Economics of Investments in Evolvable Architecture in Industrial Practice

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

Abstract[*]

Evolvable architecture provides flexibility to meet unanticipated requirements, shorten time to market and reduce development effort. Investments in evolvable architecture are large, lengthy and risky. Therefore, a decision to invest in evolvability is often postponed until maintenance cost significantly overtakes the development costs, and severely disrupts further developments. The economic benefits of investment in evolvability traditionally focus on cost reduction and underestimate its value of flexibility to facilitate earlier cash flow and enable development of unpredictable requirement changes that otherwise will mean lost market opportunities.

This paper proposes a framework for determining the economic value of investments in evolvable architecture in industrial practice that are simultaneously subject to time-to-market, development effort, and market risk. The assessment is based on real options analysis, a financial valuation technique for valuing real assets under uncertainty. The paper demonstrates the framework on valuating an investment in software phasing out. Using the framework, the organization can assess similar investments in evolvability in the future.

1. Introduction

Highly complex systems such as MRI scanners, satellites, cars, and airplanes implement a large part of their functionality in software. Over time these software intensive systems are continuously changed either by adding new functionalities (development) or modifying a software system, which has already been in use (maintenance). Over time software evolves, development stuff turnover, and the organization changes management practices and development processes that increase *private risk* of development. Private risk may be largely due to project-specific technological uncertainties that will be a focus in this paper. Private risk gives a rise to uncertainty in estimating time-to-market, likelihood of failure, development cost, and market success.

At the same time market demands are shifting dynamically. For example, the market may turn out to be immature for the product, competition may have taken over with their product release strategies, and the value of implemented requirements is unpredictable. These examples constitute *market risk* that will de determined by factors external to the organization. Time-to-market plays a significant role in determining a market vale and competitive advantage that will be explored in the following chapters.

The effective management of investment in software to anticipate market and private risk become critical to an organization's success. The organization distributes limited budget between investments in development and maintenance to maximize the economic value of investment. Frequently, vast resources are allocated for the software development to facilitate more cash flow, while investments in improving system evolvability and reducing private risks are often postponed as their direct economic value is hard to assess. In fact, only 5% of the total maintenance effort [17] is spent in activities, such as restructuring and phasing out legacy software, aimed at reducing private risk [22]. This could be explained by traditional assessment of investment in software evolvability, where the major benefits are cost savings or freeing resources to do further development or maintenance.

The traditional approach underestimates the value of system flexibility to develop unanticipated requirements in the future and be quicker to market

[*] 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.

that significantly influence market value. This poses a challenge: How to assess the economic value of investments in system evolvability that is simultaneously subject of time-to-market, development cost, and market risk? To asses the benefits of investments in system evolvability, it is important to do the assessment as early as possible in the software life cycle processes, because the costs of changes in pre-design and architecture will cascade down through all the intermediate layers of abstraction to the physical system [16].

Related work. Software architecture [4] is an established practice to guide the system design decisions towards fulfilling requirements on various quality attributes. In our Darwin project [26] we focus on evolvability, which we define as the ability of a system architecture to adapt to changing requirements with predictable, minimal effort and time.

Architecting as an economic activity in the context of uncertainty has been investigated by several research communities [2, 3, 13, 19-21, 23, 27]. Real Options theory, based on financial options theory, is becoming an established approach to valuate real-life investments under uncertainty [1, 9, 24]. An option is the right but not the obligation to take an action in the future. This allows deferring certain decisions to a future moment in time when more information is available.

Previous work in applying Real Options analysis to valuing system design under uncertainty is extensive. For example, Baldwin [3] addresses the option value of the modular design as an appealing architectural design strategy given the inherent uncertainty of future change; Bahsoon and Emmerich [2] estimate the option value of architectural flexibility to react to uncertain requirements changes; Ozkaya et al. [19, 20] address how the option value of architectural patterns supports selecting a pattern with respect to quality attributes to bring the most value under uncertain future market conditions. Eengel and Browning estimate the value of architecture adaptability to plan systems updates [11]. Wesselius [27] takes a different tack. Although he does not explicitly use the Real Options theory, his approach addresses the value of the architectural investment taking into account uncertainties, time effect, and cash flow generation in different architectural strategies.

These economic models acknowledge the strategic value of architecture to take architectural decisions in the future when more information is available. They consider cost, value, and alignment with business goals in making decisions for exploiting the possible option values of the architecture. The mathematical formulation of these models is often too complex and applicable only on theoretical model problems. Generally, these models neglect the implications of complexity in industrial practice. Ozkaya et al. [20] introduced some of the challenges to improve these models for practitioners as follows:

- *Identify uncertainty variables*
  Give guidance regarding how and when they can recognize uncertainty variables in architectural decisions and plan for them
- *Define data collection*
  Understand the types of data to be collected in the architecting process, which can assist in filling the gaps between existing theoretical valuation models and practical use.

In this paper we address exactly these challenges, aligned with the Real Options approach. The framework guides the architects (i) to learn a Real Options way of thinking in valuing architectural investment, (ii) to recognize architectural decisions with high economic impact and uncertainties, and (iii) to identify relevant economic parameters needed for constructing the value of the architectural investment.

Paper outline. The remainder of this paper is organized as follows. The Real Options way of thinking is presented in Section 2. The next section proposes how to identify architectural decisions prone to market and private risk with a high economic impact. Section 4 addresses key economic parameters needed for architectural investment valuations in practice. The evaluation of the framework using an example from industrial practice is presented in Section 5. The paper is wrapped up with future challenges and conclusion in Section 6.

2. A Real Options approach for valuating architectural investment

This chapter demonstrates applicability of the Real Options approach to valuate investment in evolvable architecture and how the Real Options analysis captures the extra value of architectural investments due to optionality.

2.1 The Real Options way of thinking

The value of an investment can be estimated in several ways. A simple approach is to add the expected cash flow (i.e., the difference between income and expenditures) for a number of years (as long as the investment is relevant). A refinement of this, called Net Present Value analysis [10, Chapter 2], takes into account that cash flow in the future is less valuable than now by multiplying future cash flow with a time-dependent factor less than one. These traditional valuation techniques are adequate for treating investment decisions with static project structures.

Net Present Value analysis underestimates the value of investments with high uncertainty because it commits to a go or no go decision today and neglects any possibilities to respond to new information in the future, see Figure 1 [8]. By contrast, the Real Options [1, 9, 24] approach calculates the value of management flexibility to take a decision in the future when more information is available. Therefore, it supports a go decision in projects with NPV close to zero, which would be abandoned with only NPV valuation.

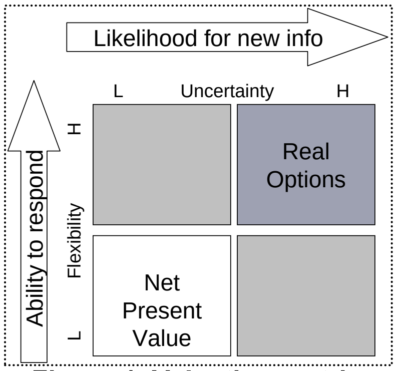

Figure 1. Valuation matrix

Architectural investments are hard to value because their benefits are elusive, spread across the company, and they often depend on the unanticipated follow-on investments in deploying the architecture that are difficult to assess. We will demonstrate a few principles of the Real Options way of thinking, introduced by Amram and Kulatilaka [1], on an architecture example as follows.

An option involves a set of decisions. An option is the right, but not the obligation to take an action in the future and it involves at least two decisions:

- The decision to *buy the option*
- The decision to make use of the right to take the specified action (this is called *exercising the option*)

We elaborate this principle with respect to the architectural investment. The concept of the architecture as a set of decisions has already been established in research and industry [15, 18, 25]. We make an additional view in analyzing architectural investment decisions in economic context. We define the architecture as *a set of time dependent architectural decisions* that have been or are being implemented, together with a number of decision points in the future where the corresponding decisions will be taken to maximize the cash flow.

We postulate that investment in architecture provides the organization the right, but not the obligation, to deploy this architecture in the future. It is very important to emphasize that the most valuable asset is not the description of the architecture but the result of the whole development process, from requirements analysis to architecture implementation and deployment. Thus, when we talk about architectural investments, following the Real Options way of thinking, we distinguish two types of decisions:

- The decision to implement the architecture, i.e., writing or modifying software following the guidelines of the architecture (*buying the option)*
- The decision to deploy the architecture, which may involve, e.g., developing new features, installing the software on systems, and selling systems with this software (*exercising the option*)

The investment in architecture consists of two consecutive stages: investment in an architecture implementation and an optional, follow-on investment in architectural deployment. The evolvable architecture will lower private risk in new development with a shorter time-to-market and reduced development cots, therefore, enabling earlier cash flow. Additionally, the evolvable architecture will decrease market risk by enabling development of unanticipated requirements only if they are requested and the market outlook is sufficiently positive, thus giving the deployment stages the flavor of an option. The Real Options analysis captures this extra value of design flexibility due to optionality.

The value of a Real Option is determined like the value of an option in the financial market. The fact that the option can be exercised at a later point in time, when there is less uncertainty about the value of the underlying asset, contributes to its value.

We elaborate this principle with respect to the investment in evolvability. The flexibility of evolvable architecture gives a right to postpone a decision to implement new requirements when they are requested from the market. The requirements will be implemented with shorter time-to-market capturing market opportunities at time. The more uncertain

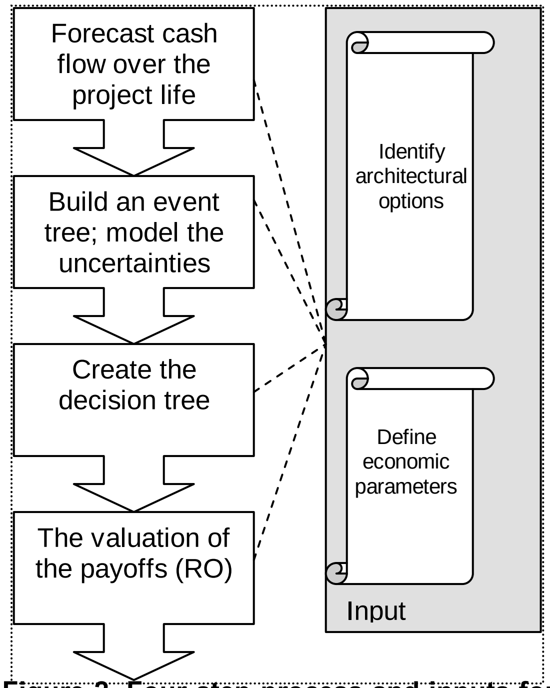
Forecast cash flow over the project life
Build an event tree; model the uncertainties
Create the decision tree
The valuation of the payoffs (RO)
Identify architectural options
Define economic parameters
Input

Figure 2. Four step process and inputs for Real Options architecture investment valuation

requirement requests and their future value, the higher value of investment in evolvability.

Real Options thinking can be used to design and manage strategic investments proactively. The Real Options thinking is all about taking decisions at the right moment.

Decisions that depend on uncertain circumstances, e.g., market demand and technology maturity, should be taken as late as possible. For example, delaying a choice of a requirements portfolio when the market demand is less uncertain, allows better assessment of their economic feasibility; the cost of their development is lower then their generated revenue. This managerial flexibility will maximize the value of the large up-front investment in the architecture implementation.

Once the real options way of thinking is established, there is a need to identify the Real Options attributes to determine the economic value of the architecture.

2.2 Real Options attributes

Real Options analysis is done in four steps as shown in Figure 2 [9]. We will follow this process to determine the value of the architectural investment. The value of the architecture mapped to the Real Options model is determined by: (1) cash flow facilitated by the architectural investment over the architecture life cycle, (2) fluctuation of the cash flow with respect to market value, development effort, and time-to-market, (3) the investment needed to deploy the architecture (exercise costs), (4) time until the architecture is deployed, (5) interest rate relative to the development effort and time-to-market (project-specific).

In Section 4 we elaborate these attributes further decomposing them into time, cost, and benefit parameters with respect to the software system development process. In the following section we point out some relevant architectural decisions that have a high impact on the value of the architectural investment.

3. Options in architectural investments

The architecture design process results in the validated architecture considering the architectural concerns and the context of the system [14]. To apply an economic framework in the architecture design process, we need to identify the value drivers and architectural options that contribute to the value of the architectural investment.

3.1 Value drivers of architectural investments

Erdogmus [12] identifies value drivers for valuing software project development using the Real Options approach: We adapt Erdogmus' classification for valuing architecture investments with a minor modification to keep a consistency in the paper. The value drivers for architectural investments are: *User-observable quality attributes*, *Non user-observable quality attributes*, *Development effort*, *Time-to-market*, *Revenue*, and *Unanticipated requirements*.

We visualize the effect of uncertainty to value drivers and how they affect the value of architectural investment in Figure 3. Value drivers with market risk are presented in gray ellipses, while drivers with private risk are shown in white ellipses. Also the ellipse form shows value volatility of each driver and we will elaborate them further.

User-observable quality attributes. The value of improvements in *User-observable quality attributes*, such as performance and usability, depends on the importance of the improvements to the user. The value of this attribute is prominent to the user only if it is absent. Therefore, these attributes are known, 'must have' attributes and their value will depend on the private risk to keep desired quality.

Non user-observable quality attributes. The value of improvements in *Non user-observable quality attributes* of the product, such as evolvability and reliability, depends on the follow-up product improvements in functionality and quality. The private risk of these drivers is high due to their long-term uncertain benefits. According to the second Real Options principle (see Section 2.1), the higher risk will contribute to the higher value of investing in architecture that enables these attributes.

Development effort and Time-to-market. *Development effort* and *Time-to-market* are dependent drivers in software projects development with similar private risk, see Figure 3. The high magnitude value of *Time-to-marke*t in this tier demonstrates a prominent effect to create the value being earlier to market. First, this will generate earlier cash flow important for increasing the value of investment and second, it may make a competitive advantage to the organization and influence its market share.

Revenue. We define *Revenue* as the income of developing anticipated requirements with the new architecture. The value of architectural investment depends significantly on *Revenue*. The volatility of *Revenue* is underlined in unpredictable user acceptance of the software product and a competition release strategy.

Unanticipated requirements. We define *Unanticipated requirements* as unpredictable requirements that were not foreseen or they could not be met with the old architecture due to their high development cost or non technical feasibility. This value driver has the highest market uncertainty, therefore, contributes most to the value of the investment in architecture evolvability, see Section 2.2.

3.2 Architectural options

The value of architecture investment depends on a particular architecture scenario and effective management of value drivers under uncertainties. We classify several architectural scenarios in different option types to demonstrate the architectural optionality nature in Table 1.

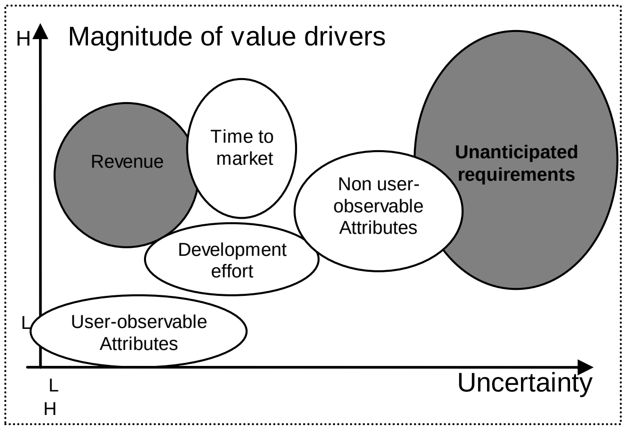
H
Magnitude of value drivers
Revenue
Time to market
Unanticipated requirements
Non user-observable Attributes
Development effort
L
User-observable Attributes
L
H
Uncertainty

Figure 3. Value creation in architectural investments

Once architectural options, the value drivers, and their uncertainties are identified, there is a need to identify how to gather parameters to valuate architectural investment under uncertainties.

4. Economic parameters for the architectural investment valuation

Clements [6] proposes an economic model of software architecture decisions based on the Structured Intuitive Model for Product Line Economics (SIMPLE) [7]. An adapted version of Clements's model could be used in our framework, but we take a different tack. We see the architecture as an asset to facilitate cash flow in the organization. The cash flow depends on the value drivers and how they are managed under uncertainties.

Here, we propose how to break down the Real Options attributes (see Section 2.2) into the set of time, cost, and benefit parameters suitable for data collection in industrial practice.

4.1 Time parameters

We define two time parameters for valuing the architectural investment, *Implementation time* and *Deployment time*. Implementation time is a period of implementing the new architecture from requirements to its development. Deployment time is a period from the moment the architecture is implemented until it becomes irrelevant by implementing a newer architecture. *Implementation time* and *Deployment time* cover a whole architecture lifecycle.

4.2 Cost parameters

We classify costs according to a moment when the cost arise in the architecture lifecycle. *Architectural investment* is cost of implementing new architecture. Cost of deploying an architecture to implement a new requirement is called *Incremental cost* (exercise cost).

Incremental cost will be present only if we decide to implement the requirements in the future. We assume that the sum of all incremental cost represent the development effort of the organization. It is important to notice that the cost of implementing the requirement with the new architecture (*Incr cost new*) and the cost of implementing the same requirement with the existing architecture (*Incr cost old*) will be different. This difference depends on the private risk and influence the value driver, *Development effort*.

Managing maintenance cost within the limited organization budget is an important factor to lower private risks as we mentioned in Chapter 1. One aim of investment in evolvable architecture is to keep the maintenance costs low and enable more resources for development. Therefore, we include maintenance costs as a separated cost parameter. Analogous to the *Incremental costs*, we distinguish the maintenance costs with the new architecture and with the existing one, *Maint old* and *Maint new.*

Table 1. Options in architectural scenarios

Options	Architectural scenario
Growth option	Architecture sets the path for future follow-on investments and strategic growth. Examples: - Architecture provides flexibility in telecommunication industry to serve larger number of customers if there is a demand in the future. - Architecture improves *reliability* in medical imaging systems and enables an entrance to the new market, e.g. image-guided treatments.
Flexibility option	Architecture facilitates development of requirements with reduced *Development effort* and a shorter *Time-to-market,* and enables development of *Unanticipated requirements* [2, 27]. Examples of investments in evolvable architecture are already explained in Chapter 2.
Timing option	Architectural implementation is optimized to be in time for market demands to gain maximum pay-offs. Example: Release the prioritized set of requirements on the new architecture earlier even if the quality attributes are not completely fulfilled to gain market share and be ahead of competition. This option represents the full management of value drivers, *Time-to-market, Revenue*, and *Quality attributes* to maximize the value of the architectural investment.
Exit option	Architecture implementation or deployment is abandoned if the architecture is not feasible or market demands are low[12].

4.3 Benefit parameters

The architectural investment provides two main benefits with respect to the cash flow: *cost savings* and *market benefits.*

Market benefits. The main market benefit comes from the *Market value* of offered requirements, features, and services to the market. We distinguish the Market value of anticipated and unanticipated requirements. A sum of anticipated requirement market values contributes to *Revenue* value driver.

The market value of unanticipated requirements will contribute most to the value of investment in evolvable architecture (see Figure 3), due to a fact that unanticipated requirements without the evolvable architecture would not be implemented. Last but not least, offering a new requirement with short time-to-market provides market benefit by enabling earlier cash flow. Therefore, *Market value* is a time and uncertainty dependant function.

Cost savings

We define two cost savings benefits in Table 2. First, a difference in implementation cost for the new requirements with the new and existing architecture. Second, the difference in maintenance cost with the new and existing architecture.

Consolidating time, cost, and benefit parameters for valuing the architectural investment, we present value influence diagram in Figure 4.

As we can see, the number of requested requirements and their market value will significantly increase the value of the investments. Additionally, the requirements that will take enormous cost to implement without the new architecture or unanticipated requirements will contribute to the architectural value tremendously.

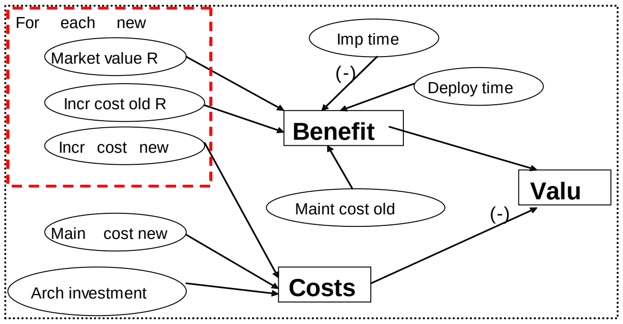

Figure 4. Value Influence Diagram

We summarize our framework for valuating the architectural investment in the four steps Real Options approach as follows:

The time, costs and benefit parameters will determine the cash flow facilitated by the architecture, addressing the first step of the Real Options approach in Figure 1. The cash flow variations will depend on private and market risk which cover step 2. The architectural decisions managing value drivers will be embedded in step 3 and valuation of the architectural investment is done in step 4.

As it can be observed, the framework for valuating the architectural investment follows the four activities in the Real Options approach rather closely.

In the following section we apply the valuation framework to a software restructuring project at Philips Healthcare.[†]

5. Case study

5.1 Introduction: software restructuring

Philips Healthcare develops and markets a wide range of medical imaging systems for creation and storage of medical images to facilitate diagnosis, treatment planning, and intervention to cure various diseases. Examples of such systems are computer

Table 2: Cost savings parameter

	New Architecture	Old Architecture	Cost Savings
Maintenance	*Maint new*	*Maint old*	**ΔMaint cost**
Increment cost	*Incr cost new*	*Incre cost old*	**ΔIncr cost**

tomography systems, radiology review workstations, interventional X-ray systems, and magnetic resonance imaging systems.

Although the hardware in these systems is often impressive, most of the development cost and effort is nowadays invested in software. The size of the software is typically several millions lines of code, with a similar amount of test code. The department that develops such a system generally evolves an existing code base, with a new version released approximately once a year.

At Philips Healthcare, a project usually aims to enhance existing systems by adding new features (development) or improving software quality (maintenance), or both. Investing in an entirely new architecture in such system is avoided because this involves an enormous amount of work and risk. However, the organization has to take a decision to improve quality properties to remain competitive in the market. One way to solve the problem is software restructuring. This is becoming an established practice to improve the quality of the software while keeping the external behavior of the code and requirements stable.

[†] We have renamed the relevant projects because their real names do not matter here.

We valued the investment in a software restructuring project aiming at removing legacy code and redesigning the system without changing its clinical functionality.

5.2 The Mini software phase-out project

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 (see Figure 5).

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 are high maintenance costs, 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, phasing out legacy software. To remove legacy Mini software and restructure current Maxi software, a four year Mini phase-out project was created with a budget of 24 man-years. The main requirement of the project was to preserve all functionality of the system during and after the phase-out project. The investment in software restructuring should support reducing costs and facilitating quicker development of new projects.

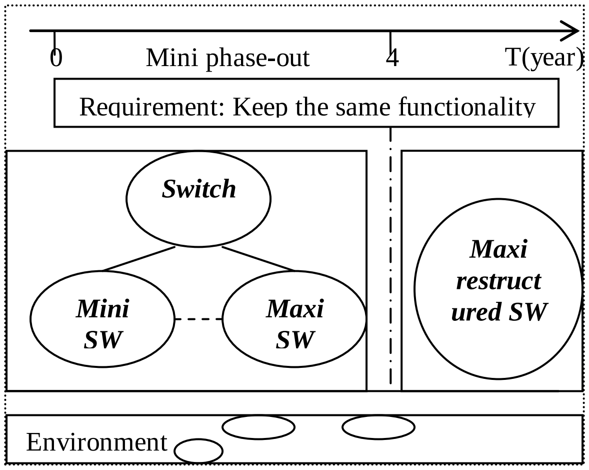

Figure 5. Mini phase-out project

The decision to phase out Mini software had already been taken. We were asked to valuate retroactively this software restructuring investment and provide a method that can be used in practice for valuing similar projects in the future.

5.3 Options in architectural investment

Investing in the Mini software phase-out and restructuring project has two types of benefits. First, this project should enable shorter *Time to market* and reduce *Development effort* in implementing new requirements in restructured Maxi software (Flexibility option). Second, the Maxi restructured software may facilitate development of new projects that would otherwise be too costly to implement in both software environments (Growth option).

5.4 Economic parameters for software restructuring valuation

We conducted interviews with the stakeholders that are affected by the software restructuring to identify key economic parameters for economic valuation of investment in this project.

5.4.1 Time parameters. The implementation time of the Mini phase-out project has been determined at 4 years. To identify the deployment time, we asked one of the architects to estimate how long the Maxi restructured software will be deployed once it is implemented. Taking into account the roadmaps of the organization, he estimated a deployment time of at least 5 years.

5.4.2 Cost parameters. 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 [5].

When the legacy code is removed, the costs of maintenance and testing of legacy code will be reduced to zero, *Maint new =0*.

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

5.4.3 Benefit parameters

Market benefits. The stakeholders that are affected by the software restructuring could not foresee any new features, applications, or business facilitated exclusively by the restructured software, although such benefits might 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 unanticipated requirements for this case.

Cost savings. The costs saving due to absence of the Mini software are two fold. New projects will not have to integrate new requirements in two environments, passing extensive testing and verification procedures. Thus, the extra costs of new development projects in legacy environment, if Mini software has not been phased out, would be saved.

After the Mini software will be phased out, its maintenance costs will be zero. Thus, the whole maintenance cost savings are equal to the estimated costs of Mini software maintenance over time if it had not been phased out.

Finally, the value of the software restructuring will be a difference between maintenance and incremental cost savings and the up-front investment in software restructuring.

5.5 Results

We needed to identify two cost parameters to valuate the Mini software phase-out investment: First, the cost of Mini software maintenance over time, if it were not phased out. Second, the extra costs of the new development projects in legacy environment.

Maintenance cost. We started with the architects' claim: "Maintenance and testing costs are doubled due to keeping both Mini and Maxi software operational".

To prove this claim, we used the time-keeping task archive to identify maintenance effort of the Mini and Maxi software two years before the phase-out project started. The archive tool supports people to write time for the assigned software development task. An experienced architect looked into the document to identify among 10000 tasks only 30 tasks relevant for maintaining and testing Maxi and Mini software. The results were surprising. The effort of maintaining the Mini legacy software was very low compared to maintenance effort of Maxi software for the last two years. Thus, the claim of double costs of maintenance of the legacy (0.1-1fte) seemed not justified the investment of 24 man year, even if this maintenance cost will rise 10 times over next 10 years, see Figure 6.

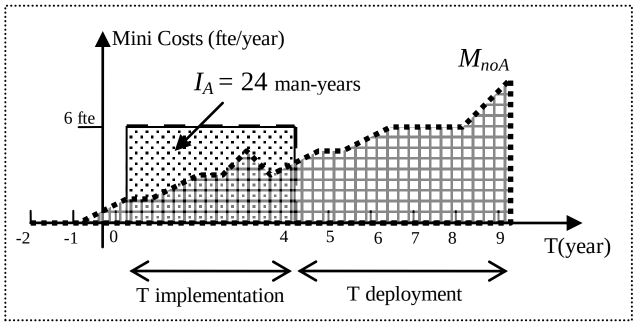

Figure 6. Mini software costs savings estimation

Since we believed the software architects' complaints of huge effort to maintain legacy software, we investigated further.

Increment Cost. We interviewed several architects, running 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 projects have to keep their software compatible with it, slowing down development and increasing their development effort. Usually, this effort of problem solving with the legacy software environment would be administrated as development effort on the new project. We concluded that the costs are not dominated by the legacy maintenance costs over time; 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.

This outcome helped us to organize a workshop for the architects involved in the projects affected by legacy Mini software to estimate the cost savings due to Mini software phase-out. We began the workshop presenting them our framework and our findings about Mini software cost saving, see Figure 6.

Next, we asked the architects involved in the current and future projects, related to the legacy Mini software, to estimate the additional effort of development if the Mini software had not been phased out, see Table 3.

Table 3. 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	-			

Collectively, the architects identified the cost savings over the projects filling the table shown in Table 3. 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 a continuous saving of 9-10 man-years each year. Over a four year period after restructuring this would add-up to 36-40 man years.

Consolidating an estimated effort with the software restructuring investment I_A = 24man year, the decision of investment in software restructuring was justified.

5.6 Lessons Learned

Lesson 1: You cannot just count man-years for development if you want to come up with the optimal architecture implementation strategy. In the development department there are generally fixed resources for enhancing the system's quality each year. Therefore, an architecture implementation strategy and decisions are often constrained by available resources in the organization.

Lesson 2: The main costs due to legacy code are distributed over the other development projects that have to be kept compatible with the legacy software. The maintenance cost of keeping the legacy software itself is not so high. We hypothesize that this phenomenon is not limited to our case study.

Lesson 3: The payback of phasing out legacy may occur at a point of time not yet covered by product roadmap.

Lesson 4: Gathering data to construct economic parameters for determining the value of architecture investment is difficult. Although identifying costs savings over the projects was intuitive for the architects, they were also hard pressed to think beyond the planned projects.

Lesson 5: The Real Options way of thinking for managing decisions was adopted quickly, although option terminology was sometimes found confusing.

6. Challenges and Conclusions

We have described a framework to determine economic value of architectural investments. We used the Real Options way of thinking, learned from other approaches in the literature, and adapted them to suit the needs of the situation at hand.

We have evaluated the framework by applying it in the case study in an industrial context. We generated reasonably accurate results justifying the architectural investment decision to do software restructuring. Although the case study was not very complicated, it still illustrates how to think strategically in terms of architectural options.

This paper presents a first step in modeling the economic value of the architectural investment under uncertainties in industrial practice. In the near future we hope to apply the approach to other case studies, where market value and uncertainties play a larger role. Future work will concentrate on a value of investing in evolvability with respect to a time-to-market and uncertainties.

Acknowledgements

We would like to thank the people at Philips Healthcare who have cooperated with us in the case study, as well as our colleagues Gerrit Muller, Wim Stut, Ferrie Aalders, and Aleksandra Tesnaovic for their comments on earlier versions of this paper.

References

[1] Martha Amram and Nalin Kulatilaka: *Real Options, Managing Strategic Investment in an Uncertain World*. Harvard Business School Press, Boston, 1999.

[2] Rami Bahsoon and Wolfgang Emmerich: Evaluating software architectures: development, stability and evolution. In *ACS/IEEE International Conference on Computer Systems and Applications*, Tunis, Tunesia, July, 2003, IEEE Press.

[3] Carliss Y. Baldwin: The Option Value of Software Product Lines. In *10th International Software Product Lines Conference*, Baltimore, Maryland, 2006.

[4] Len Bass, Paul Clements, and Rick Kazman: *Software Architecture in Practice*. Addison-Wesley, Reading, Massachusetts, 1998.

[5] 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.

[6] Paul Clements: An Economic Model for Software Architecture Decisions. In *First International Workshop on the Economics of Software and Computation*, Minneapolis, Minnesota, USA, May 20, 2007, IEEE.

[7] Paul Clements, John D. McGregor, and Sholom G. Cohen: The Structured Intuitive Model for Product Line Economics (SIMPLE). Product Line Practice Initiative, Software Engineering Institute, Technical Report CMU/SEI-2005-TR-003, 2005.

[8] T. Copeland, T. Koller, and J. Murrin: *Valuation: Measuring and managing the Value of Companies*, 3 edition. John Wiley & Sons, New York, 2000.

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

[10] Alan Dennis, Barbara Haley Wixom, and Roberta M. Roth: *Systems Analysis and Design,* Third edition. Wiley, 2005.

[11] *INCOSE 2006*, Orlando, USA, 2006.

[12] H. Erdogmus: Valuation of Learning Options in Software Development Under Private and Market Risk. *The Engineering Economist,* vol. 47, no. 3, pp. 308-353, 2002.

[14] Christine Hofmeister, Philippe Kruchten, Robert L. Nord, Henk Obbink, Alexander Ran, and Pierre America: A general model of software architecture design derived from five industrial approaches. *The Journal of Systems and Software,* pp. 106-126, 2006.

[15] Anton Jansen and Jan Bosch: Software Architecture as a Set of Architectural Design Decisions. In *5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05)*, 2005, pp. 109-120.

[16] N.G. Leveson: Intent specifications: an approach to building human-centred applicatios. *IEEE Transactions on Software Engineering,* vol. 26, p. 20, 2000.

[17] B.P. Lientz and E.B. Swanson: *Software Maintanance Management: A Study of the Maintanance of Computer Application Software in 487 Data Processing Organizations*. Addison-Wesley, 1980.

[18] Ruth Malan and Dana Bredemeyer: "Chapter 1. Software Architecture: Central Concerns, Key Decisions," in *Software Architecture Action Guide Book*, 2002.

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

[20] Ipek Ozkaya, Rick Kazman, and Mark Klein: Quality-Attribute Based Economic Valuation of Architectural Patterns. In *First International Workshop on the Economics of Software and Computation*, Minneapolis, Minnesota, USA, May 20, 2007, IEEE.

[21] 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*. Risk Books 1998.

[22] A.A. Takang and P.A. Frubb: *Software Maintnance: Concepts and Practice*. Thompson Computer Press London, 1996.

[23] Alfred Taudes: Software Growth Options. *Journal of Management Information systems,* vol. 15, no. 1, pp. 165-189, 1998.

[24] Lenos Trigeorgis: *Real Options, Managerial Flexibility and Strategy in Resource Allocation*. The MIT Press, London, 1996.

[25] Jeff Tyree and Art Akerman: Architecture decisions: demystifying architecture. *IEEE Software,* vol. 22, no. 2, pp. 19-27, March-April 2005.

[26] Piërre van de Laar, Sjir van Loo, Gerrit Muller, Teade Punter, David Watts, Pierre America, and Joland Rutgers: The Darwin Project: Evolvability of Software-Intensive Systems. In *Third International IEEE Workshop on Software Evolvability at IEEE International Conference on Software Maintenance (ICSM)*, Paris, France, October 1, 2007.

[27] J. H. Wesselius: Modeling Architectural Value: Cash Flow, Time and Uncertainty. In *9th International Software Product Lines Conference*, Rennes, France, 2005, pp. 89-95.