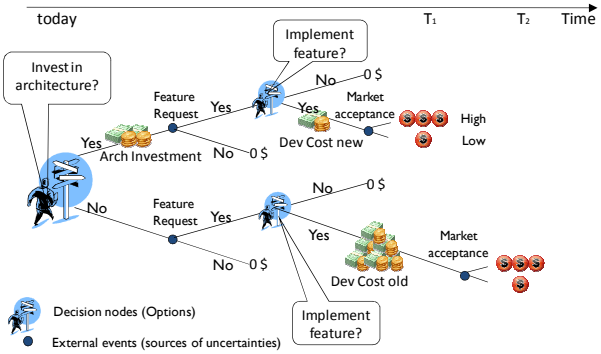

Figure 1. A decision tree of architectural investment

Parameters.

- **Cost.** Investing in architecture will cost money, *Arch Invest*. The cost of developing an individual feature is called *Development cost*, whereas *Maintenance cost* refers to the overall maintenance of the system. These will be different for implementing the feature in scenarios with the existing and the new architecture. Table 1 shows the cost savings of investment in architecture.

Table 1. Cost savings

	New Architecture	Existing Architecture	Cost Savings
Maintenance Cost	*Maint Cost new*	*Maint Cost old*	***Δ Maint Cost***
Development cost	*Dev Cost new*	*Dev Cost old*	***Δ Dev Cost***

- **Time.** *Implementation time* will define the moment to start architecture and *Deployment time* defines how long we may take the benefits of the architecture. *Time to market* is the time until the architecture is deployed to generate new cash flow.
- **Market value.** The market value is the difference in the market value of the feature deployed on the existing and the new architecture, *Δ Market Value*.
- **Uncertainty.** The probability of the feature request and market acceptance must be estimated.

The number of features deployed on the new architecture will significantly influence the value of the investments. Offering new functionality with shorter time to market provides market benefits by enabling earlier cash flow. We apply this method to the case study below.

3. Case study[2]

Disruptive innovation in professional software healthcare is often provided in parallel with the existing solutions to reduce risk of development and ascertain market acceptance of the new solution. Over time the older software is used less frequently and becomes legacy. The medical imaging product line in our case study includes several products with two software releases per year supported by several million lines of code.

The legacy Mini software exists in parallel with the Maxi software and they are tightly coupled. The user runs applications either in one or the other software environment, switching manually between these two working environments. If any new feature is requested, it has to be implemented and tested in both software environments. The architects claim that due to legacy Mini software, there is high maintenance cost, double test effort, and low extensibility.

Therefore, it was decided to replace the functionality available in the legacy Mini environment by new functionality in the Maxi environment keeping all functionality of the system during and after the phase-out project. The phase-out project will last for four years.

The decision to phase out Mini software had already been taken and the phase-out project is still in progress. We were asked to valuate this software restructuring investment decision retrospectively.

3.1 Method: Parameters

We apply the method described in the previous section. To estimate parameters needed for Net Present Value, we conducted several interviews with the stakeholders whose work involved development with the Mini software environment and used their input for analysis.

Cost. The up-front software restructuring investment for Mini software phase-out had been already estimated, *Arch Investment* = 24 *man-years*, by the software architects using the COCOMO II model [2]. When the legacy code is removed, the costs of

[2] We have renamed the relevant projects because their real names are not relevant for the paper

maintenance and testing of legacy code will be reduced to zero, *Maint new* = 0. The whole maintenance cost savings are equal to the estimated costs of Mini software maintenance over time if it had not been phased out.

To estimate *Development cost* in implementing new features, we will need to identify what these features are and how likely they will be requested, as described in the following section.

Time. The implementation time of the Mini phase-out project has been estimated at four years. Based on the roadmaps of the organization the architects estimated that Maxi restructured software will be deployed for at least 5 years after its implementation.

Market value. The stakeholders affected by the software restructuring could not foresee any new features, applications, or businesses facilitated exclusively by the restructured software in the future. Such benefits have high uncertainty and may be realized once the restructured code is in use. The benefits they pointed out did not have a significant market value that could drastically influence our evaluation. Therefore, we simplified our model, neglecting the *Market Value* of investing in restructuring. Without new features envisioned, the development cost savings (*Δ Dev Cost*) are also equal zero.

4. Results

Maintenance cost. We needed to identify the cost of Mini software maintenance over time, if it was not phased out. We started with the architects' claim that maintenance and testing costs are doubled due to keeping both Mini and Maxi software operational. To verify this claim, we traced maintenance effort of Mini and Maxi software in the time-keeping archive two years before the phase-out project started. The archive contains the time spent on assigned tasks written by software developers. We asked an experienced architect to identify relevant tasks for maintaining and testing Maxi and Mini software from the archive. He identified 30 tasks among 10000 tasks per year relevant for our case. The results were surprising. The effort of maintaining the Mini legacy software (0.1-1fte) was very low compared to maintenance effort of Maxi software for the last two years. Thus, the claim of double cost of maintenance of the legacy software would not justify the investment of 24 man years.

Since we believed the software architects' complaints about the large effort associated with legacy software, we investigated further.

Cross-project cost. We interviewed several architects involved in different development projects that have to be integrated with the legacy Mini software. The findings were the following: Due to the presence of the legacy Mini software, the new development projects have to keep their software compatible with the legacy, slowing down development and increasing their development effort. For example, Maxi software used an event mechanism to deal with asynchronous inputs, while Mini used polling. Thus, any new development has to support both mechanisms, resulting in larger development effort and increased software complexity. Usually, this effort of problem solving with the legacy software environment would be administrated as development effort related to the new development project. We concluded that the costs are not dominated by the cost of maintaining legacy software; rather they are dominated by keeping other parts of the software compatible with the legacy over time.

Thus, the main cost savings that we are going to estimate are the extra costs of new development projects in a legacy environment, if Mini software had not been phased out.

We organized a workshop to estimate the cost savings due to Mini software phase-out inviting the architects involved in the projects affected by legacy Mini software. We began the workshop presenting them the framework (including Figure 1) and our findings about Mini software maintenance cost. Next, we asked the architects involved in the current and future projects related to the legacy Mini software to estimate the additional effort in the new development projects if the Mini software had not been phased out.

Collectively, the architects identified the cost savings over the projects as shown in Table 2. We collected cost savings only during the first five years after the Mini software phase out started, since no projects had been planned yet for the years after. However, we see a pattern emerging when we consider that P2 is a successor project for P1 and P5 is a successor for P4. This suggests continuous savings of 9-10 man-years

Table 2. Estimated additional effort per project, if Mini software had not been phased out

Year / Project	0	1	2	3	4	5	6
P1	2	6	6	-			
P2				6	6	..	..
P3	5	-					
P4	3	3	-				
P5			3	3	-	..	..
P6			3	-			

each year. Over a four year period after restructuring this would add-up to 36-40 man years.

In this case the exact calculation of the Net Present Value was not of interest as the *Architectural Investment* was split over 4 years and cash flow was generated over 5 years. The difference in present value of cash flow was negligible.

Consolidating an estimated effort with the software restructuring investment $I_A = 24$ man-years the decision of investment in software restructuring was justified.

4.1 Lessons Learned

Lesson 1. The maintenance cost of keeping legacy software itself is not so high, because the legacy code is very stable. The main cost due to legacy code is distributed over the other development projects to ensure compatibility with the legacy software. We hypothesize that this phenomenon is not limited to our case study.

Lesson 2. The pay-off of phasing out legacy may extend to a point in time not yet planned by product roadmap and this should be discussed additionally.

Lesson 3. Gathering data to construct economic parameters for determining the value of architecture investment is difficult. Although identifying cost savings over the projects was intuitive for the architects they were also pressed hard to think beyond the planned projects.

Lesson 4. The pay-off of a phase-out investment may already start before the end of the project. This is because new developments can often afford to be incompatible with the phased-out software, since they will be released after the phase-out is completed.

5. Conclusion

We have described a framework to support a decision process for architectural investment on an economic basis in industrial practice. Real Options and Net Present Value approaches were adapted to suit the needs of the situation at hand.

We have evaluated the framework by applying it to a case study in an industrial context. We generated reasonably accurate results justifying the architectural investment decision to conduct software restructuring.

This paper presents a first step of defining a sound decision-support framework on how to take architectural decisions in industrial practice under uncertainty. Currently we are applying the approach to other projects to evaluate architectural investments including market value. Future work will also focus on investigating how uncertainty influences architectural investment decisions.

Acknowledgements

We would like to thank the people at Philips Healthcare, as well as our colleagues Zharko Aleksovski and Aleksandra Tesanović for their comments on earlier versions of this paper.

References

[1] Rami Bahsoon and Wolfgang Emmerich: Applying ArchOptions to Value the Payoff of Refactoring. In *Sixth International Workshop on Economics-Driven Software Engineering Research*, Edinburgh, Scotland, UK, 2004.

[2] Barry W. Boehm, Ellis Horowitz, Ray Madachy, Donald Reifer, Bradford K. Clark, Bert Steece, A. Winsor Brown, Sunita Chulani, and Chris Abts: *Software Cost Estimation with Cocomo II*. Prentice Hall, 2000.

[3] Tom Copeland and Vladimir Antikarov: *Real Options, A Practitioner's Guide*. TEXERE, New York, 2003.

[4] Hakan Erdogmus: Valuation of Software Initiatives Under Uncertainty: Concepts, Issues, and Techniques. In Stefan Biffl, Aurum Aybuke, Barry W. Boehm, H. Erdogmus, and Paul Gruenbacher, eds.: *Value-Based Software Engineering* Springer, 2006.

[5] Rick Kazman, Jai Asundi, and Mark Klein: Quantifying the costs and benefits of architectural decisions. In *23rd International Conference on Software Engineering*, Toronto, Canada, 2001.

[6] Ipek Ozkaya, Rick Kazman, and Mark Klein: Quality-Attribute-Based Economic Valuation of Architectural Patterns. Software Architecture Technology Initiative, Software Engineering Institute, Technical Report ESC-TR-2007-003, May, 2007.

[7] Kevin Sullivan, Prasad Chalasani, Somesh Jha, and Vibha Sazawal: Software Design as an Investment Activity: A Real Options Perspective. In Lenos Trigeorgis, ed.: *Real Options and Business Strategy: Applications to Decision Making*. 1998.

[8] *Value-Based Software Engineering* Springer, 2006.

[9] J. H. Wesselius: Modeling Architectural Value: Cash Flow, Time and Uncertainty. In *9th International Software Product Lines Conference*, Rennes, France, 2005.