Pós-Graduação em Ciência da Computação

“An Integrated Cost Model for Product Line Engineering”

by

Jarley Palmeira Nóbrega

M.Sc. Dissertation

Universidade Federal de Pernambuco

posgraduacao@cin.ufpe.br

www.cin.ufpe.br/~posgraduacao

RECIFE , March 2008

UNIVERSIDADE FEDERAL DE PERNAMBUCO

CENTRO DE INFORMÁTICA

PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

JARLEY PALMEIRA NÓBREGA

"An Integrated Cost Model for Product Line Engineering"

Este trabalho foi apresentado à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

A master dissertation presented to the Federal University of Pernambuco in partial fulfillment of the requirements for the degree of M.Sc. in Computer Science.

ADVISOR: SILVIO ROMERO DE LEMOS MEIRA

CO-ADVISOR: EDUARDO SANTANA DE ALMEIDA

RECIFE, MARCH 2008

Nóbrega, Jarley Palmeira
An integrated cost model for product line engineering / Jarley Palmeira Nóbrega. – Recife : O Autor, 2008.
144 p. : il., fig.,tab.

Dissertação (mestrado) – Universidade Federal de Pernambuco. CIn. Ciência da Computação, 2008.

Inclui bibliografia, glossário e apêndice.

1. Engenharia de software. I. Título.

005.1 CDD (22.ed.) MEI2008-059

This work is dedicated to my loving wife, Carol, my little son, Guilherme, and to my parents and sister, Januir, Céu and Jane.

Acklowledgements

In the text of this dissertation the term *worthwhile* is used many times to express a situation where an investment is indicated. Considering my master studies an investment itself, I would like to say: it was worthwhile for my life!

In the long road to conclude this work I would like to thank a lot of persons and entities that made their best to support me in this journey.

I would like to thank the RiSE group to give me the support and the space to discuss the aspects that were useful to complete my work. In special, the guys whom I have the honor of working together: Bart, Ana Paula, Fred, Vinícius, Alexandre Álvaro, Alexandre Martins, Rodrigo, Ednaldo, Lica and Kellyton (sorry if I forgot someone).

I would like to thank the people of SERPRO for giving me the time necessary to develop my master activities. In special, I am thankful to Mônica Falcão, Reinaldo Soares and João Veloso for recognizing the importance of my work for SERPRO.

I am very grateful to the people of Universidade Salgado de Oliveira due to the incentives for my work, in special Mêuser Valença and Eleonora Oliveira for the support during these last years.

My special thanks to the professors Carlos Ferraz, Patrícia Tedesco and Flávia Barros. Your classes were very important to my background knowledge, and of course, I had a good time studying your lessons.

There are two persons that I have special thanks. The first one, my advisor and guru, Sílvio Meira with his peculiar vision of the world. His life has been an inspiration for my professional career (and I am not outsourced my career in a public job!). I am proud to be his student. The second person is my co-advisor, Eduardo Almeida, for his patience in giving me the directions for my work. Eduardo's work for RiSE group is a real proof that is possible to transform a dream in a real thing.

I would like to thank my parents, Januir and Céu, to teach me the importance of a good education in my life. I promise you both that I will do the same for my son.

This work is dedicated to the most important persons of my life: my beloved wife, Carol, for her love and patience with me during my master studies period. It would be impossible to finish my work without her incentive. For my son, Guilherme, that changed my life with his birth. All the effort that your father spent in this work was worthwhile because of you.

Finally, I would like to thank God for giving me the serenity and health to support the pressure of writing this work. In the end, You never abandoned me!

Jarley Palmeira Nóbrega

Recife (PE)

March 2008

After a while you learn the subtle difference
Between holding a hand and chaining a soul,

And you learn that love doesn't mean leaning
And company doesn't mean security.

And you begin to learn that kisses aren't contracts
And presents aren't promises,

And you begin to accept your defeats
With your head up and your eyes open
With the grace of a woman, not the grief of a child,

And you learn to build all your roads on today
Because tomorrow's ground is too uncertain for plans
And futures have a way of falling down in mid-flight.

After a while you learn...
That even sunshine burns if you get too much.

So you plant your garden and decorate your own soul,
Instead of waiting for someone to bring you flowers.

And you learn that you really can endure...

That you really are strong

And you really do have worth...

And you learn and learn...

With every good-bye you learn.

Jorge Luís Borges in *You Learn*

Abstract

In the software development community, the process of using existing artifacts rather than building them from scratch – generally known as *software reuse* – has been advanced as a way in which the problems associated with cost and schedule overruns can be avoided. Despite the potential rewards from an effective reuse program, it appears that its large-scale adoption is not particularly prevalent. Among the factors that inhibit reuse adoption there are the economic obstacles faced by organizations, which are concerned with the cost related to develop software *for* reuse and *with* reuse. Currently, the decisions concerning large-scale reuse are often related with an economic viewpoint, since the development of software to be reusable can be considered as an *investment*. Moreover, the adoption of a software product line in a reuse context comes up with some inhibitors, such as the application of cost models in a restricted way, the lack of an investment analysis strategy, and the fact that a few cost models have a reuse scenario-based approach. In this context, this work presents an integrated cost model for product line engineering in order to help the decisions concerning reuse investment. The foundations of the model were based on an extensive survey on cost models for software reuse and its extension to the product line approach. The model presents the definition of a set of cost and benefits functions, the description of reuse scenarios for product line engineering, and an investment analysis strategy. In addition, a simulation model based on the Monte Carlo method was proposed for simulating the reuse scenarios. Finally, this work discusses the results of a case study in the context of a real software development environment where the model was applied.

Keywords: Software Reuse, Cost Models, Software Product Line, Investment Analysis, Monte Carlo Simulation, Software Economics.

Resumo

Dentro da comunidade de desenvolvimento de software, o processo de reutilizar artefatos ao invés de construí-los do zero – normalmente conhecido como *reuso de software* – tem se mostrado uma maneira efetiva de evitar os problemas associados ao estouro de orçamentos e cronogramas de projeto. Apesar do imenso potencial, a adoção de reuso em larga escala ainda não prevalece dentro das organizações. Entre os fatores que contribuem para isso, estão os obstáculos econômicos enfrentados pelas empresas, com uma clara preocupação sobre os custos para desenvolver software *para* e *com* reuso. Atualmente, as decisões relacionadas com reuso são tratadas sob um ponto de vista econômico, devido ao fato do desenvolvimento de software reutilizável ser considerado pelas organizações como um *investimento*. Além disso, a adoção de linhas de produto de software dentro desse contexto traz à tona alguns inibidores de reuso, como por exemplo, a aplicação dos modelos de custo para reuso de forma restrita, a falta de uma estratégia para a análise de investimentos, e o fato que poucos modelos de custo possuem uma abordagem baseada na utilização de cenários de reuso. Nesse contexto, esse trabalho apresenta um modelo integrado de custo para engenharia de linhas de produto, com o objetivo de auxiliar as organizações em seus processos de tomada de decisões na avaliação de investimentos em reuso. Os fundamentos para o modelo foram baseados em uma vasta pesquisa sobre modelos de custo para reuso e sua especialização para linhas de produto de software. O modelo apresenta a definição de funções de custo e benefícios, cenários de reuso e uma estratégia de investimento para linhas de produto. Também é apresentado um modelo de simulação baseado na técnica de Monte Carlo. Por último, um estudo de caso discute os resultados de dentro do contexto de um projeto real de desenvolvimento de software, onde o modelo foi aplicado.

Palavras-chave: Reuso de Software, Modelos de Custo, Linha de Produto de Software, Análise de Investimento, Simulação de Monte Carlo, Economia de Software.

Table of Contents

List of Figures

List of Tables

List of Acronyms

Terms	Description
ARVB	Average Return on Book Value
COCOMO	Constructive Cost Model
COPLIMO	Constructive Product Line Investment Model
COTS	Component Off-the-Shelf
InCoME	Integrated Cost Model for Product Line Engineering
IRR	Internal Rate of Return
GQM	Goal-Question-Metric
NATO	North Atlantic Treaty Organization
NPV	Net Present Value
PB	Payback Value
PH	Persons-Hour
PI	Profitability Index
PLE	Product Line Engineering
qCOPLIMO	Quality-based SPL Cost Estimation Model
RCA	Reuse Cost Avoidance
RCR	Relative Cost of Reuse
RCWR	Relative Cost of Writing for Reuse
RiSE	Reuse in Software Engineering group
ROI	Return on Investment
SD	Start Date (of an investment)
SEI	Software Engineering Institute
SIMPLE	Structured and Intuitive Model for Product Line Economics
SLOC	Size of Lines of Code
SoCoEMo-PLE	Software Cost Estimation for Product Line Engineering
SPL	Software Product Line
UML	Unified Modeling Language

1 Introduction

During the last decades the software development community has been studying the adoption of systematic reuse processes as a key factor for significantly improving software quality and productivity. In this context, a growing number of software development organizations are adopting approaches that emphasize proactive reuse, interchangeable components, and planning cycles, in order to construct high-quality products faster and cheaper (McGregor et al., 2002). A set of standard methods, known as *software product lines*, has been developed around these approaches (Clements et al., 2001).

However, successful application of such processes is intrinsically associated with the capability of the organizations in measuring their progress and identifying the most effective reuse strategies. The lessons learned on applying this approach highlight that the software community needs more quantitative data to support software product lines adoption (Northrop, 2002). Currently, the organizations give an "*investment level*" to product line engineering, and a new problem arises with it: the decision makers want clear and accurate numbers in their business cases.

Thus, the instantiation of this problem is the main subject of this dissertation, which will discuss it in detail, beginning with the problem formulation, passing through the state-of-the-art of existing solutions, ending with the current proposal and its validation.

1.1. Motivation

In the software development community, the process of using existing artifacts rather than building them from scratch – generally known as *software reuse* (Krueger, 1992) – has been advanced as a means in which the problems associated with cost and schedule overruns can be avoided. There are several reports describing the benefits of software reuse within a large set of organizations (Margano et al., 1992), (Poulin et al, 1993), (Lim, 1994), (Brownsword et al., 1996), (Lim, 1998), (Wiles, 1999), (Mili et al., 2001), (Poulin, 2006), (Muthig et al., 2006). The benefits potentially achieved by software reuse include reduced development time and cost, improved software quality, increased overall productivity, increased level of knowledge sharing, improved maintainability of applications, easier adoption of standards, among others (Rothenberger et al., 2002).

Despite the potential rewards from an effective reuse program, it appears that its large-scale adoption is not particularly prevalent. Among the factors that inhibit reuse adoption we can highlight the economic obstacles faced by organizations (Sametinger, 1997), which are concerned with the cost related to develop software *for* reuse and *with* reuse (Poulin et al., 1993). In general, the development of software to be reusable in future projects is more expensive than developing it for a single use (Poulin, 1997a).

Currently, the decisions concerning large-scale reuse are often related with an economic viewpoint, since the development of software to be reusable can be considered as an *investment* (Wiles, 1999). In this context, cost models for software reuse can help the organizations to make decisions concerning reuse investment, including whether or not to invest in a reuse program, whether to choose a specific type of reuse over another, and whether not consider reuse and invest in some other type of technique or process. Frequently, these decisions are strongly related with the financial evaluation of a *reuse scenario* – a development plan that describes which features should be developed in a specific period of interest (Schmid, 2003).

In the literature, there are many studies on the field of software reuse economics , and a large number of cost models have been proposed (Bollinger et al., 1990), (Barnes et al., 1991), (Gaffney et al., 1992), (Margano et al., 1992), (Schimsky, 1992), (Poulin et al., 1993), (Malan et al., 1993), (Frakes et al., 1994a), (Kain, 1994), (Lim, 1994), (Boehm et al., 1995), (Favaro, 1996), (Mili, 1996), (Devanbu et al., 1996), (Favaro et a., 1998), (Wiles, 1999), (Mili et al., 2000), (Nazareth et al., 2004).

However, when we are considering the adoption of a product line in a reuse context some issues arises: **(i)** the existing cost models can be applied only in a restricted way, since they do not reflect some fundamental assumptions for that approach (Böckle et al., 2004), i.e., they do not express the cost nature of a product line, which the development perspective is *product-based*, in opposition to the basic cost models that have a *component-based* viewpoint (Schmid, 2003); **(ii)** the existence of a few economic models dealing with product line engineering compared with the number of basic reuse cost models (Poulin, 1997b), (Cohen, 2003), (Peterson, 2004), (Boehm et al., 2004), (Clements et al., 2005). (Lamine et al., 2005); **(iii)** some of the existing cost models for product line engineering can be applied only to estimate costs savings, lacking of an investment analysis strategy (Peterson, 2004), (Clements et al., 2005); **(iv)** a few models have a reuse scenario-based approach in order to evaluate the dynamic situations that can occur in a product line; and **(v)** none of the currently available models includes in its definition a formal simulation model in order to investigate the uncertainty that can exist in input parameters of reuse scenarios (Muthig et al., 2006).

1.2. Problem Statement

According to the issues gathered from the discussion of the previous section, the work described in this dissertation focuses in achieving the following goal:

This work defines an integrated cost model for software product line engineering to perform investment analysis for a set of reuse scenarios in order to help the stakeholders of an organization in their decision-making tasks.

1.3. Overview of the Proposed Solution

In order to achieve the goals stated in the previous section, the Integrated Cost Model for Product Line Engineering (InCoME) is proposed. A summarized view of InCoME is presented in Figure 1.1. Accordingly, the model is based on the following foundations[1]:

[1] In Chapter 4 all these elements are discussed in details

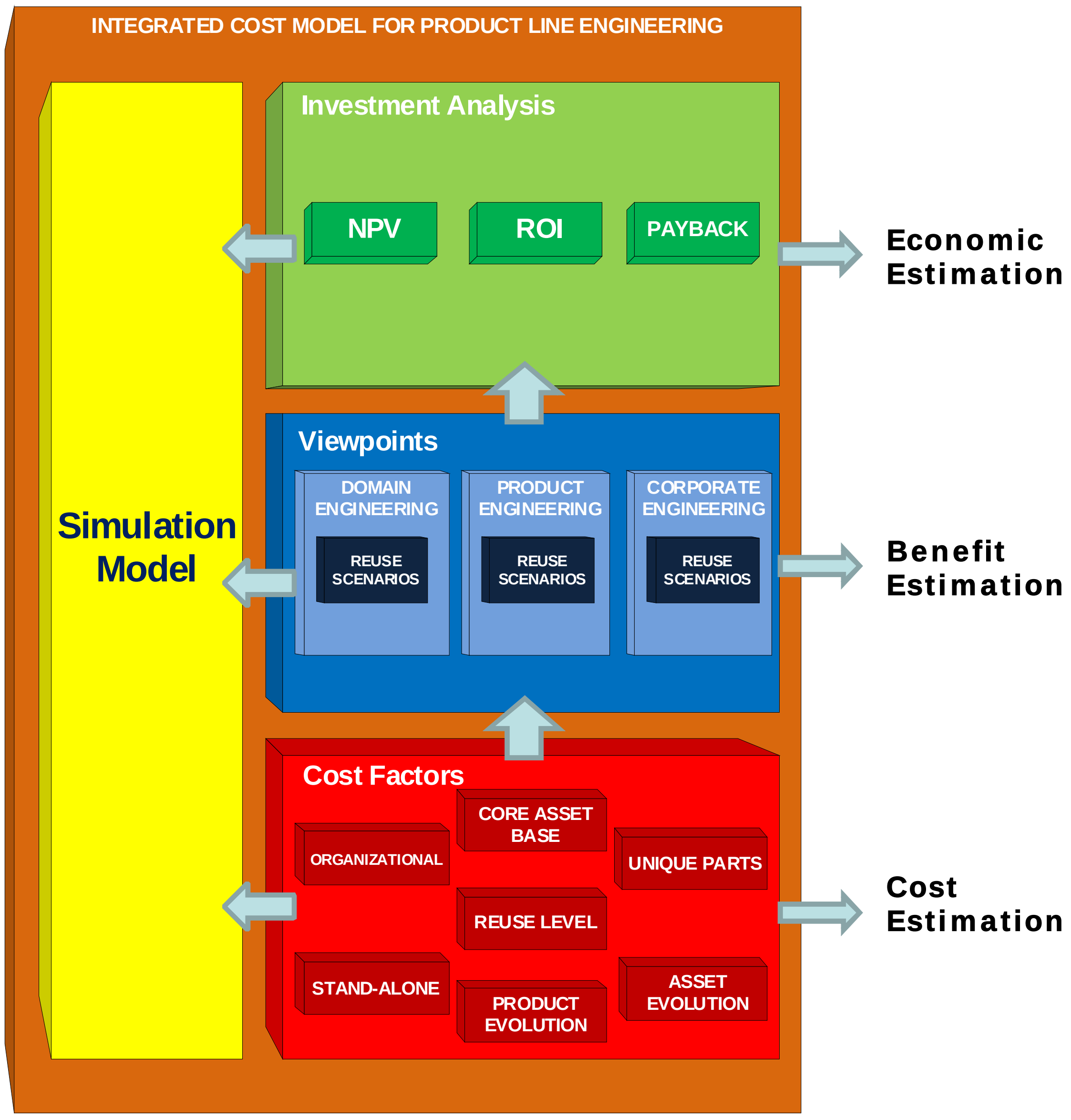


Figure 1.1 – Integrated Cost Model for Product Line Engineering (InCoME)

- **Investment Analysis**. InCoME presents a set of economic functions in order to analyze an investment in a product line. The equations used by the model are known by economics community and its application in a reuse cost model was influenced by the work described by Favaro et al. (Favaro et al., 1998). This work defines functions to calculate the *Net Present Value (NPV)*, *Return on Investment (ROI)*, and other financial estimations for software reuse.
- **Viewpoints**. The model defines three viewpoints in order to provide different visions of an investment in a product line, according to the stakeholders associated with each viewpoint. This strategy was

influenced by the model defined by Mili et al. (Mili et al., 2001) which states that a reuse organization elaborates on four engineering cycles that propagates costs and benefits into each other.

- **Reuse Scenarios**. The model addresses the benefits in adopting a product line by the definition of reuse scenarios. This approach is based on the cost models defined by Schmid (Schmid, 2003), Peterson (Peterson, 2004) and Clements et al. (Clements et al., 2005) which could be useful to model the dynamic situations that can occur in a product line.
- **Cost Factors**. The InCoME cost factors are based on the model defined by Clements et al. (Clements et al., 2005), which includes functions to estimate the most relevant cost drivers for product line engineering. Furthermore, the approach used in the definition of InCoME to derive the cost equations of component engineering cycle was influenced by Mili et al. (Mili et al., 2001) which defines the lowest level of granularity for reuse cost.
- **Simulation Model**. InCoME presents a simulation model for reuse scenarios in order to investigate the sensitivity of the output computed by the model with respect to changes in its input data. This approach was based on the work described by Muthig et al. (Muthig et al., 2006) which uses the technique of Monte Carlo simulation to handle uncertainty for ROI estimations of a product line.

1.3.1. Context

This work is part of a broader reuse initiative promoted by the Reuse in Software Engineering (RiSE)[2] (Almeida et al., 2004). According to (Almeida, 2007): "*RiSE's goal is to develop a robust framework for software reuse in order to enable the adoption of a reuse program. The proposed framework has two layers, as shows in Figure 1.2. The first layer (on the left side) is formed by best practices related to software reuse. Non-technical aspects, such as education, training, incentives, program to introduce reuse, and organizational management are considered. This layer constitutes a*

[2] http://www.rise.com.br/research

fundamental step before the introduction of the framework in organizations. The second layer (on the right side), is formed by important technical aspects related to software reuse, such as processes, environment, and tools."

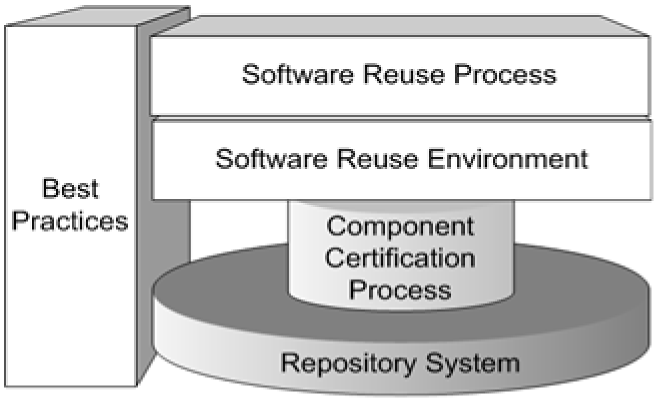


Figure 1.2 – RiSE Framework for Software Reuse

According to Figure 1.2, the RiSE project addresses several reuse aspects not included in the scope of this dissertation, such as software reuse processes (Almeida, 2007), component repository management (Burégio, 2006) and component certification (Alvaro et al., 2006), besides other tools proposed by the project, including domain analysis tools (Lisboa et al., 2007), reverse engineering (Brito, 2007) and component search engines (Garcia et al., 2006), (Mascena, 2006), (Vanderlei et al., 2007).

These efforts are coordinated and will be integrated in a full-fledged enterprise scale reuse solution. The role of InCoME in the RiSE project is to provide a model to evaluate an investment within a product line context, which is included in the **Software Reuse Process layer**.

1.4. Out of Scope

As cited in the previous sections, since InCoME is part of a broader context, a set of related aspects are left out of this work scope. Moreover, we recognized that there is a set of other directions that were discarded in this dissertation due to scope limitations:

- **Reuse Process**. A cost model for software product line is a subject strongly related with the process in which it is inserted (Clements et al., 2005). Since the RiSE group is committed with the definition of a robust reuse process (Almeida, 2007), this dissertation do not take into account the definition of a product line engineering process. There are a set of directions in this sense, including the frameworks defined by SEI (Clements et al., 2004) and Fraunhofer Institute (Bayer et al., 1999).

- **Intangible Benefits**. InCoME focuses on evaluating reuse scenarios performing an investment analysis according to the cost savings computed by the model. In this sense, a set of intangible benefits different of cost savings were not addressed by this work. The work performed by Peterson (Peterson, 2004) has a set of research directions in order to investigate intangible benefits, such as improved time-to-market, market share, among others.
- **Decision Model**. According to Schmid (Schmid, 2003) an investment analysis provided by a reuse cost model can be improved by the management of the risks related with reuse adoption. In this sense, the use of decision-making techniques, such as decision trees, can help organizations in their product lines strategies. A direction for this approach is shown in the work conducted by Schmid (Schmid, 2003), which approaches the application of decision tree analysis in order to mitigate the risks involved on investment decisions.
- **Tool Implementation**. The adoption of product line engineering by software development organizations has led to the emergence of software reuse tools and techniques (Krueger, 2007). It is known that an automated tool can help in using a reuse cost model, as presented by Mili et al. (Mili et al., 2001) in the implementation of archival and analysis modules for their model. Some directions for tool implementation were also given by Clements et al. as a list of features that can be added into the tool. As cited previously, RiSE group has a research branch that focuses on tool implementation and due to scope limitation this subject is not discussed in this dissertation.

1.5. Statement of the Contributions

As a result of the work presented in this dissertation, the following contributions can be enumerated:

1. The extension of a study on the key developments in the field of software reuse cost models, in an attempt to analyze this research area and identify the next trends to follow;
2. A survey based on the state-of-the-art of software product line cost models in order to understand and identify its strengths, weakness and improvement opportunities;
3. The formal definition of an integrated cost model for product line engineering, including its basic cost functions, viewpoints, reuse scenarios, economic functions and a simulation model;
4. The definition, planning, operation, analysis, interpretation and packaging of a case study, which evaluated the accuracy of the proposed model.

In addition, some intermediate results of this work can be found in the literature, as follows:

- (Nóbrega et. al., 2006) Nóbrega, J.; Almeida, E. S.; Meira, S. R. L., "**A Cost Framework Specification for Software Product Lines Scenarios**", in the Sixth Workshop on Component-Based Development (WDBC), Recife, Brazil, 2006.

1.6. Organization of the Dissertation

The remainder of this dissertation is structured as follows.

Chapter 2 surveys the origins of reuse cost models concepts and ideas, features, classifications, and future directions for development in the area.

Chapter 3 surveys the state-of-the-art on software product line cost models, discussing its foundations, features, strengths and weakness, making a comparison between them and highlighting the directions to create a basis for the model defined in this work.

Chapter 4 presents the InCoME, including its objectives, assumptions, foundations, elements and a process to use it within an organization.

Chapter 5 presents the InCoME evaluation, with its context, definition, planning, operation, analysis, interpretation and packaging of the case study, which evaluated the accuracy of the proposed model.

Chapter 6 summarizes the contributions of this work, presenting the related work and directions for future researches.

Appendix A describes the Monte Carlo simulation method, presenting a formal definition of its algorithm.

2 Key Developments in the Field of Software Reuse Cost Models

In the literature, several cost models have been proposed for estimating, predicting and analyzing the costs of software reuse. The importance of this subject is related with the explosive growth of software demand, and reuse in special, in conjunction with the perception that there is *a software crisis* in progress (Gibbs, 1994), (Mili et. al., 2002).

The term *software crisis* has been raised earlier, during the NATO Conference on Software Engineering (Naur et al., 1969), where concerns about ***low development productivity***, ***poor reliability***, ***lack of user acceptance*** and ***maintenance difficulties*** contributed for creating a set of systematic approaches for dealing with this issues (Pressman, 2004).

The work presented by McIlroy (McIlroy, 1968) during the NATO conference was the basis to establish the ideas of software reuse, which is considered a means to solve the problems associated with cost and schedule overruns (Learch, 1997), (Lim, 1998). In this context, this chapter surveys the main aspects of cost models for software reuse, including their importance for a systematic reuse adoption by organizations and a comparison among these models.

2.1. Introduction

The correct estimation of costs and benefits for software development products has been a bottleneck for a wide range of companies (Verhoef, 2005). While financial economics is well established and applied in many areas, the use of cost models it is not a trivial issue when considering the context of software reuse approach. According to Frakes et al. (Frakes et al., 1994), **economics** is one of the six factors that have direct impact on the adoption of software reuse, and the effective use of cost models can increase this level of adoption by helping organizations in their decision-making tasks.

2.2. Motivation

The activities of estimating, predicting and monitoring the costs of software development life cycle are an important part of a software process.

Within the context of software reuse, most decisions can be rationalized in terms of economic considerations. Software reuse processes are intrinsically dependent on the cooperation of many parts (Mili et al., 2001). Each part of a process can have its goals quantified in economic terms and its achievement can determine the success of the entire reuse program.

The study of reuse cost models can highlights many aspects that have a direct impact on the adoption of software reuse. Accurate software cost estimation is critical to CEOs, project managers, developers, and, at last instance, to the costumers. They can be used for generating request for proposals, contract negotiations, scheduling, monitoring and control (Leung et al., 2001). In general, cost models can help organizations to make decisions concerning reuse investment. They enumerate reuse costs and benefits and break down them into combinations of parameters and data that can determine whether or not to invest in reuse (Wiles, 1999).

2.3. Definitions

The definition of a software reuse cost model can vary according to researchers and their viewpoint.

Guerrieri et al. define a reuse cost model as a framework "*to evaluate and compute the profitability of a reusable asset*" (Guerrieri et al., 1988). Gaffney & Durek presents their definition as "*the cost of developing software with reuse relative to that of developing software without reuse*" (Gaffney et al., 1992). Frakes & Terry describe a reuse cost model as the one that "*specifies a relationship between (reuse) metrics*" (Frakes et al., 1996). According to

Favaro, a reuse cost model is "*a way of capturing economics benefits and costs associated with a reuse program*" (Favaro, 1996). Wiles & Bott define an economic model for reuse as "*mathematical formulae that predict whether or not a reuse investment would be worthwhile*" (Wiles et al., 1998). Mili et al. defines a cost model as "*a set of investment decisions on reuse activities justified by an economic rationale*" (Mili et al., 2000). Nazareth et al. considers a reuse cost model the quantification of "*the benefits accrued through a reuse process, using standards costing techniques, comparing development without reuse and development with reuse, moderated by its accompanying costs*" (Nazareth et al., 2004). Tomer et al. introduced cost model as "*a systematic and straightforward way of calculating the overall cost of various reuse alternatives in order to select the one that is the most cost-effective*" (Tomer et al., 2004).

In this dissertation, the vision of Rothenberger et al. (Rothenberger et al., 2004) for a software reuse cost model will be adopted:

"*A cost model for software reuse is the notation of the explicit costs and benefits associated with a reuse program.*"

This vision is adequate to the purpose of this dissertation, since we are interested in the study of the features for reuse cost models in order to express the costs and benefits that are related with the adoption of a reuse program.

2.4. Basic Features

Since of the late 80's there has been a proliferation of software reuse cost models. Even these models appear to be dealing with the same problem they differ significantly from each other (Mili et al., 2000). In order to understand the characteristics of a given cost model, it is important to understand the characteristics of the reuse program on which the model is to be applied.

According to the reuse program, there is a set of features that distinguish among these different cost models (Mili et al., 2001). These features are presented in the next sections.

2.4.1 Investment Cycles

The most of decisions related with software reuse can be modeled as *Return of Investment* (*ROI*) analysis. *ROI* typically measures the relation between the costs savings and the cost of investment (Böckle et al., 2004). It also can be expressed as the following equation (Boehm et al., 2003) (Erdogmus et al., 2004):

ROI = (benefits – costs) / costs (Equation 1)

Within the context of a software reuse process, there are four distinct investment cycles (Mili et al., 2001): the ***Corporate Investment Cycle***, the ***Domain Engineering Investment Cycle***, the **A*pplication Engineering Investment Cycle***, and the ***Component Engineering Cycle***. Each of these cycles provides a specific economic rationale and they can be expressed by a variety of economic functions. The main feature of a generic reuse cost model is the specification on how these investment cycles will be dependent from the information generated internally. This dependency of information propagates the cost values from one cycle to the next.

- **Corporate Investment Cycle**. At this cycle, the costs can be quantified by the consolidation of reuse infrastructure and the costs to initiate a reuse program. It includes purchasing and installing a repository to hold reusable assets; specialized personnel required hiring and training; operational and physical modifications within the corporation; and, the cost of initially populating the reuse library. It also includes the cost to adapt the organization process in order to accommodate the new activities that will be performed within the reuse program. In addition, all domain engineering costs of all projects must be considered as *episodic costs*. This cycle incorporates all benefits generated by all application engineering cycle activities.
- **Domain Engineering Investment Cycle**. It addresses the costs of domain analysis plus component engineering costs, which involves the cost to develop and catalog assets for reuse, including the cost to perform domain engineering activities, such as variability and commonalities features for each software component. The term *domain* is used to

describe a specialized body of knowledge, an area of expertise, or a collection of related functionality (Clements et. al, 2001). The term *domain engineering* expresses the development of the core asset base and an acquisition strategy (Clements et al., 2001). Its benefits are calculated by the sum of all benefits generated during the development of the components used as part of the domain, according to its frequency of reuse (Mili et al., 2001).

- **Application Engineering Investment Cycle**. This cycle has its costs defined by reuse adoption costs, which involves training activities, operational impact of reuse process and tools acquisition. According to Clements et al. (Clements et. al, 2001), the term *application engineering* represents the development of a product by using the core asset base. Other episodic costs must consider the operational risks on using significant software components and the purchase of them in a third-part market. Its benefits can be estimated by the costs savings achieved by using reusable assets for the development of a new software product rather than writing custom code for the same set of assets on the same product.
- **Component Engineering Investment Cycle**. At this cycle, costs are calculated by the development of assets for reuse and the library overhead, which includes costs for component certification and library insertion, according to its quality model attributes (Álvaro et al., 2006). Episodic costs are calculated by library operation and maintenance during a specific period of interest. Its benefits are gained from productivity increasing, which can be estimated by the difference between custom development and reuse cost, and quality level increasing, which is expressed by the maintenance costs over the lifetime of reusable assets.

A summary of the investment cycles is presented in Figure 2.1.

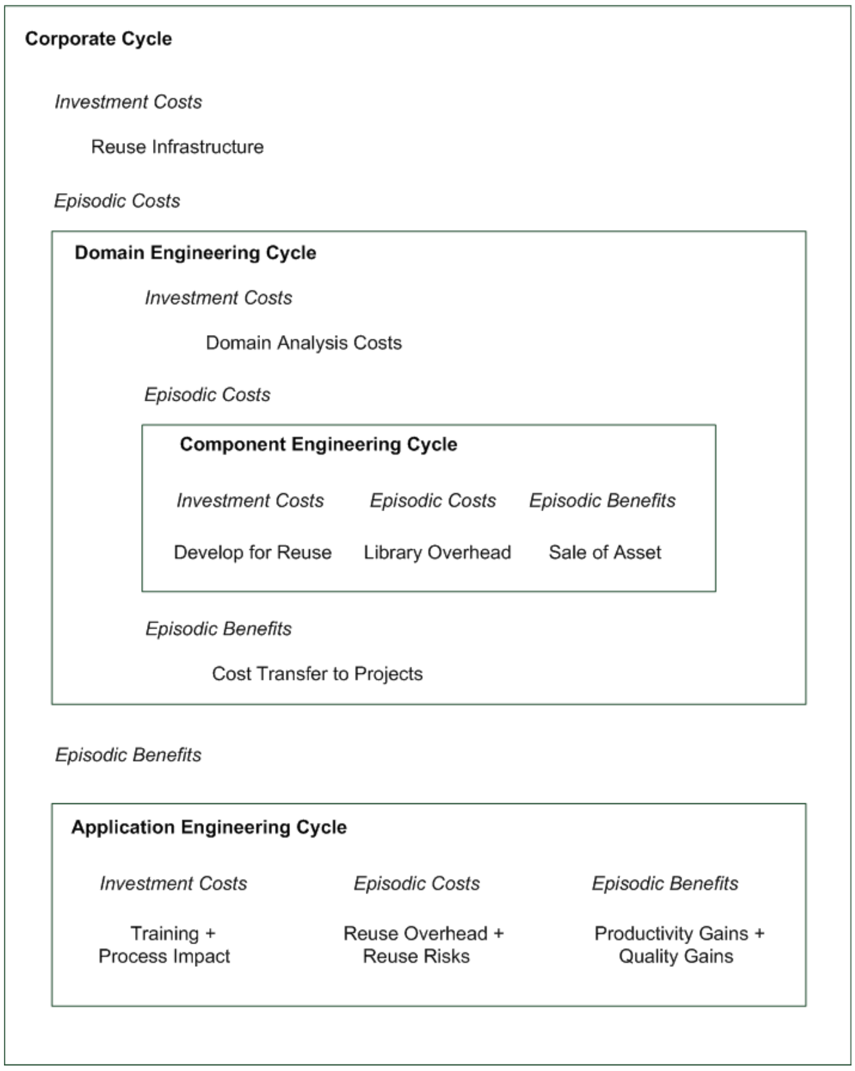


Figure 2.1 – Investment Cycles (Mili et al., 2001)

2.4.2 Cost Factors

According to a given investment cycle and a given set of economic functions it is possible to define the aspects of the reuse decision that will be taken into account. At general, investments can be quantified by six cost factors (Mili et al., 2001): *investment cycle*, *discount rate*, *start date*, *investment costs*, *episodic benefits*, and *episodic costs*.

- **Investment Cycle**. Denoted by Y, this factor is measured in number of years and counted from a start date;
- **Discount Rate**. Denoted by d, it is an abstract quantity that reflects the time value of money within a year. If a unit of money (e.g. Brazilian Real,

American Dollar) is spent today, it can get back at least *1 + d* units one year after a given start date.

- **Start Date**. Denoted by *SD*, it means the date where the investment starts and the initial costs are incurred.
- **Investment Costs**. Denoted by *IC*, this factor is measured in persons-month (*PM*) and it expresses the amount of effort required to start an investment in a given start date (*SD*). Persons-month has become a standard metric to specify the effort required to build software [also known as man-month (Brooks, 1995)]. This cost factor can also be measured in monetary terms or another measure of software size, such function points (Albrecht, 1979). A conversion of these alternative measures to persons-month is strongly recommended.
- **Episodic Benefits**. Let *y* a year, where $SD + 1 \le y \le SD + Y$, with SD as the start date of the investment and Y as the investment cycle. This factor is denoted by *B(y)* and expressed in persons-month. Applying *B* to the year of SD implies in $B(SD) = 0$.
- **Episodic Costs**. Let *y* a year, where $SD + 1 \le y \le SD + Y$, with SD as the start date of the investment and Y as the investment cycle. This factor is denoted by *C(y)* and expressed in persons-month. Applying C to the year of SD implies in $C(SD) = IC$, where *IC* is the investment cost starting in *SD*.

2.4.3 Economic Functions

Favaro (Favaro, 1996) identifies five different economic functions that can be applied on a reuse cost model: *Net Present Value*, *Payback*, *Average Return on Book Value*, *Internal Rate of Return*, and *Profitability Index*.

- **Net Present Value**, denoted by *NPV*, is measured in persons-month and can be defined by the following equation:

$$NPV = \sum_{z=0}^{Y} \frac{B(SD+z) - C(SD+z)}{(1+d)^z} \quad \text{(Equation 2)}$$

where *SD* is the start date of the investment, *d* is the discount rate, *Y* the investment cycle in years, *B* is the benefit function and *C* is the cost estimation function. An investment is valuable whenever the *NPV* exceeds zero. This leads directly with the concept of the ***present value*** of a predicted ***cash flow*** (Favaro et al., 1998).

- **Payback**, denoted by *PB*, is defined as the shortest investment cycle that makes the *NPV* a positive number. It can be expressed by the smallest integer value *x* in *Y* such that

$$\exists x \in N, 0 \le x \le Y, \sum_{z=0}^{Y} \frac{B(SD+z)-C(SD+z)}{(1+d)^z} \ge 0 \quad \text{(Equation 3)}$$

where *SD* is the start date of the investment, *d* is the discount rate, *Y* the investment cycle in years, *B* is the benefit function and *C* is the cost estimation function. An investment is worthwhile if *PB* is smaller than the amount of time necessary to amortize the investment.

- **Average Return on Book Value**, denoted by *ARBV*, is defined in function of an *amortization schedule* of the investment cost over the investment cycle. It can be used considering a software component as a *capital asset*. To calculate *ARBV* it is necessary to define an amortization function, *Am(y)* that satisfies $\sum_{z=1}^{Y} Am(SD+z) = IC$, where SD is the start date of the investment cost IC for an investment cycle Y. The *ARBV* is given by the following equation:

$$ARBV = \frac{1}{Y \times IC} \times \sum_{z=1}^{Y} \frac{B(SD+z)-C(SD+z)-Am(SD+z)}{(1+d)^z}$$

(Equation 4)

where *SD* is the start date of the investment, *d* is the discount rate, *Y* the investment cycle in years for the investment cost *IC*, *B* is the benefit function, *C* is the cost estimation function and *Am* the amortization function starting in SD. Although Favaro & Mili discourage the use of this factor, due to the subjective accounting distortions, it is possible to assign $Am(y) = \frac{IC}{Y}, \forall y, SD+1 \le y \le SD+Y$. The *ARBV* function means that if the book rate of return meets some target defined by a manager or financial analyst (e.g. 20% of profitability a year) then the capital can be invested for a set of assets (in the reuse context, the assets can be software components).

- **Internal Rate of Return**, denoted by *IRR*, defines the value of the discount rate *d* that makes the $NPV = 0$. An investment is worthwhile if IRR is smaller than the corporate *d*.
- **Profitability index value**, denoted by *PI*, is defined by the following equation:

$$PI = \frac{1}{IC} \times \sum_{z=1}^{Y} \frac{B(SD + z) - C(SD + z)}{(1 + d)^{z}} \quad \text{(Equation 5)}$$

where *SD* is the start date of the investment, *d* is the discount rate, *Y* the investment cycle in years for the investment cost *IC*, *B* is the benefit function and *C* is the cost estimation. This equation calculates a pro-rate of the potential profit over the investment cost (*IC*). An investment is valuable if *PI* is greater than 1 and it is more attractive when PI grows.

As an extension of these factors, the Return on Investment (ROI) can be redefined as a relation between the net present value *NPV* and the investment costs *IC*:

$$ROI = \frac{NPV}{IC} \quad \text{(Equation 6)}$$

2.4.4 Viewpoints

Within the context of software reuse program, there are many stakeholders involved, with different roles concerning the economic analysis of the entire program (Mili et al., 2000). Each of them has a particular vision of what will influence the interpretation of the economic functions. The main roles identified in the literature includes corporate managers, domain engineering teams, application engineering teams, individual producers of reusable assets, quality assurance engineers, and so on.

Each stakeholder has a different ROI equation, according to the cost factors presented previously. Only two of these factors can be considered uniform for an entire organization: Y and d. The remaining cost factors [(IC, $B(y)$ and $C(y)$] are cascaded from one investment cycle to the next. The cost propagation can be observed in Figure 2.2

The different viewpoints are related with investment cycles, and they are summarized in the following:

- **Component Engineering Viewpoint**. The investment decision that can be made in this viewpoint is whether or not to develop a reusable asset, considering how much it cost to develop it, how much savings project team will achieve by reusing it, and the frequency of use expected for future projects;
- **Application Engineering Viewpoint**. The investment decision here is whether or not to adopt reuse in a given development project, considering the alignment between the project needs and the available reusable assets. One point to consider here is the level of reuse adoption that involves the project;
- **Domain Engineering Viewpoint**. This decision evaluates whether or not to initiate a domain engineering activity, considering how much development effort is needed for a specific domain and how much effort is needed for domain analysis and design activities;
- **Corporate Viewpoint**. The investment decision of this viewpoint is whether or not to initiate a reuse program, considering the expected infrastructure costs, the operational impact of reuse adoption, and the expected volume of development activity.

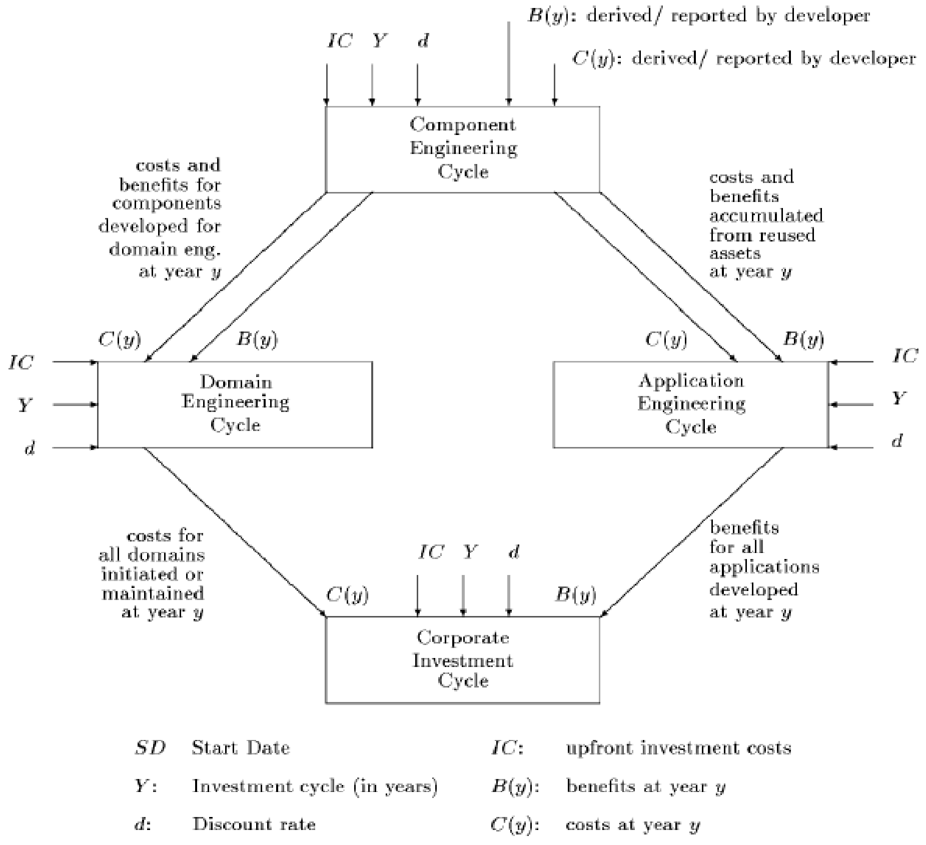

B(y): derived/ reported by developer
C(y): derived/ reported by developer
IC Y d
Component Engineering Cycle
costs and benefits for components developed for domain eng. at year y
costs and benefits accumulated from reused assets at year y
C(y) B(y)
IC Y d
Domain Engineering Cycle
Application Engineering Cycle
costs for all domains initiated or maintained at year y
benefits for all applications developed at year y
Corporate Investment Cycle
SD Start Date
Y: Investment cycle (in years)
d: Discount rate
IC: upfront investment costs
B(y): benefits at year y
C(y): costs at year y

Figure 2.2 – Cost Propagation into Viewpoints (Mili et al., 2001)

2.4.5 Reuse Organizations

Many works in literature have identified different types of reuse for an organization. Caldiera & Basili (Caldiera et al., 1991), Fafchamps (Fafchamps, 1994) and Coulange (Coulange, 1998) had summarized these types of reuse organizations:

- **Lone Producer**. Provides reuse services to at least two consumers teams. Its basic role is to design, develop and maintain reusable components;
- **Nested Producer**. In this type of organization each product team has a member dedicated for providing reuse services and expertise;
- **Pool Producer.** This organization provides two or more teams collaborating to both produce and share components;

- **Team Producer**. This type has an organizational structure based on a component producer team division. All producers team have as a target to interact with all consumers team to provide components on demand;
- **Experience factory**. This type of organization develops and packages software components upon demand. It also creates and maintains a component repository for future projects.

According to Mili (Mili et al., 2001), a generic reuse cost model can be fully influenced by the organization reuse type. As a result, a set of features appear on the study of these types:

- A clear separation between the producer and consumer teams;
- A well-defined pricing and cost structure between producer and consumer teams;
- A pricing structure to acquire assets from a third-part;
- A reward structure, in order to give some credit to the producers according to the volume of assets produced and the frequency of that assets are reused by the consumer teams;
- A well-defined metrics measurements policy; and
- A well-defined policy to track costs and library insertion procedures.

2.4.6 Assumptions

To make a reuse cost model the most generic as possible, some assumptions must be made. This implies in choosing if the costs equations will be a function of code size, in order to use an estimation method such as *COCOMO II* (Boehm et al., 1995), or other measurement method available.

Other point to note is the strategy for integration costs. Typically, this factor is measured by reused assets, adapted assets and custom developed software. In this way, a provision for integration costs must be made in a generic cost model. Assuming that μ calculates development cost in function of software size, then the cost to integrate two components with size *A* and *B* are

$$\mu(A+B) - [\mu(A) + \mu(B)] \quad \text{(Equation 7)}$$

As seen previously, the quality and productivity gains can be achieved in persons-month, measuring the cost savings by the use of reusable assets over a lifetime within the investment cycle. Time-to-market considers these savings when increased sales volume and market share can be achieved (Lim, 1996).

2.5. Classifications

Frakes & Terry (Frakes et al., 1996) defined a set of categories for reuse models. In that work, the models are categorized together with reuse metrics. This particular vision is dominant in reuse field of research, being followed by other relevant works (Nazareth et al., 2004).

The categories of reuse cost models can be summarized as the following:

- **Cost and Productivity models**. They show the cost of developing reusable assets and the cost for reusing these assets into a product. It can also present the effect of reuse on software quality and estimated schedules. Frakes & Terry state that the use of reusable assets can result in higher overall development costs. However, these costs must be recovered through many reuses;

- **Quality of Investment models**. Reuse activities are often divided into *producer* activities and *consumer* activities (Barnes et al., 1991). These activities will have influence on the quality of investment model, which will estimate the reuse benefit for all subsequent activities that profit from a reuse investment;

- **Business Reuse Metrics**. Poulin et al. (Poulin et al., 1993), (Poulin, 1997a) define a set of metrics used to estimate the effort saved by reuse and the Reuse Cost Avoidance (*RCA*). This category of model addresses the financial benefit of reuse to a project, consisting of the sum of benefits measured within the project minus the cost of building the assets for other projects to reuse.

Nazareth & Rothenberger (Nazareth et al., 2004) characterize the models as simple *metric-based* or *cost-based*. Mascena (Mascena, 2006) also divides the models into two categories, called *Economic Oriented Metrics (EORM)* and *Software Structure Oriented Reuse Metrics (SORM)*, with almost the same meaning of the work of Nazareth & Rothenberger. Next section will survey the main features of the most significant cost models for software reuse.

2.6. State-of-the-Art

Since the reuse field becomes a promise to solve the quality and productivity issues in software development, various cost models have been created to predict the costs and benefits associated with a reuse program. Lim (Lim, 1996) describes the main features of seventeen models, pointing out how to select the most suitable for a given organization, including its equations for cost factors and economic functions. Wiles (Wiles, 1999) make a comparison between twenty-four models, trying to find out the most accurate through a validation process.

As a summary of the works cited in the beginning of this section, it will be presented the most relevant cost models for software reuse found in the literature.

In 1990, Bollinger & Pfleeger (Bollinger et. al., 1990) presented equations to calculate the costs associated to activities of a project. Because of this, all costs are attached with reusable components and distributed by an amortization schedule. One year later, Barnes & Bollinger (Barnes et. al., 1991) defined the reuse investment relation as the comparison of the reuse investments with reuse benefits. This model is totally project-centered, defining a reuse investment as any cost that does not support directly the completion of an activity of the primary development goals but is instead intended to make more work products of that activity easier to reuse. Reuse benefits are the difference between the activity cost *with* and *without* reuse.

Next, Gaffney et al. (Gaffney et. al., 1992) proposed a combination between domain engineering costs and application engineering costs, using a unique equation to achieve the total costs. It does not consider the integration costs and predetermines the number of applications that make up the engineering domain effort. It also does not provide a pricing structure for domain engineering team and application engineering team. Finally, there is no provision for COTS external acquisition.

In the same year, Margano & Rhoads (Margano et. al., 1992) defined a model that the savings by reuse are based on the productivity rate (*Source Lines of Code / Labor Month*) and monthly labor rate (*money unit / labor month*) of the producer and the consumer with some additional costs (management overhead, problem analysis, error correction, and code reintegration costs). This model considers the costs at component and project level and no modification or adaptation costs are taken into account. In addition, it addresses the savings on the design phase, but there is no cost account for system reliability, understandability and maintainability of the reuse component.

Next, the model defined by Schimsky (Schimsky, 1992) addresses the relationship between cost functions drivers and *size of lines of code (SLOC)*. The cost factors are dependent on the cost to develop, maintain and provide the code. They are dependent on the investments and periodic costs of component engineering cycle. The benefits are defined as the cost avoided by not developing the code from scratch, using a relation between the development cost with and without reuse, and the breakeven point. This model does not consider maintenance and reliability costs.

In 1993, Poulin et al. (Poulin et. al., 1993) defined a model with a clear focus on application engineering cycle costs and it does not take into account the domain engineering costs. It calculates the ROI based on the *Internal Return of Investment (IRR)*, assigning the corporate reuse startup costs as the sum of the savings over all the revenue years minus the costs divided by ***(1+d)***, where *d* is the discount rate. Poulin et al. model also calculates the *NPV* by the difference of the ROI and the initial costs and introduces two fundamental metrics for software reuse cost models: ***Relative Cost of Reuse (RCR)*** – the cost of writing reusable assets -, and the ***Reuse Cost Avoidance (RCA)***, which quantifies the financial benefit of reuse. *RCA* metric can be considered a pattern for corporate viewpoint and a milestone for reuse cost model research field (Mili et al., 2001).

In the same year, Malan & Wentzel (Malan et al., 1993) described a model that uses a set of cost factors from development and maintenance phases, which includes reuse-specific overhead, setup costs, and development with and without reuse. It includes the time value of the money for costs, and introduces the *uncertainty factor*, whether the asset will be used or not. The model was validated through a hypothetical scenario. Furthermore, though time-to-market gains are discussed, they are not quantified.

In 1994, Frakes & Terry (Frakes et. al., 1994) introduced a set of reuse level metrics and frequency metrics, and they made a comparison between internal and external reuse, presenting the concept of *threshold levels*, which addresses the question of when the reuse is to be applied. This work does not define a model itself, but some cost equations that can be assigned to the previous metrics. The viewpoint of this work is focused in both application and corporation engineering.

In the same year, Kain (Kain, 1994) and Lim (Lim, 1994) proposed two new models concerning reuse investment. Kain proposes an object-oriented model creating an abstraction with a set of reusable assets. It takes into account only the corporate level and it does not address the time variance of resources. It also does not estimate the quality and productivity gains. On the other hand, Lim defined a model with a set of *Net Present Value* equations to calculate the quality and productivity gains of a reuse program. It takes the estimated value of reuse benefits and subtracts it from its associated costs, taking into account the "*time value of money*". It defines cost factors for time-to-market and risks management events, but it is considered a non-practical model due the difficulties to assess those factors.

In 1995, Boehm et al. (Boehm et. al., 1995) introduced its COCOMO 2.0 as an evolution of the original COCOMO (Boehm, 1981). It incorporates the reuse paradigm into the previous model and focuses on the component level lifecycle costs, expressed in man-months. Boehm et al. model defined a cost factor called *ruse* that expresses the costs of domain engineering activities. This model also addresses the costs of application engineering by a factor named *ESLOC*, which prorates the size of reused software as a fraction of newly developed software. The most significant cost models for software reuse uses COCOMO 2.0 to estimate the cost factors for component and application levels.

The reuse investment analysis field had a great improvement with the work performed by Favaro (Favaro, 1996) which argues on how the *NPV* can be the best function to estimate reuse costs and benefits. He discusses the characteristics of *NPV*, focusing on its additive nature, its immunity to arbitrary factors, and its provision for the time value of money. Favaro also focuses on component engineering, specifically in the economics of COTS production and marketing. However, his analysis is not carried out from the viewpoint of an asset developer, but rather from the viewpoint of a corporate manager. It implies that the model do not consider the investment analysis for lower level activities, such domain engineering and product engineering.

In the same year, Mili (Mili, 1996) defined a set of metrics to calculate the ROI associated with component development for reuse, and also the reusability in adapting it from a project for the purpose of reuse. This model has a focus on component level costs and potential domain engineering benefits. Again in 1996, Devanbu et al. (Devanbu et. al., 1996) presented a model that has an axiomatic approach for a reuse benefit function. It reflects not only how much code is being reuse, but also in what manner it is being reused. They had tested the function through a group of empirical data and a comparison with other models has been made.

Two years later, Favaro et al. (Favaro et al., 1998) reinforce the focus on NPV to represent the net totally of all contributions to the value of an investment. Favaro et al. expands his previous work by the use of the *Capital Pricing Asset Model (CAPM)* to determine the discounted cash flow, providing a method for calculating the time value of money over the operational benefits and costs. Favaro et al. also presented two techniques for evaluating investments: *Decision Tree Analysis* and *Contingent Claim Analysis*.

The University of Southern California (USC) proposes in 1999 a cost model that estimates the manpower for software development using COTS products (COCOTS, 1999). The authors divide the model into two parts: *Early Design* and *Post Architecture*. The difference among them is how early the product is deployed. The level of information and the required precision can also contribute to distinguish the values estimated. COCOTS define five cost factors: costs of candidate component for assessment, component tailoring, glue code generation, system level programming, and verification/validation.

In the same year, Wiles (Wiles, 1999) performed a study for defining a generic cost model. Wiles do not define a model itself, but he creates a framework that summarizes the main features of a generic cost model. According to Wiles, in order to evaluate a cost model it is necessary to define its development benefits and its development and maintenance effort. An investment analysis framework to estimate the system and corporate viewpoint can use these cost factors.

In the next two years, the most relevant study in reuse cost models were performed by Mili (Mili et al., 2000), (Mili et al., 2001). In the first work, Mili et al. define a model that calculates the *NPV* and the *Profitability Index (PI)* for corporate, domain engineering, application engineering and component engineering cycles. The cost values estimated by the model can interpreted in many different viewpoints, depending on the stakeholders need for decision-making. The main feature of this model is the relationship of the different levels of information: cost estimation for one level of reuse decision propagates itself to the next. The second work performed by Mili et al. defined an integrated cost model for software reuse. This model was defined by an extensive study of the most important cost models available on the literature and it can be elected as a fundamental milestone in this field of research.

After the works of Mili et al., only the study performed by Nazareth et al. (Nazareth et. al., 2004) can be considered significant to the reuse cost model research field. In this work, Nazareth et al. attempt to examine the benefits of software reuse according to the models adopted by the organizations. It creates a classification of reuse cost models as *metric-based models* and *cost-based models*. Next, it describes a domain-specific software reuse model, which is based on the computation of reuse costs and the reuse rate. This work also makes a study comparing the size of asset repository and the cost savings related with this metric. Another aspect of this model is a comparison of its cost curves with the most relevant models for software reuse.

A summary of the most important cost models for software reuse can be viewed in the timeline at Figure 2.3.

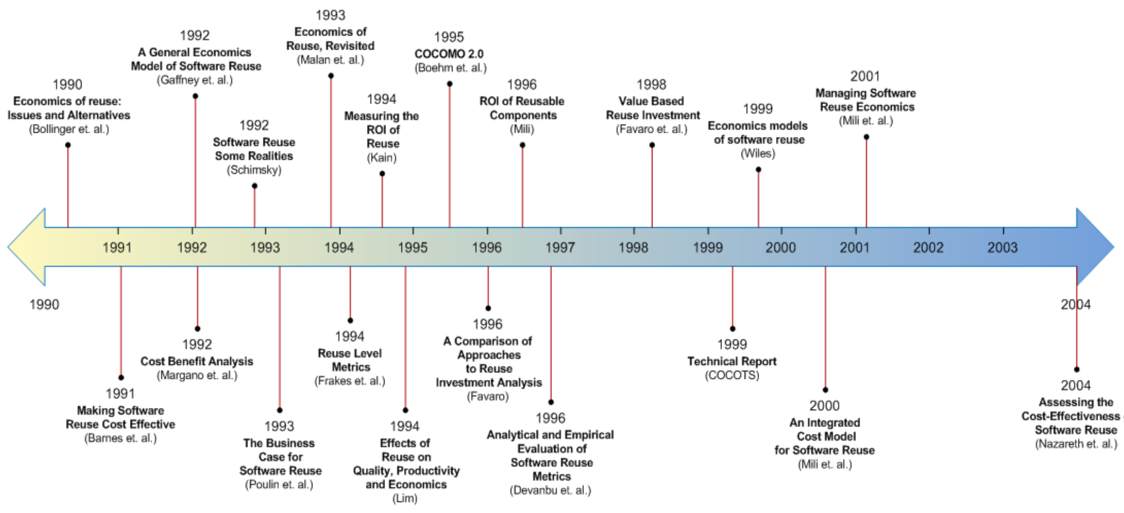

1990 Economics of reuse: Issues and Alternatives (Bollinger et. al.)
1992 A General Economics Model of Software Reuse (Gaffney et. al.)
1992 Software Reuse Some Realities (Schimsky)
1993 Economics of Reuse, Revisited (Malan et. al.)
1994 Measuring the ROI of Reuse (Kain)
1995 COCOMO 2.0 (Boehm et. al.)
1996 ROI of Reusable Components (Mili)
1998 Value Based Reuse Investment (Favaro et. al.)
1999 Economics models of software reuse (Wiles)
2001 Managing Software Reuse Economics (Mili et. al.)
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
1991 Making Software Reuse Cost Effective (Barnes et. al.)
1992 Cost Benefit Analysis (Margano et. al.)
1993 The Business Case for Software Reuse (Poulin et. al.)
1994 Reuse Level Metrics (Frakes et. al.)
1994 Effects of Reuse on Quality, Productivity and Economics (Lim)
1996 A Comparison of Approaches to Reuse Investment Analysis (Favaro)
1996 Analytical and Empirical Evaluation of Software Reuse Metrics (Devanbu et. al.)
1999 Technical Report (COCOTS)
2000 An Integrated Cost Model for Software Reuse (Mili et. al.)
2004 Assessing the Cost-Effectiveness of Software Reuse (Nazareth et. al.)

Figure 2.3 – Reuse Cost Models Timeline

2.7. Models Comparison

In order to understand the main aspects of reuse cost models, some authors emphasized the *investment side* of them (Bollinger et al., 1990), (Malan et al., 1993), (Favaro, 1996), (Poulin, 1997a),. In general, the assessment of the worth of reuse as an investment can be divided into two activities (Wiles, 1999): **(i)** Cost Estimation and **(ii)** Investment Analysis.

(i) Cost Estimation. All activities demanded by a reuse program can assign financial values determining the cost estimation as a collection of *cash flows*.

(ii) Investment Analysis. When a reuse program produces a set of cash flows by a cost estimation model, its results can be used for comparing the time value of money. After this, management can start a decision-making process, in order to evaluate the cost-benefits factors.

All reuse activities cost factors must be converted into cash flows in order to complete the investment analysis. To perform this is a challenge for a reuse program due to the intangibility of the features involved (Wiles, 1999). However, it is recommended to include these intangible factors to take costs into account.

Another point to consider when analyzing cost models is the uncertainty factor related to estimation output. There is a risk encapsulated on these estimations and they must be taken into account. In (Favaro et al., 1998) the risk analysis for reuse costs includes activities to estimate a unique risk for the entire reuse program and also estimates the market risks handled by altering the discount rate.

In this comparison all economic models captures cost estimation, including factors to determine cost and benefits. In addition, all cost values can be rearranged to produce financial values. Some sort of investment analysis is included on these models, with some level of complexity distinguishing them. Some models simply subtract costs from benefits, while another group uses more sophisticated probabilistic models to perform this.

In the major part of models, cost estimation and investment analysis can be considered independent techniques and the separation between them is straightforward. Cost estimation is the part of models that have a reasonable level of variance, while investment analysis presents some level of uniformity among the models surveyed.

One common aspect is the cost summing and averaging over subparts, which will compose the total cost for a specific viewpoint. The level of granularity varies according to the model, but in most cases its addresses the costs of systems, subsystems, software components, reuse of a component and code units (e.g. LOC). In some models, the cost to locate a software component and modify it is considered as an important part of the overall cost.

Development cost can involve the building of new reusable software from the scratch or adding reusability for it. Almost all models assume that systems will produce reusable software.

Some part of models assumes that an important benefit of software reuse is to increase its profits due higher sales.

The most preferred way to express the models is the *cost prediction* (Lim, 1996), which try to estimate the cost to build a system with reuse. Only a quarter of models surveyed have some kind of investment analysis using cash flow techniques. These models consider the period of one year as a standard investment cycle.

A summary of the reuse cost model features is presented in Table 2.1.

Table 2.1 – Reuse Cost Model Comparison

Model	Reuse Cycle	Economic Function	Cost Factors	Viewpoints
(Bollinger et. al., 1990)	Component eng., Corporate eng.	Reuse benefits	Component eng. with and without reuse	Corporate
(Barnes et. al., 1991)	Domain eng., component eng., application eng.	Breakeven, ROI, quality gains	Application eng. with and without reuse, comp. eng., domain eng.	Producer, consumer
(Gaffney et. al., 1992)	Domain eng., application eng.	ROI, breakeven	Application eng. costs, prorated domain eng. costs, productivity	Corporate
(Margano et. al., 1992)	Component eng.	Payback, NPV, productivity gains	Components costs, overhead, investment	Project manager, component developer
(Schimsky, 1992)	Domain eng.	Breakeven for application eng. costs	Develop, maintain and reuse code	Project
(Poulin et. al., 1993)	Component eng., application eng., corporate eng.	ROI, NPV, profitability index	KLOC	Corporate
(Malan et. al., 1993)	Domain eng., component eng.	NPV	Overhead lifecycle development	Domain manager, asset developer
(Frakes et. al., 1994)	Application eng., corporate eng.	Reuse level, reuse frequency	Number of reference to items	Project, corporate, component
(Kain, 1994)	Domain eng.	ROI	Domain eng. costs, application eng. with and without reuse	Project decisions at corporate level
(Lim, 1994)	Application eng.	NPV	Component costs, productivity, reuse with KNCSS	Corporate, project-wide
(Boehm et. al., 1995)	Domain eng, application eng.	Lifecycle costs	*RUSE*, *ESLOC*	Corporate, project
(Favaro, 1996)	Component eng.	NPV, profitability index, ARBV, Internal rate of return, payback	Cost for domain eng. and component eng.	Corporate
(Mili, 1996)	Component eng.	ROI	Component level factors	Producer, corporate
(Devanbu, 1996)	Application eng., corporate eng.	Reuse benefit	Size and structure of application	Project
(Favaro et. al., 1998)	Corporate eng.	PV and NPV	Cash flows, discount rates and risk assessment	Corporate

(COCOTS, 1999)	Application eng.	Application eng. costs	Assessment, tailoring and glue code volatility	Project, corporate
(Wiles, 1999)	Component eng.	Development and maintenance benefits	Frequency of reuse	Producer, consumer, component
(Mili, et. al, 2000)	Component eng., domain eng., application eng., corporate eng.	NPV, payback, ARBV, internal rate of return, profitability index	Investment cost, episodic costs, episodic benefits	Component, domain, application, corporate
(Mili et. al., 2001)	Component eng., domain eng., application eng., corporate eng.	NPV, payback, ARBV, internal rate of return, profitability index, ROI	Investment cost, episodic costs, episodic benefits	Component, domain, application, corporate
(Nazareth et. al, 2004)	Domain eng.	Development benefits	Reuse rate, degree of fit	Component, domain

2.8. Chapter Summary

Software reuse cost models can be defined as the notation of the costs and benefits of a reuse program.

They can be divided into a set of features, such as investment cycles, cost factors, economic functions, viewpoints, type of reuse organizations and hypothesis for its use.

This chapter presented a classification for reuse cost models and a brief description for the most relevant ones. A comparison was made in order to discuss the main aspects of those models.

In the next chapter it will be presented an evolution of reuse cost models, based on software product lines approach for a reuse program.

3 Software Product Line Cost Models: State-of-the-Art

When organizations adopt software reuse strategy, they want to decrease development costs and improve the quality level of their products. However, when a reuse program is merely focused on reusing small pieces of code, which is a simple cloning of code designed for one system for using it in another one, it has been unprofitable (Northrop, 2002). This type of reuse approach is frequently called as *opportunistic reuse* (Schmid, 2002) and it implies in reuse the assets in a non-systematic way.

In the literature, there are a large number of works that show the obstacles faced during the adoption of a reuse program that is non-systematic (Sametinger, 1997). The work conducted by Ezran et al. (Ezran et al., 2002) states that reuse is *a systematic software development practice*, which means that to be effective a reuse program must consider a consistent process in order to establish itself within an organization. The adoption of a systematic reuse process is one of the key factors for success in reuse programs (Morizio et al., 2002).

A growing research area within software reuse context is the approach related with *product line engineering* (Clements et al., 2001). It focuses the development from the perspective of a whole set of products, called **product line**, instead of individual products. In this approach, reuse happens systematically, i.e., the key characteristics of products in which the assets will be reused are already well known.

Cost models have an important role for an organization that is adopting a reuse program using the software product line approach. In this context, this chapter presents nine cost models for product line engineering and discusses the main aspects that configure an effective model.

3.1. Introduction

Software product lines approach is a relatively new concept, but it is emerging as a practical and important software development paradigm (Clements et al., 2001). It has been succeeded because organizations exploit their commonalities and variability in applications to achieve economies of development.

There are several case studies presenting the benefits on adoption of software product lines (Northrop, 2002). **CelsiusTech Systems**, a Swedish company supplying control systems for defense navies, used a product line to deliver more than fifty systems based on the same set of assets. They have savings by shortened delivery schedules, allocating a smaller staff to produce more systems. Their software reuse level is about 90% (Brownsword, 1996).

Cummins, the world's largest manufacturer of diesel engines, saved almost a year when developing their engine control software (Northrop, 2002). This fact was possible by adopting a product line approach, offering a mix of features and platforms that otherwise would require almost four time their current staff.

The **US National Reconnaissance Office** outsourced their assets development, creating a product line to spacecrafts command and control software (Northrop, 2002). They saved a 50% in the overall cost and schedule, and decreased the number of development staff and defects.

Market Maker Software, a German development company, produces the most popular stock market software in Europe. They adopted a product line approach and it takes their product available for customers as few as three days, even if the software must be tailored (Northrop, 2002).

Other key players reported success stories when using product line engineering. The cases were derived from **Alcatel** (Coriat, 2000), **Hewlett-**

Packard (Toft, 2000), **Philips** (Philips, 2000), **Boeing Company** (Sharp, 2000), and **Robert Bosch GmBh** (Thiel et al., 2000).

Despite the success in adopting software product line reported in the works cited previously, the community needs more quantitative data to support that approach due to move to product lines implies in considering reuse as an investment (Wiles, 1999). The next sections will discuss about software product line features, cost models for software product lines and a study for achieving an effective model.

3.2. Software Product Lines

According to Clements et al. (Clements et al., 2001) a software product line can be defined as a *set of "software-intensive systems that share a common, managed feature set satisfying a particular market segment needs that are developed from a common set of core assets".*

Core assets are the basis for a software product line and they often include architecture, reusable software components, domain models, requirement statements, documentation, performance models, schedules, budgets, test plans, test cases, work plans, and process description.

In a product line, each system is a product in its own right. Each product is created by taking specific components from a core asset base and managing the variation among them. New components can be added to the core asset base and they can be assembled according to the rules specified by a common architecture. The term *development* in a product line can describe how the core assets and products will be developed. It implies that software development within the context of product line can occur in three different ways (Northrop, 2002):

- **Make it**. The organization builds it from the scratch or by mining legacy systems;
- **Purchase it**. The organization acquires *Components Off-the-Shelf* (COTS) from the market and provides its integration for a product;
- **Commissions it**. The organization contracts with a third-part the development.

In summary, the term *development* can be assigned for a set of activities that involves building, acquiring, purchasing, integrating, or combinations of them.

We agree with the SEI's vision (Clements et al., 2001) that distinguishes between the terms *domain* and *product line*. Domain can be described as a specialized body of knowledge, an area of expertise, or a collection of related functionality. Core asset development can be defined as a **(i)** *domain engineering* activity and product development as a **(ii)** *product engineering* activity. In addition, in these activities reuse must be planned, enabled and enforced, requiring a certain level of **(iii)** *management* as an important part of the reuse program. Figure 3.1 shows the essential product line activities.

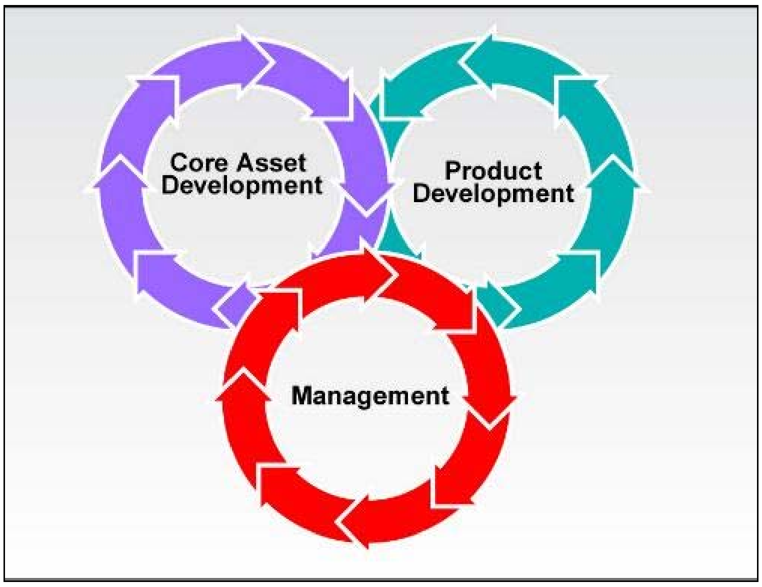


Figure 3.1 – Software Product Line Activities (Clements et. al., 2001).

3.2.1 Domain Engineering

The main target of core asset development is to establish a production capability for its products. It is an iterative activity, with its inputs and outputs affecting each other, as presented in Figure 3.2. The inputs to core asset development include:

- **Products constraints**. Commonalities and variations among the products that will be used to produce the product line, including a set of behavioral features;

- **Production constraints.** Standards and requirements that apply to the products in the product line;
- **Style, patterns and frameworks**. The architectural pieces meeting the product and production constraints;
- **Production strategy**. The overall approach for building the core asset. It can be categorized as top-down strategy (starting with a set of core assets and creating products with them) or bottom-up strategy (starting with a set of products and generalizing its components to produce the assets); and
- **Inventory of existing assets**. Software and other organizational artifacts available that can be included in the asset repository.

The outputs to core assets development include:

- **Product line scope**. Describes the products that will compose the product line;
- **Production plan**. Describes how products are built from the core assets; and
- **Core assets**. The elemental components to produce products in a product line.

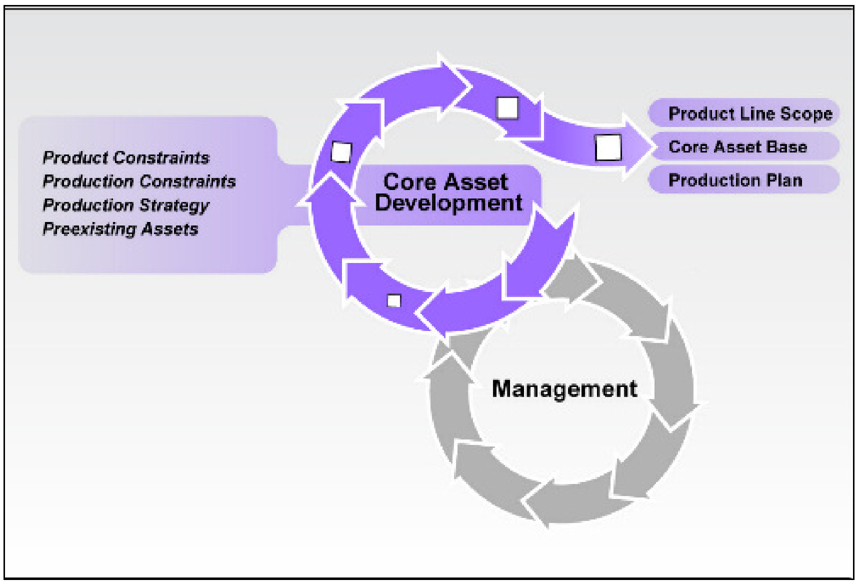


Figure 3.2 – Core Asset Development (Clements et. al., 2001).

3.2.2 Product Development

This activity is dependent from the requirements for specific projects. It also can vary according to the assets, production plan and organizational context.

As the same way the domain engineering, product development is also an iterative activity, as seen in Figure 3.3. Creating products affects the product line scope, production plan, core assets and requirements for specific products.

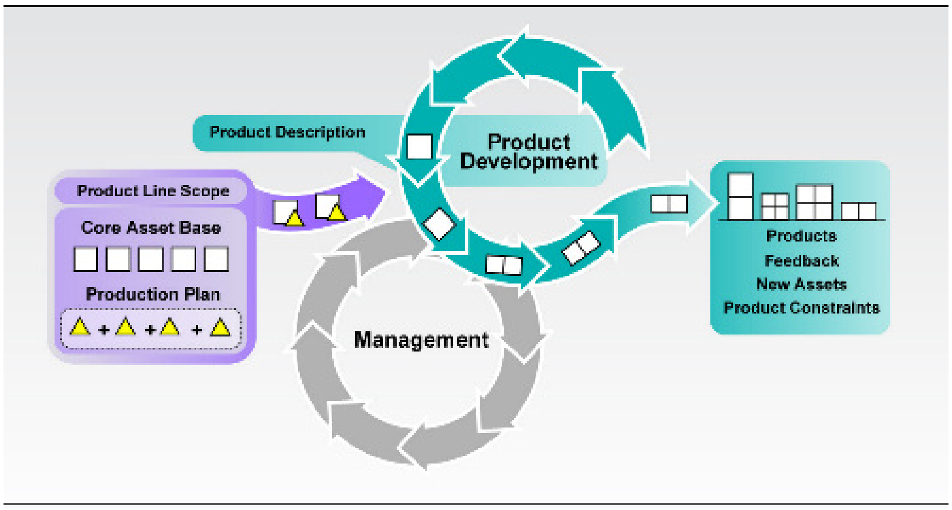


Figure 3.3 – Product Development (Clements et. al., 2001).

3.2.3 Management

In product line context, there are two types of management: ***technical*** and ***organizational***. The first type addresses the core asset development and product development activities. It must ensure that the development staff will perform its task according to the process defined for the product line. In addition, technical management has to collect data to track project progress. The other type of management takes into account the organizational structure. It can determine the necessary funding to core asset evolution and coordinates the technical activities and iterations between core asset development and product development.

Finally, it must address the risks mitigation within a product line, ensure the perfect communication between customers and suppliers and create an adoption plan in order to achieve the organizational goals.

3.2.4 Product Line Engineering

Product line engineering focuses on producing multiple variants of a system by exploiting the commonalities among systems in the form of reuse (Toft, 2000). The key to successful product line engineering is to identify early an architecture that provides a guide to build the products in a product line (Bayer et al., 1999).

According to the work conducted by SEI (Clements et al., 2001), under the umbrella of the three essential areas there are 29 *practice areas* that must be mastered for a successful product line. A practice area is a "*body of work or a collection of activities*". In a product line context, each practice area has a particular significance and they can be categorized as following: **(i)** software engineering, **(ii)** technical management, and, **(iii)** organizational management. Figure 3.4 presents the relationships among categories of practices areas.

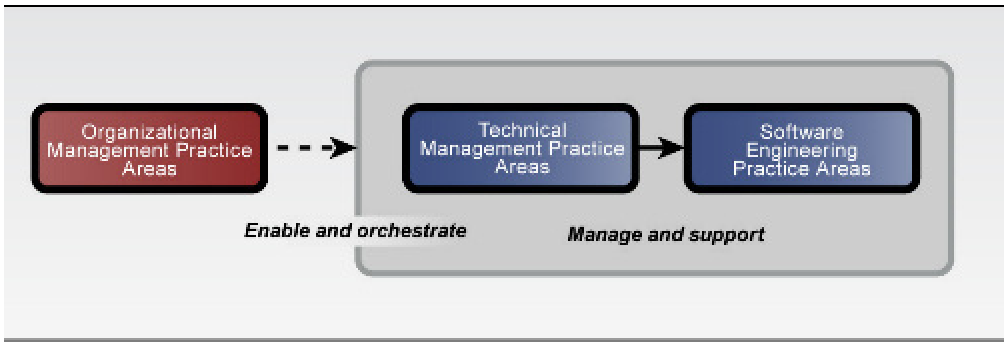

Organizational Management Practice Areas
Technical Management Practice Areas
Software Engineering Practice Areas
Enable and orchestrate
Manage and support

Figure 3.4 – Relationships among Categories of Practice Areas (Clements et. al., 2001).

3.3. A Survey on Cost Models for Software Product Line

One important aspect of a reuse process is to determine its effect on software attributes, such as cost, quality and time-to-market (Schmid, 2002). Basically, cost is the only attribute that can be valued in an absolute manner. Within this context, software development managers want to predict the costs and benefits of a development approach.

Product line engineering is often the most economical choice in a long run, but this fact can only be verified if the organization has a solid framework in order to calculate cost factors and benefits during a given period of time (Clements et al., 2001). Another point to consider is the various scenarios that can occur in a product line that brings to the organization some difficulties to distinguish among software reuse approaches. A cost model for software product line can help distinguish these situations. Next topics will present the most relevant cost models found in the literature and their main features.

3.3.1. Poulin's Cost Model for Software Product Lines

Poulin (Poulin, 1997b) uses two parameters for estimating the effects of reuse: ***Relative Cost of Reuse (RCR)*** and ***Relative Cost of Writing for Reuse (RCWR)***.

RCR is a ratio that compares the effort needed to reuse software without modification to the costs associated with developing the same software for a regular and single use. This metric, when applied to a set of assets, can predict the percentage of effort needed to develop them using the comparison among the two situations.

RCWR is a value that compares the costs of creating reusable assets to the cost of writing software for a unique usage. It also measures the effort in some percentage value.

Poulin's model uses *RCR* and *RCWR* to calculate two additional values, predicting the savings for an entire project: ***Reuse Cost Avoidance (RCA)*** and ***Additional Development Cost (ADC)***.

RCA compares the savings of reusing assets over writing the equivalent software for a single use. The RCA can be calculated by two new values:

- **Development Cost Avoidance (DCA)**, which measures the savings based on the total software reused and the cost of reusing that software. It can be assigned according to the following equation:

$$DCA = RSI * (1 - RCR) * \left(\frac{CSU}{LOC}\right) \text{ (Equation 8)}$$

where, *RSI* means the *Reused Source Instructions* percentage, *RCR* is the relative cost of reuse, *CSU* is the cost of single-use code and *LOC* the amount of lines of Code.

- **Service Cost Avoidance (SCA)**. It represents the maintenance savings related to the eliminations of repair costs. It can be calculated according to the equation:

  $$SCA = RSI * ER * EC \quad \text{(Equation 9)}$$

  where RSI is the reused source instructions percentage, ER is the error rate and EC the error cost.

  Then, the RCA can be expressed in terms of the sum of DCA plus SCA.

ADC Is the cost of writing software for reuse and is based on the RCWR and the actual code written for reuse. It reflects the cost of writing the reusable assets, related with RSI, over the cost of writing the software for a single use. The equation that defines ADC is the following:

$$ADC = (RCWR - 1) * RSI * NCC \quad \text{(Equation 10)}$$

where $RCWR$ is the relative cost of writing for reuse, RSI is the reused source instructions and NCC is the new code cost.

Finally, the model of Poulin establishes the return of investment (ROI) value for the set of assets of a product line. The equation for the ROI is defined by $ROI = \sum_{i=1}^{n} RCA_i - ADC$ (Equation 11), where n represents the number of successive uses of the reusable assets.

3.3.2. ABC Approach

In (Cohen, 2003), Cohen introduces an approach to determine the investment and the projected ROI for the development of a set of reusable assets. This approach is called ABC and it is based on the following factors:

- **Applications**. The different systems that an organization might develop using the product line assets. The time period for an investment cycle must be taken into account.
- **Benefits**. The project costs savings when using the product line assets.
- **Costs**. The actual costs of reuse than an organization incurs when developing and using the assets.

In order to produce the *ROI* values, Cohen defines two values for measurement: ***Degree of Reuse (DOR)*** and ***Cost of Reuse (COR)***.

DOR represents the assets percentage of use for the development of a typical software product. *COR* is the value considers the cost of developing reusable assets and the cost of applying those assets in the development of products.

3.3.3. Schmid Model

In this model, Schmid (Schmid, 2003) addresses the formal definition of a cost model for software reuse, which can be considered as a basis for the work conducted by Böckle et al. (Böckle et al., 2004) and Clements et al. (Clements et al., 2004). Schmid defines its model into three steps, where each higher level model can be seen as a refinement and an extension of the next lower level model. The steps are:

- **First Order Model**. It allows making explicit the tradeoffs that are involved in analyzing the economics of a certain product line situation and thus in determining an optimal scope for a reuse infrastructure.
- **Second Order Model**. Addresses the time and monetary aspects in the context of reuse economics.
- **Third Order Model**. It step accounts for risks and opportunities in the context of product line economics.

As cited previously, the concept of a reuse scenario for a product line was originally described in this work, as "*a product development plan that describes which products with which features should be fielded at which point of time. In particular, a product line scenario determines a product portfolio*".

Moreover, this work approaches the investment analysis for a product line as the financial theory of *options*, which represents the right without obligation to perform a discretionary action in the future.

Finally, Schmid presents the use of the technique of decision tree analysis (Harrison et al., 2002) in order to perform risk analysis into a product line scenario.

3.3.4. Convergys Experience

The product line cost model defined by Convergys Corporation (Peterson, 2004) is based on the comparison of software product development in two scenarios: ***Independent scenario*** and ***Software Product Line (SPL) scenario***.

The Independent scenario is based on a family of products developed independently from one other. Each product has its own dedicated funding source and development organization. This scenario is considered the most common inside software organizations.

SPL scenario assumes that a set of assets common to multiple products are developed and supported by a *"component factory"*. This factory is responsible to deliver component versions to the product groups, which are responsible to integrate the reusable assets, adapting or extending them to deliver new products for specific vertical markets.

This approach focuses on the benefits associated with productivity improvement and leverage associated with establishing a common set of components upon which members of the product family would be based. To determine the benefits in that way, the model defines a *demand function*, expressed in function points per unit time, according to the equation above:

$$D_\pi = \frac{1}{T} * M_\pi \quad \text{(Equation 12)}$$

where π is set of requirements for a given product within a product line, T is the time period considered for benefits analysis and M_π a "mapping" between the requirements and the effort (expressed in functions points units) needed to build those requirements. The demand function is a fundamental aspect of the model due its utilization to calculate a set of costs and benefits factors:

- **Commonality and Leverage**: in one product line scenario, the common component factory develops the functionality common for two or more products. The product group team implements the functionality specific to a vertical market. A set of *commonality parameters* is defined to express the effort shared by other products in a product line. The leverage notion of this model addresses the fraction of new functionality produced by the component factory that benefits the k^{th} product.

- **Productivity and Throughput**: software development organizations respond to market demand by delivering new releases of their product that increase its value to potential customers. The productivity needed to deliver new releases can be calculated by the average size of new products versions over the planned period, for a product k. It also considers the average staffing level allocated to that product development and its value is expressed in function points per unit time. The throughput has the same unit value.
- **Independent and SPL Staff Level**: in the SPL scenario, the development team is allocated between the common component factory and the individual product groups. The SPL staffing is reduced due the elimination of redundant demand and due to an increase on productivity levels. The model has an equation that shows that an organization can decrease their staffing levels and still maintains throughput equivalent to the Independent Scenario.
- **Financial Flexibility:** the SPL scenario has a lower staffing requirement than Independent scenario. Based on the loaded cost per person per year the model can estimate the annual investment requirements and the cost avoidance annualized. The flexibility comes up when instead of reducing the staffing level, they can be reallocated to the component and product groups in order to increase the throughput.
- **Time Dependence of the Benefits:** during the period when the component is being deployed to various products, the benefits will have a dependence on the number of products using the component and the time period which the products are using that component.

The model itself does not define a set of benefit functions, but it presents a list of possible benefits to achieve in applying product line engineering. This list may serve as a good starting point for producing a useful set of benefit functions, as follows:

- **Research and Development (R&D) Investment**. The ability to leverage the R&D investment across the product family by reusing a common set of components.

- **Subject Matter Expertise**. The ability to leverage subject matter experts across the product family by concentrating domain subject matter experts with similar skills and knowledge into centralized groups that serve all products.
- **Productivity and Quality**. Improving productivity and quality by breaking large, monolithic applications into smaller, more manageable projects and by using components that encapsulate the functionality of applications.
- **Time to Market**. Increasing the rate of delivery of new capabilities to market and enabling new products to be delivered faster by reusing well-established components.
- **People Mobility**. Providing employees with more career development opportunities by standardizing the development environment and processes, thereby reducing the learning curve associated with a move to a new project.
- **Supplier Relationships**. Standardizing the platforms and development environment enables a more effective leverage of supplier relationships.
- **Geographic Flexibility**. Standardizing component-based development facilitates the distribution of development responsibilities across locations.
- **Sourcing Flexibility**. Using a modular architecture to enable greater flexibility to build, license, or acquire software.
- **Product Refresh**. Using a modular architecture to facilitate the process of refreshing the product family as new technologies and/or software components becomes available.

3.3.5. Tomer Model

The model created by Tomer et al. (Tomer et al., 2004) addresses the issues in evaluating reuse scenarios for product lines, defining three dimensions for all engineering activities: *development*, *maintenance* and *reuse*. The model assumes that an asset repository is already established and two cost factors are associated with it:

- **Mining.** The activities of fetching reuse candidates from specific products and copying them into the repository.
- **Acquisition.** The activity of copying artifacts from repository and integrate them into a product.

In this sense, Tomer et al. model assume that exist only two types of reusable assets: *Repository Assets* and *Private Assets*. The first stands for reusable artifacts that are stored in the repository, and the later refers to the set of artifacts that are contained within a specific product and are available either for mining or catalog or for private modification.

The model also describes two types of reuse operations: ***Transformation*** and ***Transition***. Transformation addresses activities such as adaptation for reuse, repository construction from scratch, private assets modifications and build of new assets. Transition stands for the operations to catalog assets acquired externally and other aspects from mining and cataloging processes.

One important aspect of this model is the definition of a reuse scenario as "*any sequence of elementary (reuse) operations*". Finally, Tomer et al. define the cost of reuse as the sum of the costs of operations, transformation and transition, according to the application of a cost policy.

3.3.6. Constructive Product Line Investment Model (COPLIMO)

Boehm et al. (Boehm et al., 2004) describes a software product line economics model that consists of two components: a ***Product Line Development*** Cost Model and an ***Annualized Post-Development Life-Cycle Extension***. This model is an extension of Constructive Cost Model II (COCOMO II) (Boehm et al., 1995).

The *Product Line Development Cost Model* in this model calculates values similar to RCR and RCWR from (Poulin, 1997b). To estimate RCWR, COPLIMO uses a COCOMO II multiplier, called *development for reuse (RUSE)*, and two constraints factors, namely *required reliability (RELY)* and *degree of documentation (DOCU)*. The RCWR correspondent value is expressed by the equation:

$$(1+RUSE)*RELY*DOCU \text{ (Equation 24)}$$

To calculate RCR, COPLIMO makes equivalence from size of code using a factor known as *assessment and assimilation (AA)*. Next, COPLIMO analyses the type of reuse is being used, **black-box reuse** or **white-box reuse**. According to the type of reuse COPLIMO accounts for modifications of design, modifications of code, integration effort, the level of system understanding, and, the programmer unfamiliarity. All these factors are used to calculate an equivalent size of code and allow COPLIMO to use COCOMO II traditional equations to perform effort estimation.

The *Annualized Life-Cycle Model* in this model takes into account the costs to maintain the product line within a life-cycle. To perform this it uses the COCOMO II approach, adding the initial development costs in Persons-Month to the maintenance costs based on the number of years in maintenance. Finally, the model uses an equation to estimate that costs, according to the following expression:

$$PM(N,L)=PM(N)+L*N\left(A*AMSIZE^{B}*\prod EM\right) \text{ (Equation 25)}$$

where N is the number of products under maintenance, L is the number of years, A and B are COCOMO II adjustment factors, $AMSIZE$ is the annualized maintenance size, and EM is a COCOMO II's effort multiplier.

3.3.7. Structured Intuitive Model for Product Line Economics (SIMPLE)

The model proposed by Clements et al. (Clements et al., 2005) can be used to compute estimates for various economics measures to build, sustain and evolution of software product lines. This model summarizes its features in the following situation:

*"An organization has **p_init** product lines, each comprising a set of products, and **s_init** stand-alone products. Over a period of time, the organization wishes to transition to the state in which it has **p_final** product lines, each comprising a (perhaps different) set of products, and **s_final** stand-alone products. Along the way, the organization intends to add **k** products or delete **d** products."*

Figure 3.5 shows the general scenario for SIMPLE.

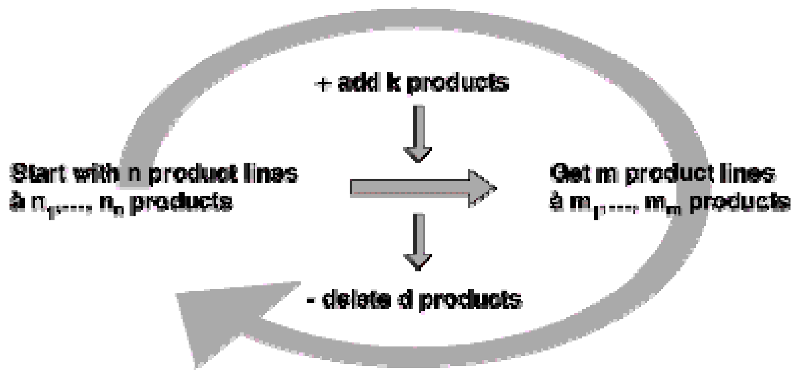

+ add k products
Start with n product lines à $n_1, \ldots, n_n$ products
Get m product lines à $m_1, \ldots, m_m$ products
- delete d products

Figure 3.5 – SIMPLE General Scenario (Clements et. al., 2005).

Clements et al. state that organizational decision makers will want to know what their plan will cost, what benefits it will bring, and how it compares with other alternatives. SIMPLE can be used to weight the costs and benefits of one or more product line alternatives, using four basic cost functions: ***Organizational Costs***, ***Core Asset Base Costs***, ***Software Unique Parts Development Costs***, and ***Asset Reuse Costs***.

Organizational Costs, denoted by $C_{org}()$, it is a function that returns how much it costs the adoption of product line approach for a set of products of an organization.

Core Asset Base Costs, denoted by $C_{cab}()$, it is a function that returns how much it costs to develop core asset base to satisfy a particular scope.

Software Unique Parts Development Costs, denoted by $C_{unique}()$, it is a function that returns how much it costs to develop the unique parts of a product that are not based on the assets in the core asset base. These parts include software and other documentation artifacts.

Asset Reuse Costs, denoted by $C_{reuse}()$, it is a function that returns how much it costs to build a product reusing core assets from a core asset base.

By the use of these four basic cost functions it is possible to establish the cost of building a product line containing *n* products, according to the following equation:

$$C_{org}(\ )+C_{cab}(\ )+\sum_{i=1}^{n}\left(C_{unique}(product_i)+C_{reuse}(product_i)\right) \text{ (Equation 13)}$$

This equation does not handle with all type of situations that can occur in a product line, but SIMPLE can be used to derive new cost functions in order to estimate the costs and benefits of several reuse scenarios.

Often is necessary to evaluate if is worthwhile to build *n* products using the core asset base or build them independently without sharing the assets. SIMPLE has a cost function called ***Cprod*** that returns the cost of building a product *pi* in a stand-alone fashion. This function is often associated with traditional software engineering cost models and can be expressed according to the following equation, to build *n* products independent from the product line:

$$\sum_{i=1}^{n} C_{prod}(p_i) \text{ (Equation 14)}$$

Clements's model allows the estimation of cost savings simply by [(Equation 14) – (Equation 13)].

To account the costs in which a product appears in a new version, SIMPLE presents a cost function called ***Cevo***. This function, when parameterized with the product and version numbers, returns the cost of building that version. Clements et al. recommend start this estimation with a historical percentage of an entire product, e.g., $C_{evo} = 0.2 * C_{prod}(\ )$. The model also defines an analogous cost function under a product line umbrella, called ***Ccabu***, which means the costs of core asset base update, when releasing a new version of a product.

One important aspect of SIMPLE is that there is no limit for benefits function creation. Clements et al. recommend that each organization can define its own benefits functions. To perform this, let assume that there is a benefit *benj*, and *B(benj)* its benefit function. Each benefit function is parameterized by the organization time period of interest, denoted by *t*. The following equation can be used as a general benefit function, for a set of *n* benefits:

$$\sum_{j=1}^{n} B_{ben_j}(t) \text{(Equation 15)}$$

Finally, SIMPLE defines nine reuse scenarios in order to capture the dynamic situations that can occur into an organization. The scenarios are summarized below:

- **Scenario 1: The cost of building a software product line**. In this scenario, an organization wants to know the costs of producing a set of products as a software product line. The equation for this scenario, considering a *t* time period and *n* products, is given by the following:

$$C_{org}(t) + C_{cab}(t) + \sum_{i=1}^{n}\left(C_{unique}(product_i, t) + C_{reuse}(product_i, t)\right) \text{ (Equation 16)}$$

- **Scenario 2: The cost of building a software product line vs. building the products independently**. This scenario calculates the savings (or losses) when considering these two situations. The equation is:

$$\sum_{i=1}^{n} C_{prod}(product_i, t) -$$

$$\left(C_{org}(t) + C_{cab}(t) + \sum_{i=1}^{n}\left(C_{unique}(product_i, t) + C_{reuse}(product_i, t)\right)\right) \text{ (Equation 17)}$$

- **Scenario 3: The cost of releasing a new version of a product line member**. This scenario calculates the cost of a new product in a product line using the equation below:

$$C_{cabu}(\ ) + C_{unique}(\ ) + C_{reuse}(\ ) \text{ (Equation 18)}$$

- **Scenario 4: Comparing costs of converting to a product line vs. the cost of evolving the existing set of stand-alone products**. This scenario calculates the cost of setting up a product line for its first evolutionary cycle, given by the equation 13. It also returns the value of evolution for a set of stand-alone products, given simply by C_{evo} for each product *i*. The equation for this option is:

$$\sum_{i=1}^{n} C_{evo}(product_i) \text{ (Equation 19)}$$

When considering the cost savings (if any) for the comparison of the two options the equation can be defined as the difference between [Equation 18] and [Equation 13].

- **Scenario 5: Return on Investment (ROI)**. In this scenario, an organization wishes to know the ROI achieved by setting up a product line. The model assumes that ROI can be calculated by dividing the cost savings from the costs of investments. SIMPLE also assumes the later as C_{org} + C_{cab}. So, the ROI after one round of evolution of a product line is expressed by the equation

$$\frac{\sum_{i=1}^{n} C_{evo}(product_i) - \left[C_{org}(\ ) + C_{cab}(\ ) + \sum_{i=1}^{n} \left(C_{unique}(product_i) + C_{reuse}(product_i) \right) \right]}{\left(C_{org}(\ ) + C_{cab}(\ ) \right)}$$

(Equation 20)

- **Scenario 6: Constructing and evolving a product line**. In this scenario an organization wishes to know the costs savings (or losses) over a given number of time periods, denoted by *nbr_periods*, for the construction of *s1* products using a product line, versus constructing and evolving them in a stand-alone fashion. To calculate the benefits associated with this scenario, SIMPLE *translates nbr_periods* into a number *p* of evolutionary updates. Finally, this scenario has the following equation:

$$\sum_{t=1}^{nbr_{periods}} \left[\sum_{i=1}^{s1} C_{prod}(product_i, t) \right] -$$

$$\sum_{t=1}^{nbr_{periods}} \left[\left(C_{org}(t) + C_{cabu}(t) + \sum_{i=1}^{nj} \left(C_{unique}(p_i, t) + C_{reuse}(p_i, t) \right) \right) \right]$$

(Equation 21)

- **Scenario 7: Redistributing a product among existing product lines**. In this scenario, an organization wishes to determine the optimum division of product among the optimal number of product lines to minimize the cost of initial construction and maintenance for a given period of time. Let S_x the number of stand-alone products, n_j the number of products in the jth product line, npl the number of product lines, with $0 \leq npl \leq \left(S_1 + \sum n_j\right)$ and $nbr_{periods}$ the number of periods (in years). The equation to solve this scenario is defined below:

$$COST\left(npl, n_j, S_x\right) =$$

$$\sum_{t=1}^{nbr_{periods}} \left[\sum_{j=1}^{npl} \left(C_{org}(t) + C_{cabu}(t) + \sum_{i=1}^{n_j} \left(C_{unique}(p_i, t) + C_{reuse}(p_i, t) \right) \right) + \sum_{i=1}^{S_2} C_{prod}(t) \right]$$

(Equation 22)

- **Scenario 8: Adding new products to existing product lines**. In this scenario, an organization wishes to determine the optimal allocation of the k products over the existing product lines. The cost of adding a product in a product line PL (assuming this is done in a time period t) is:

$$C_{org}(t) + C_{cab}(t, PL, product) + C_{reuse}(t, PL, product) + C_{unique}(t, PL, product)$$ (Equation 23)

- **Scenario 9: Build vs. buy**. In this scenario, an organization wishes to determine the optimal split between building and buying additional assets for k products. This scenario can be solved by $C_{cab}(t)$, which returns the original costs of all assets. In the case where lease, rent or royalties must be made and t>1, $C_{cab}(t)$ returns the additional cost for that year payments. When assets are created in-house and t>1, $C_{cab}(t)$ returns the amount anticipated for maintenance.

3.3.8. Software Cost Estimation Model for Product Line Engineering (SoCoEMo-PLE)

The model created by Lamine et al. (Lamine et al., 2005) provides cost estimation for a software product line by using Poulin's (Poulin et al., 1993) and Mili et al. (Mili et al., 2000) models as start point. It assumes that an organization adopts a reuse program through four engineering cycles: component engineering cycle, domain engineering cycle, product engineering cycle, and corporate engineering cycle. All these cycles have cost functions based on the Reuse Cost Avoidance (RCA), Relative Cost of Reuse (RCR), Relative Cost of Writing for Reuse (RCWR), Service Cost Avoidance (SCA), and Additional Development Cost (ADC). It also addresses economic functions as the same way that Mili et al. in their integrated cost estimation model for reuse.

This model also establishes a process in order to run the model with the definition of four actor's roles:

- **Corporate engineer**. Decides to invest or not in a product line engineering approach;
- **Domain engineer**. Decides to invest or not in a domain engineering activity;
- **Application engineer**. Decides to adopt or not a product line engineering development approach in a specific project;
- **Components engineer**. Decides to develop or not a reusable asset to the product line to satisfy a set of specified requirements.

3.3.9. Quality-based SPL Cost Estimation Model (qCOPLIMO)

Boehm et al. (Boehm et al., 2006) describe a quality-based product line life cycle cost estimation model, namely qCOPLIMO, which is derived from previous authors models COQUALMO (Chulani et al., 1999) and COPLIMO (Boehm et al., 2004), both are extensions of COCOMO II (Boehm et al., 1995). Figure 3.6 presents the qCOPLIMO structure.

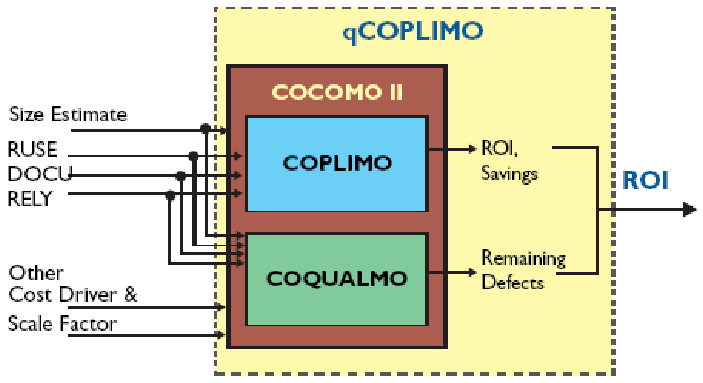


Figure 3.6 – qCOPLIMO structure (Boehm et al., 2006).

The main motivation for qCOPLIMO is the fact that the most significant product line cost models does not addresses software quality costs. This model consider how much is spent on removing undetected defects after a product release and it use the Poulin's metrics (Poulin, 1997b), $\boldsymbol{RCWR}$ and $\boldsymbol{RCR}$, to estimate quality-based software product line cost for developing N products, according to the following equation:

$$C_{PL}(N) = C_{RCWR} + (N-1) * C_{RCR} \quad \text{(Equation 26)}$$

3.4. Towards an Effective Software Product Line Cost Model

According to the common features of software product line cost models identified in the models of the previous sections, we are describing a set of these features that can be considered significant when creating a new cost model. Next, a brief description of these features is presented.

3.4.1. Costs and Benefits Functions

According to the work described by Clements et al. (Clements et. al., 2005), a cost model must allow users to create its own cost and benefits functions, allowing them to drive the model down to the level of detail for which sufficiently accurate data can be provided. This type of model provides some basic functions and new ones can be derived from the original costs and benefits functions.

Clements et al. also recommend that cost modelers can do their work with different levels of information. This vision of modeling is particularly dominant in models that have the focus on defining an integrated cost model for software reuse (Mili et al., 2001). In this approach, modelers can define viewpoints for a group of cost and benefits factors, e.g., *corporate viewpoint*, *domain engineering viewpoint*, *product engineering viewpoint*, and, *component engineering viewpoint*.

In addition, a cost model for software product line can be flexible as possible to allow modelers to "plug" different cost and benefits functions. For example, if an organization is trying to find out a cost model that fits its needs, they can consider trying on SIMPLE in a first round, and next trying on

COPLIMO for the same set of data. Next, the organization can configure the most suitable model for its use (Nóbrega et al., 2006).

Another point to consider when creating a model is the definition of different reuse scenarios for costs and benefits functions. It is particularly important in the model defined by Böckle et al. (Böckle et. al., 2004) which defines nine different scenarios in order to apply the basic costs functions. A flexible cost model for product line must define its main reuse scenarios and extends costs and benefits functions for calculating savings or losses for those scenarios.

Finally, the costs equations must express its values in a standard unit, such *persons-month*, in order to present the magnitude of saving or losses of reuse scenarios. Some basic cost models can perform this by using other units (e.g. lines of code), but we consider that effort estimation is more effective for decision-making tasks.

3.4.2. Reuse Scenarios

As cited previously, an effective cost model must define a set of reuse scenarios and extend the costs and benefits functions to provide savings or losses estimations. The dynamical aspects found in a specific organization justify the creation of different scenarios and the use of them to provide different costs and benefits viewpoints.

Using cost models is frequently associated with a technical audience due its mathematical and economics aspects. The use of predefined scenarios can break down the complexity of a cost model by allowing the creation of *wizards* for “running” the model. Each wizard can conduct a user in the task of analyzing the scenario costs, avoiding entering unnecessary data. To permit the application of reuse scenarios, a “default” scenario configuration is necessary for the entire organization. This scenario can configure the basic cost factors, such infrastructure and organizational needs.

3.4.3. Investment Analysis

According to the models described on the previous chapter, a set of common economic functions is available to estimate whether or not to invest in a product line initiative.

Thus, product line cost models does not have the focus on economic analysis, but if we are considering an economic point of view for an organization this aspect must be modeled. The output of an economic viewpoint highlights if is valuable to invest or not in one or more reuse scenarios within a product line. In this context, traditional economic functions can be used to perform cost-benefits balance, e.g. *Net Present Value* (NPV), *Return on Investment* (ROI), *Internal Rate of Return* (IRR), and so on.

A reuse process can define a cost-benefit task analysis as the phase immediately after the costs and benefits estimation for each scenario defined previously. This phase can make a sheet balance over the costs associated with different viewpoints. Some authors recommend that a decision analysis model can be incorporated into the cost model in order to evaluate heuristically the values returned by economic functions and decide which scenario is more suitable for an organization.

3.4.4. Approaches for Implementation

When we are considering an implementation of cost, benefits and economic functions, there are some ways to do that (Mili et al., 1999), (Clements et al., 2005).

The simplest case for cost function implementation is when the user estimates previously each cost. Some of data can come from historical information, such as the cost to build a stand-alone product.

Another approach is to get the data from the available community benchmarks, which can be used as a start point for organizations that have no product line development records. Cost models, such as COPLIMO (Boehm et al., 2004) or Poulin (Poulin, 1997b), defined cost drivers that reflect average values for those set of data. Boehm et al. estimates that to build a reusable component an organization will spent 150% of the cost for building it from the scratch.

Other implementation strategy is the creation of *utility functions* that will help the estimation of a cost over a number of consecutive time periods. Modelers can use a utility function to translate their assumptions about a scenario into a value. This is particularly interesting when we are considering that an asset has a value (or cost) and it may be incurred all at once or over time. To perform cost-benefits analysis in non-numeric values there is a technique defined by SEI (Kazman et al., 2002), called *Cost-Benefit Analysis Method* (CBAM) for performing costs analysis on architecture decisions and quantifying the benefits associated with them.

One point to consider in this study is a lack of some abstraction layers for the cost models. In some of them, it is possible to "plug" different costs, benefits and economics functions, in order to evaluate the most suitable for a specific organization.

Finally, one important point to consider is the "*divide and conquer*" nature of cost models implementation for product lines. Additional cost functions can be defined by decomposing the basic functions into other specialized functions. Peterson (Peterson, 2004) suggests that costs can be decomposed into factors. For example, for establishing architectural costs the model can take into account domain analysis costs, target architecture costs, technology standard costs, migration strategy, and so on.

3.5. Summary of the Study

Table 3.1 summarizes the set of features presented in this study. It makes a relationship between the most relevant cost models for software product line and the features discussed previously.

By analyzing Table 3.1 it is noted that just a few models have focus on investment analysis. It also noted that the models do not explore features such as pluggable functions and reuse scenarios. Only one model defines a decision analysis model to present different options of investment based on the variability of data entry. Some of the models studied cite the need to implement such decision models to help manager on decision-making tasks.

Table 3.1 – A Summary of Features of PL Cost Models

Cost Model	Features						
-	Cost Function	Benefit Function	Predefined Reuse Scenarios	Viewpoint	Pluggable Function	Investment Analysis	Decision Analysis Model
Poulin	X	X	-	-	-	-	-
ABC	X	X	-	-	-	-	-
Convergys	X	X	X	-	-	X	-
SIMPLE	X	X	X	-	X	-	-
COPLIMO	X	X	-	-	-	-	-
SoCoEMo-PLE	X	X	-	X	-	X	-
qCOPLIMO	X	X	-	-	-	-	-
Tomer	X	X	X	-	-	-	-
Schmid	X	X	-	-	-	X	X

We can state that, to be effective, a cost model for software product lines must have the flexibility to define new reuse scenarios in order to provide different cost visions (viewpoints). It is recommended that after the investment analysis task, some type of simulation can be done to decide for different scenarios or to mitigate the risks associated with them.

3.6. Chapter Summary

Applying a cost model for a reuse program is an effective way to analyze whether or not to invest in a product line approach. However, the models available fail to provide an integrated vision of costs, benefits and economics analysis for different viewpoints and scenarios.

This chapter surveyed the most significant cost models for software product lines and made a comparison among them, bringing to the light a set of features that defines an effective model.

Based on these features, the next chapter will presents a proposal for an integrated cost model for software product lines.

4 InCoME: Integrated Cost Model for Product Line Engineering

Based on the study conducted in Chapter 2, which describes the main features of a reuse cost model, and its specialization for software product lines, described in Chapter 3, this work proposes the **Integrated Cost Model for Product Line Engineering (InCoME)**, which will produce cost and benefits values in order to help an organization to decide if an investment in a product line is worthwhile. Accordingly, InCoME focuses on providing different return on investment visions for a set of reuse scenarios and it includes the possibility to simulate a range of input parameter values to evaluate the investment for those scenarios.

In addition, a proposal for using the model is discussed here and the requirements for its implementation are presented. The chapter highlights also the features of InCoME, its foundations and basic elements.

4.1. Introduction

There are several economic models dealing with software development costs for reuse (Lim, 1996), (Mili et al., 2000). However, a few models deal with Product Line Engineering (PLE) cost estimations. Moreover, most of the existing models do not have tools supporting them. Since the PLE appears to be a very attractive approach for software development with large-scale reuse, we are interested in economic models for PLE.

Managers, developers and others can estimate the intended PLE benefits through InCoME. The study performed on the previous chapters indicates that

Mili's *Integrated Cost Model for Reuse* (Mili et al., 2001) and the *Structured Intuitive Model for Product Line Economics (SIMPLE)* (Clements et al., 2005) could be used as a basis for InCoME. In addition, the model incorporates the Monte Carlo simulation technique (Malvin et al., 1986) in order to allow stakeholders to mitigate the risks for different visions of an investment. The foundations of the model are discussed on the next sections and all features used for its creation are detailed.

Before understanding InCoME it is interesting to take into account the main features that come up with a model. According to Hartmann et al. (Hartmann et al., 2006), a model can be defined as "*a theoretical construct that represents something, with a set of variables and a set of logical and quantitative relationships between them*". Models in that sense are constructed to enable reasoning within an idealized logical framework about these processes and are an important element of scientific theories. A model may make explicit assumptions that are known to be false (or incomplete) in some details. Such assumptions may be justified on the grounds that they simplify the model, allowing the production of acceptably accurate solutions.

Several economic models for software reuse have been developed and applied to evaluate various aspects of software development projects. These have been created mainly for the purposes of either facilitating more accurate software project planning, supporting managers in making decisions about reuse scenarios or predicting the effects of processes changes. Each of these models accomplishes their purpose by estimating overall net measurements of the process, such as development time, cost and quality. The obvious relevancy of this domain to our research lies in our intended adoption of two of these models as a basis upon which to create InCoME.

Based on the cases studies presented by the cost models from the previous chapters, InCoME has a clear focus on providing different viewpoints for costs and benefits for a product line, encapsulating these factors into *reuse scenarios*. Reuse scenarios will be the basis for InCoME investment analysis, which will evaluate if a specific scenario is valuable from an economic point of view.

4.2. Overview of the Model

As defined in Chapter 2, a reuse cost model is the notation of the explicit costs and benefits associated with a reuse program (Rothenberger et al., 2004) and managers in their decision-making tasks can use it through an economic rationale. When we are considering the reuse adoption through product line engineering, the cost model associated with it can define reuse scenarios (Clements et al., 2005) in order to provide different costs and benefits viewpoints (Mili et al., 2001).

To be effective, a cost model must answer the questions regarding the managers understanding on current and future product-line-related parameters into cost and return on investment (*ROI*) expectations associated with a product line approach (Muthig et al., 2006). It also allows the organization instantly gets feedback on potential product line effects and improvements.

A summary of the model defined in this chapter can be viewed on Figure 4.1, which denotes the structure of InCoME in a block diagram notation. It is composed by the following elements: a ***Cost Factors Layer***, which encapsulates a set of ***Cost Functions***; a ***Viewpoint Layer*** composed for three ***Viewpoints***, each of them encapsulating a set of ***Reuse Scenarios***; an ***Investment Analysis Layer*** with a set of ***Economic Functions***; and, a ***Simulation Layer***.

Each layer defined for InCoME plays a specific role during the evaluation of an investment for a product line. The idea behind its definition in form of layers is to modularize the evaluation through three levels of estimations:

- **Cost Estimation**. It addresses the estimation of the basic cost factors that are related with a product line.
- **Benefit Estimation**. It is responsible to produce estimations concerning one or more reuse scenarios. It will be expressed in terms of cost savings or losses in adopting such scenarios.
- **Economic Estimation**. It reflects the estimation over an investment cycle for the cost savings found in the benefits estimations. It produces values to help the decision concerning economics aspects of a product line.

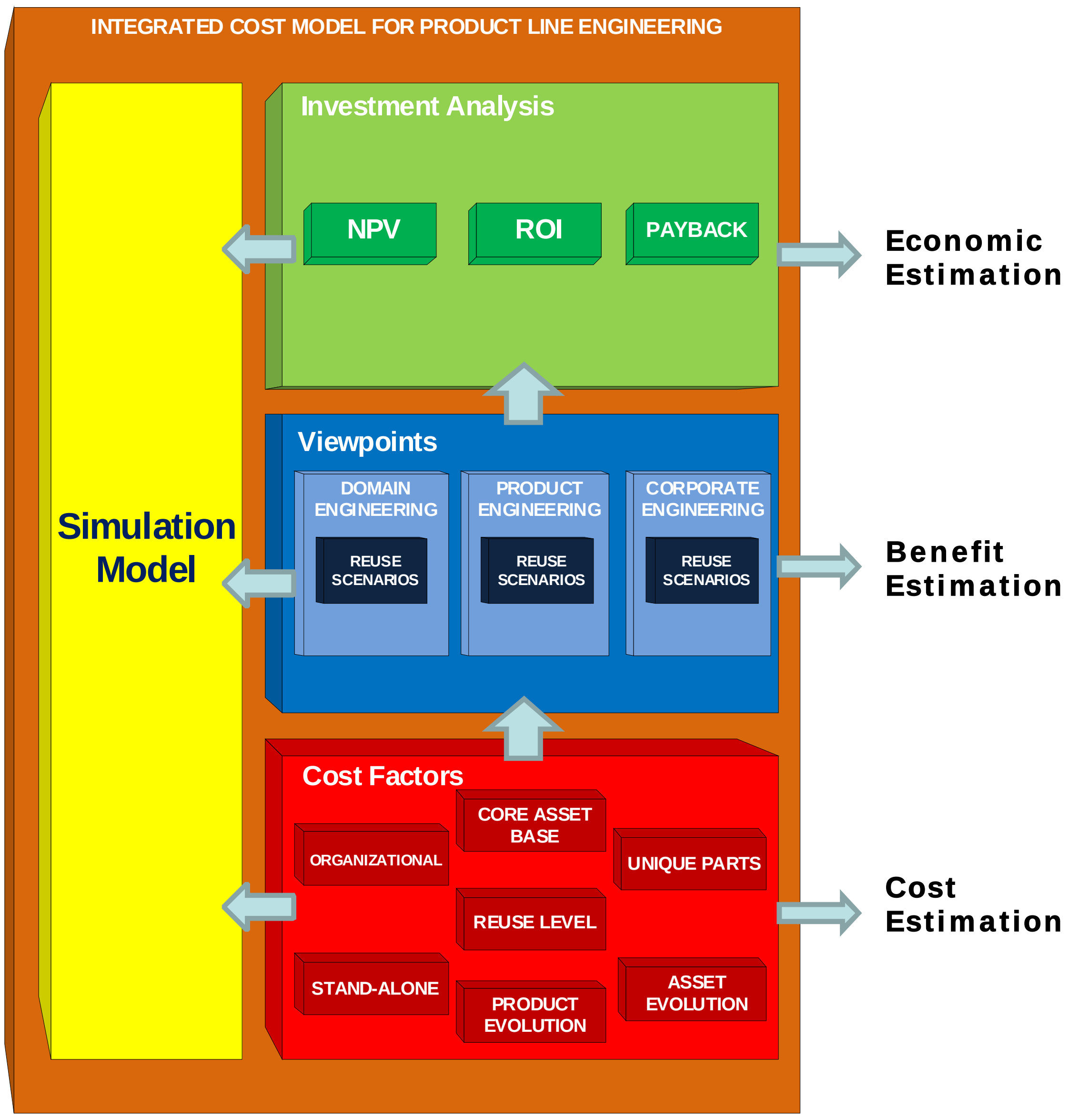


Figure 4.1 – Integrated Cost Model for Product Line Engineering (InCoME)

Figure 4.2 presents InCoME elements through a meta-model denoted by using UML. In this figure, the relationships between the elements can be viewed and it can be considered as a basis for the model instantiation[3].

[3] All attributes and methods are not described in this diagram due to make it as simple as possible.

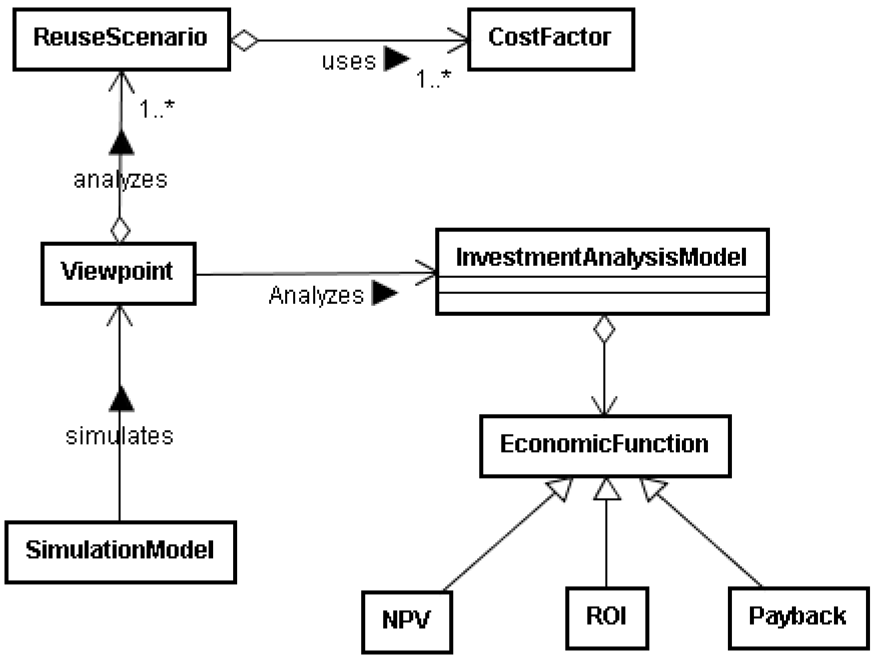


Figure 4.2 – InCoME Meta-Model

The *Cost Factors Layer* is the lowest level of the model and it is responsible to feed upper layers with cost estimations computed by its *cost functions*. The cost functions address the cost estimation by seven product line engineering factors:

- **Organizational.** The cost related with upfront investments to establish a product line infrastructure;
- **Core Asset Base.** The cost to build a set of reusable assets for a specific domain;
- **Unique Parts.** The cost associated with the development of unique parts of software for a product within a product line;
- **Reuse Level.** The level of reuse related with the integration of reusable assets into a product;
- **Stand-Alone.** The cost to build a product in a stand-alone fashion, i.e. outside a product line regime;
- **Product Evolution.** The cost to evolve a product in a stand-alone fashion;
- **Asset Evolution.** The cost to evolve the core asset base through product line engineering.

The *Viewpoint Layer* is fed by the values calculated by the *Cost Factors* level. This level uses those values to calculate the functions for each *Reuse Scenario*, which expresses the cost savings (or losses) vision of a specific type of stakeholder. InCoME applies viewpoints for three product line engineering cycles: *domain engineering cycle*, *product engineering cycle* and *corporate engineering cycle*. We are assuming that this set of viewpoints is adequate for representing the different reuse visions for product line engineering. All these viewpoints are detailed in next sections.

Next, the values produced by each scenario are classified according to its viewpoints and they are submitted to the *Investment Analysis Layer*, which is responsible to make several economic calculations for each viewpoint. InCoME can make investment computations for three *economic functions*: *Net Present Value (NPV)*, *Return on Investment (ROI)* and *Payback Value (PV)*.

Finally, the economic values computed for each viewpoint can be simulated by the *Simulation Model*, which applies Monte Carlo simulation technique (Malvin et al., 1986) to achieve certain *NPV*, *ROI* and *PV* predictions for ranges of values through product line engineering cost factors.

In the next sections will be presented the objectives of the model and its elements are detailed.

4.2.1. Objectives

According to the structure presented in the Figure 4.1, each element plays a specific role when the model is used. At general, InCoME was defined under the need to answer the following questions:

Q1. Does the model express the different viewpoints of reuse for stakeholders of an organization?

Q2. Does the model perform an investment analysis over the organization viewpoints?

Q3. Does the model can simulate reuse scenarios through its cost parameters in order to help the decision-making tasks?

Based on the expected answers for these questions, the main objective of the Integrated Cost Model for Product Line Engineering (InCoME) can be stated as follows:

"InCoME is defined to perform reuse investment analysis from different viewpoints of an organization that are adopting (or has an intention to adopt) the software product line approach in order to **help stakeholders in their decision-making tasks**"

4.2.2. Model Assumptions

When we are considering the definition of InCoME, some assumptions must be taken into consideration in order to apply it:

- **Reuse Process**. Despite InCoME has been defined under the umbrella of the RiSE process (Almeida, 2007), it must be considered a process-independent model, since the set of activities defined for a process that uses InCoME should not have a direct impact on cost factors estimation. However, it makes no sense to apply it in a context different of a product line, since its essence is driven by a product-based approach;
- **Product Family**. Assuming the model has a product-based approach, we can state that the environment where it is to be applied has an existing family of products sharing some common features between them. Moreover, the model is not affected by the way the products were built: independently in a stand-alone fashion or over a product line engineering regime;
- **Reuse Cycles**. Product line frameworks, such as the defined by SEI (Clements, 2002) have a well-defined group of activities where reuse happens. As presented in Chapter 3, SEI framework defines three fundamental sets of practices (domain engineering, product engineering and management). Despite InCoME be an independent-process model, we are assuming that an organization using it has at least its reuse activities performed by three cycles (Mili et al., 2001): **(i)** Domain Engineering cycle, **(ii)** Product Engineering cycle and **(iii)** Corporate Engineering cycle. These cycles are explained below:

- **Domain Engineering Cycle.** This cycle addresses all activities related with the analysis of commonalities and variability of core assets, construction, certification, and repository insertion. Thus, all activities related with domain engineering have its costs estimated during this cycle;
- **Product Engineering Cycle.** In this cycle all activities to develop a product occur, including the development of the unique part of the product;
- **Corporate Cycle.** It is related with establishment of the product line infrastructure and the issues associated with the reuse process adopted by the organization.

This assumption is particularly important when we define the different investment viewpoints for the stakeholders, as we can see later in this chapter.

4.3. The Foundations

The foundations of the model reflect the basis in which it was defined, and they can be considered as a guideline to describe the elements of InCoME. This model was written using the assumptions and definitions of two fundamental models for estimating and predicting costs and benefits for software reuse in general. Moreover, since InCoME defines a simulation model, a popular technique to generate random input values for the cost factors was used. The foundations used as guideline for InCoME are:

- **Integrated Cost Model for Software Reuse** (Mili et al., 2001). This model was chosen to be the basis for the viewpoints, the economic functions and the investment analysis foundations;
- **Structured Intuitive Model for Product Line Economics (SIMPLE)** (Clements et al., 2005). This model details the cost and benefits functions and it has the definition of nine reuse scenarios for a product line. InCoME uses a reduced subset of SIMPLE scenarios, with some extensions;
- **Monte Carlo Simulation** (Malvin et al., 1986). This technique addresses uncertainty for the model and it was chosen to be the basis to

generate a range of random input parameters according to a set of well-defined probability distributions.

4.3.1 Integrated Cost Model for Software Reuse

The work performed by Mili et al. (Mili et al., 2001) is an extensive study of the state-of-the-art for software reuse cost models. The same type of study has already been done by Wiles (Wiles, 1999) where the main features of a generic cost model were presented, but no model was explicitly defined in such work. Lim (Lim, 1996) also had aggregated the main aspects of a cost model into a group of general features, and at the end of his work he concludes with recommendations of how to select an economic model.

The main differential of the work presented by Mili et al. is the proposal of a more comprehensive model, characterized by a set of generic features. On the other hand, the model does not address the economic features for product line engineering and its costs and benefits functions are achieved by traditional ***component-based development*** approach. Those features were extensively studied by revisiting a large number of existing cost models for software reuse, extracting from them the following general features, used by InCoME:

- **Investment Cycles**. The model defines four cycles that can be used to provide different corporate decisions: *Corporate Investment* cycle, *Domain Engineering* cycle, *Application Engineering* cycle, and, *Component Engineering* cycle. All these cycles can estimate the *Return on Investment (ROI)* for an organization that is adopting a reuse program. InCoME uses this approach in order to define its *viewpoints* for the same set of investment cycles, except the component engineering cycle;
- **Cost Factors**. They can be considered as a start point for costs and benefits estimations within investment analysis, representing the organization defaults for such cost factors. InCoME encapsulated its cost factors in the *Investment Analysis Layer* by reusing the concepts of *Investment Cycle*, *Start Date* and *Discount Value*. The other cost factors for InCoME are supplied by the model defined by Clements et al.

(Clements et al., 2005) which are more specific to a product line engineering approach. They will be presented later in this chapter;

- **Economic Functions**. The functions used to assess the worthiness of an investment decision. The functions used by InCoME come from the Mili's model are: *Net Present Value*, *Return on Investment*, and *Payback Value*;
- **Viewpoints**. Based on the investment cycles, the viewpoints defined by Mili et al. reflect the different economic visions that can exist within an organization. They are used for different stakeholders by using distinct return on investment equations for its specific areas of actuation. We agree with this concept and InCoME was influenced to provide a similar set of corporate visions with some extensions. The difference of InCoME viewpoints from the Mili et al. viewpoints relies on the application of costs and benefits functions based on a software product line environment, which is *product-based*, in opposition to a *component-based* approach. It helps the model to reflect the real nature of a product line, with the benefit of the viewpoints.

Mili et al. model defines another set of features that characterizes a generic cost model for software reuse, such as the type of reuse organizations and a list of hypotheses for cost and benefits estimation. In this dissertation, it is our intention to derive a specific cost model for a product line context and it implies in not using those aspects.

When defining InCoME we are concerned in its practical utilization for the managers within an organization. The model defined by Mili et al. has the same concerns about this and it describes a framework in order to propagate the costs for one investment cycle to a subsequent cycle. For example, a corporate manager would assess the impact of the reuse program by taking into consideration the costs of the reuse infrastructure as well as domain engineering costs, which are accumulated across different domains. He or she could also balance these costs against the benefits measured from quality and productivity gains achieved in application engineering cycle (which are accumulated across the development projects). Moreover, a manager of the organization that are producing the core asset base would assess the impact of a domain engineering effort by considering the cost of producing that assets against the benefits reaped from *selling* them to project managers.

The process to use InCoME is influenced by Mili's model in the following terms:

- All investment cycle decisions can be quantified in economic terms and they are justified by an economic rationale;
- and, All investment cycle decisions provide different viewpoints for costs and benefits and they are interconnected (a change in a cost factor in one viewpoint has a direct impact in other viewpoints).

4.3.2 Structured Intuitive Model for Product Line Economics (SIMPLE)

The model defined by Clements et al. takes into account the emerging adoption of product line engineering by organizations. SIMPLE are related with the following software product line practices defined by the Software Engineering Institute (Clements et al., 2004): architecture evaluation, data collection, metrics tracking, Make/Buy/Mine/Commission analysis (Clements et al., 2001), scooping, technical planning, technical risk management, tool support, business case building, acquisition strategy development, funding, launching and Institutionalization, market analysis, and technology forecasting.

Even where costs and ROI calculations are not an essential part of these practices, SIMPLE provides to decision makers a way to evaluate different economic visions for a product line. The cost models available in the literature lack on handling economic aspects for product line engineering. As presented in Chapter 3, there are a few models concerned with this approach (compared with the number of models addressing basic reuse cost models).

The other relevant models studied in this dissertation are the Poulin's *Measuring Software Reuse* (Poulin, 1997b) and *COPLIMO* (Boehm et al., 2004). Despite Poulin's model considers the use of assets in developing individual products and the potential for cost savings, it do not take into account the dynamic scenarios of product lines and its implications. As with Poulin, COPLIMO is essentially a reuse cost model, assuming the use of a set of assets for building a set of related products. COPLIMO goes further than Poulin by considering variations in the cost of reuse and in considering maintenance, but it makes some simplified assumptions as well. On the other hand, COPLIMO relies on the availability of a range of parametric values that must be accurately calibrated, making it less suitable for a non-technical audience.

The main advantage of SIMPLE is its *"divide and conquer"* strategy, which means to define costs and benefits functions into other functions by decomposition.

In this sense, we agree that SIMPLE, as your name promises, is a simple, structured and intuitive model, allowing a wide range of people inside an organization to view the level of information that is most useful for their jobs. In the context of this dissertation, we are electing the following SIMPLE features for InCoME definition:

- **Cost Functions**. SIMPLE presents seven cost functions that reflect the costs of a product line. InCoME uses this set of cost functions as a basis for the entire model.
- **Reuse Scenarios**. According to the cost functions, SIMPLE provides nine reuse scenarios in order to estimate the costs and benefits for several situations that may occur in a product line.

When defining InCoME we are concerned on how the reuse scenarios can provide different viewpoints for costs and benefits for a product line, in order to help managers in decision-making tasks. One issue to consider in InCoME definition is the mapping between the set of reuse scenarios (derived by SIMPLE) and the viewpoints (derived by Mili's model), which will be the basis for the investment analysis. A solution for this is discussed in the section where the elements of the model are defined.

4.3.3 Monte Carlo Simulation

The idea behind Monte Carlo simulations gained its name and its first major use in 1944 (Pllana, 2000), in the research work to develop the first atomic bomb. The scientists working on the Manhattan project (Manhattan, 2004) had intractably difficult equations to solve in order to calculate the probability with which a neutron from one fissioning Uranium atom would cause another to fission. The equations were complicated because they had to address the complicated geometry of the actual bomb, and the answer had to be right because, if the first test failed, it would be months before there was enough Uranium for another attempt. They solved the problem with the realization that they could follow the trajectories of individual neutrons, one at a time, using teams of humans implementing the calculation with mechanical calculators (Feynman, 1985), (Manhattan, 2004). At each step, they could compute the probabilities that a neutron was absorbed, that it escaped from the bomb, or it started another fission reaction. They would pick random numbers, and, with the appropriate probabilities at each step, stop their simulated neutron or start new chains from the fission reaction.

In the context of InCoME definition, the Monte Carlo simulation was chosen as the way to manage uncertainty that can occur in the input values of the model. At last instance, this technique allows the model to improve the risks mitigation when analyzing product line engineering adoption. A more formal definition of Monte Carlo simulation method is presented in Appendix A.

4.4. Elements of the Model

As cited in the previous sections, InCoME is composed by a group of elements, which each of them has specific responsibilities. In the next sections we describe the details of those model elements.

4.4.1. Cost Factors

4.4.1.1. Demand Function

In order to define the cost and benefit functions for product line engineering, it is necessary to establish the lower level of granularity for cost factors. We can accomplish this by the definition of a ***demand function***. Let first define a product line in terms of its products.

Let P a product line with n software products. Each product is denoted as p_k, where k = {1, 2, 3, ..., n}.

Next, it is necessary to define p_k in terms of its assets and requirements. According to Clements et al. (Clements et al., 2005), in a product line every asset is managed as a core asset and it is available for using within a product development activity. So, let assume that p_k is composed by two distinct sets of assets: **Unique Assets** and **Shared Assets**.

- **Unique assets**. This set of assets is related with both software and non-software of a product that are not based on the core asset base.
- **Shared assets**. This is a subset of the core asset base and it addresses the shared functionality needed for p_k.

Let now assume that the unique set of *i* assets for p_k, namely U_k, is expressed as follows:

$$U_k = \{u_1, u_2, u_3, \ldots, u_i\} \text{ (Equation 27)}$$

Next, we make the same assumption for the set of *m* shared assets that has been used by p_k. In this point it is necessary to define the set of all *j* assets of *P*, denoted by A_P, as follows:

$$A_P = \{a_1, a_2, a_3, \ldots, a_j\} \text{ (Equation 28)}$$

The set of shared assets required for p_k development, denoted by C_k, is defined by $C_k = \{c_1, c_2, c_{3,} \ldots, c_m\}$ (Equation 29), where $C_k \subset A_P$, and the intersection of the *j* elements of A_P with the *m* elements of C_k has the same functionality[4].

Then, the total set of assets for p_k, denoted A_k, is defined by the following expression:

$$A_k = U_k \cup C_k \text{ (Equation 30)}$$

Now, we assume that each set of assets A_k for p_k has an association with a group of requirements, namely R_k. In order to estimate the size of A_k, we need a unit that may represent the amount of functionality for R_k. One possible standard unit to perform that representation is *Function Points* (Albrecht, 1979). It measures the size of functionality that is encapsulated into a requirement and can be used for the purpose of this dissertation. In addition, we are defining a ***mapping function*** M_k^A that returns the size, in function points, of the set of assets A_k that has its functionality represented by R_k:

$$M_k^A = M(R_k) \text{ (Equation 31)}$$

[4] We can translate the term *functionality* as the set of requirements needed to build the asset.

Next, we extend the previous mapping function by the definition of a ***Demand Function***, namely $D(A_i)$, which returns the size per time unit necessary to develop the asset A_i, as follows:

$$D(A_i) = M(A_i) * \left(\frac{1}{T_p}\right) \text{ (Equation 32)}$$

where T_p is a planning period at which the asset is going to be analyzed. As a consequence, **using equations 30 and 32**, we can infer the demand function for A_k using the following equation:

$$D(A_k) = D(U_k) + D(C_k) \text{ (Equation 33)}$$

Replacing the terms of **equation 33 by equation 32**, we have the demand function expressed as follows:

$$D(A_k) = \frac{M(U_k) + M(C_k)}{T_p} \text{ (Equation 34)}$$

The equation 34 cannot be used as a function to estimate the cost to develop a product *p_k*. It is only estimating the amount of functionality over a period of time to develop its set of assets, but it does not take into account the costs for development *with* reuse. The cost model defined by Poulin et al. (Poulin, 1993) has been addressing this subject, distinguishing between the development *for reuse* and development *with reuse*.

Therefore, we agree with the vision of Mili et al. (Mili et al., 2001) that states that the cost of a reusable component is defined by the cost to develop it for reuse plus the cost to certify it plus the cost to insert it into a component library (or into a component repository). In this sense, it is reasonable to say that the cost to build a core asset denoted by $C(A_i)$ can be expressed by the following equation

$$C(A_i) = D(A_i) + CERT(A_i) + INS(A_i) \text{ (Equation 35)}$$

where $CERT(A_i)$ is the demand function for A_i certification and $INS(A_i)$ is the demand function for library insertion for A_i. This approach does not take into account the time influence on the asset cost, since the costs of certification and library insertion could be diluted over time. This issue is addressed by reuse scenarios equations, which takes into consideration the effect of time in the cost savings. Despite Mili et al. original equations deals with the effect of operation and maintenance of a reusable component when defining its *episodic costs*, we are adopting the vision of SIMPLE that considers those costs in the level of domain engineering cycle.

One point to consider here is that the demand function for an asset (or for the entire core asset base) will be considered as a **transfer price** (Lim, 1998) for the subsequent engineering cycles.

4.4.1.2. Cost Functions

The first step on the cost functions definition for InCoME was given in the previous section when we defined the lowest level of costs, specifically to the component-level costs. With these cost equations, we can derive another set of functions for the remaining cost factors.

- **Organizational**. In the corporate level, an organization wants to estimate the costs of adopting product line engineering. We can state that the cost to establish a product line is related with the costs of product engineering and domain engineering. Moreover, we must consider that an organization accounts for overhead when preparing itself to product line establishment. SIMPLE (Clements et. al., 2005) defines a set of cost drivers which contributes to the organizational costs, namely C_{org}:
    - Internal reorganization costs;
    - Process improvement;
    - Training;
    - Other organizational remedies costs.

  Mili et al. model (Mili et al., 2001) defines some additional cost factors that influence the organizational costs:
    - Purchase and installation of a repository (assuming the repository costs will be divided equally between all projects that shares the core asset base);
    - Operational costs to manage the product line infrastructure.
- **Core Asset Base**. This cost function returns how much development effort is expected for a core asset base for a specific domain. Both models (Mili et al. and SIMPLE) consider this cost is influenced by the effort to perform domain analysis and design for reuse. Let $DA(\delta)$ a function that

returns the costs for domain analysis for a specific domain δ, which can be measured by the following factors:

- Commonality and variability analysis effort;
- Product line scope definition;
- Design and evaluation of a generic software architecture;
- Build of the Production Plan;
- Establishment of the development environment;
- Build of a testing architecture;
- Other artifacts development effort related with core asset building (except code).

SIMPLE defines a cost function, C_{cab}, to express the costs to develop a core asset base to a particular scope within a domain. Assuming A_δ as the set of assets of δ, InCoME calculates this factor using the following equation:

$$C_{cab}(\delta) = DA(\delta) + \sum_{\forall A_i \in A_\delta} C(A_i) \text{(Equation 36)}$$

where $C(A_i)$ is the cost to develop the asset A_i, and A_i belongs to the set of assets of δ.

- **Unique Parts**. This factor, denoted C_{unique}, represents the cost to develop the unique parts of the product that are not based on assets in the core base asset. This cost factor can be estimated by the demand function for the unique set of assets for a product P_k (defined by equation 27) and calculated using Equation 31, as follows:

$$C_{unique}(p_k) = D(U_k) \text{ (Equation 37)}$$

- **Reuse Level**. This factor, denoted C_{reuse}, represents the cost to develop a product reusing core assets from a core asset base. C_{reuse} can be estimated by the effort of using the core asset base for p_k development. These cost factors includes:
  - Cost of locating a core asset;
  - Cost of checking out a core asset of a repository;
  - Cost of tailoring;
  - Cost to perform extra tests.

Frequently, this cost factor is expressed as a fraction of the effort to build a product out of the product line umbrella. Some case studies had indicated that this factor is about 7% of the cost of building the product from scratch (Böckle et al., 2004).

- **Stand-Alone**. This cost factor represents the cost of development for a product in a stand-alone fashion, denoted C_{prod}. In this way, each product is built from scratch, with no reuse of core assets. In general, C_{prod} can be estimated using traditional cost models (e.g. COCOMO, Function Points Analysis, and so on) or it relies on historical data from an organization. So, the cost of building *n* products independently, C_{ind}, without sharing the core asset base is expressed by the following equation:

$$C_{ind} = \sum_{k=1}^{n} C_{prod}(p_k) \text{(Equation 38)}$$

- **Product Evolution**. This cost function returns the cost to evolve a product p_k in a stand-alone fashion and C_{evo} denotes it. This function is normally expressed as a percentage value of the cost of building a product in stand-alone fashion. According to Böckle et al. (Böckle et al., 2004), the cost to evolve a product p_k is roughly estimated in 20% of the cost to build it from scratch.
- **Asset Evolution**. As the same way as C_{evo}, this cost factor namely C_{cabu}, can be expressed as percentage value of the cost to build a core asset base from scratch. This factor is influenced by new version required for an asset (new commonalities exposed) and bugs fixes in existing core assets. Böckle et al. (Böckle et al., 2004) estimates this value as 10% of C_{cab}.

4.4.2. Viewpoints

The economic benefits of software reuse have long been recognized. In general, they can be divided into two major categories (Favaro et al., 1998):

- **Operational Benefits**. They consider factors such as improved quality, higher productivity, and reduced maintenance costs;
- **Strategic Benefits**. Includes the opportunity to enter in new markets and the flexibility to respond to competitive forces or changing market conditions.

This section defines a set of benefit functions for product line engineering that can be categorized as operational benefits. It is known for reuse community that strategic benefit functions are difficult to develop due to their "intangible" nature. It is out of the scope of InCoME to define such functions. The economic analysis generated by operational benefits estimation can be considered as a basis for strategic benefits functions (Brealey et al., 1996).

As cited previously, InCoME has as assumption a set of engineering cycles where reuse happens. In order to distinguish between the different investment cycles, we are defining a set of viewpoints for product line engineering. Each viewpoint has a set of cost and benefits functions that will reflect if an investment is worthwhile on it. Each set of cost and benefits functions for a given viewpoint is called a *reuse scenario*.

The main objective to define viewpoints is the possibility of decision-making according to the point of view from different stakeholders (Mili et al., 2001). The model do not addresses a mapping of a specific set of stakeholders with the viewpoints, but we can assume that the visions of investment for a product line can be presented according to the Table 4.1 below. The column *Viewpoint* presents the investment vision and the column *Stakeholders* shows a possible audience for that vision.

Table 4.1 – Viewpoints and Stakeholders

Viewpoint	Stakeholders
Domain Engineering	Domain Engineer, Core Asset Engineer
Product Engineering	Product Engineer, Project Manager
Corporate Engineering	Project Manager, Financial Manager

The strategy to define the reuse scenarios is basically to map the vision of investment of each viewpoint to one or more cost factors, which encapsulates the cost estimations for that viewpoint. Next, we present the InCoME viewpoints and the reuse scenarios adapted for them.

4.4.2.1. Domain Engineering Viewpoint.

This viewpoint describes whether or not to initiate a domain engineering effort in a specified domain. It will be achieved by measuring the **total benefits for the set of reusable assets** that compose a **domain**.

Reuse Scenario. Our target here is to evaluate if the building of a set of assets can saves costs to the domain engineering cycle. The strategy to perform it is to assume that the benefits for this cycle are the sum of the benefits of all assets built for a specific domain less the cost to build the core assets for the same domain. The benefits in developing reusable assets can be quantified in terms of how fast a product can be build reusing these assets, which means gains in terms of productivity, and the value-added in quality for the core asset base (Mili et al., 2001). It is also known that there is a linear relation between the gains in productivity and quality and the number of times an asset is reused in many products in a product line (Mili, 1996). In this sense, the benefit function for core asset base can be expressed by the frequency of use of reusable assets for a period of time. We are assuming here that the *frequency of reuse* for an asset has a direct impact in productivity and maintenance (and hence, it has impact in quality) for the entire product line. Formalizing this concept let A_i a core asset and $B(A_i)$ the following benefit equation:

$$B(A_i, t) = freq(A_i, t) * D(A_i) \text{ (Equation 39)}$$

where $freq(A_i)$ is the frequency of use of A_i in a period of time *t*, and $D(A_i)$ is the demand function for A_i. To calculate the cost savings or losses it is important to take into account the cost to build the core asset base. According to the function defined by Equation 39, we can extend it for the set of assets of a product line, denoted by A_δ, which was built for a specific domain δ. Next, we derive the reuse scenario for domain engineering viewpoint, denoted $B(A_\delta)$, as the sum of the benefits for all assets built as part of domain, within a period of interest *t*, less the cost for building the core asset base for the same domain:

$$\forall A_i \in A_\delta, B(A_\delta, t) = \sum B(A_i, t) - C_{cab}(\delta, t) \text{ (Equation 40)}$$

4.4.2.2. Product Engineering Viewpoint.

The investment decision that appears in this viewpoint reflects if whether or not to invest in product line engineering for a given development project. This can be achieved by measuring the **cost savings** to build products using a **product line approach** versus the **cost of build products independently**.

Reuse Scenario. Here the idea is to find out if there is a cost savings or losses when applying a product line approach in order to develop a product. We can use the set of equations defined previously to calculate the cost to build a product sharing the commonalities of the core asset base, within a product line. In this point, we have to take into account a period of interest where an event can occur. This is particularly important under reactive product line development (Clements, 2002), (Clements et al., 2004), which focuses just-in-time core asset development, as opposed to proactive approach in which the entire core asset base is built up front.

If we consider the development of a product p_k for a domain δ, where p_k is composed by the set of assets A_k, with $A_k \in \delta$, then the cost of p_k, namely $C(p_k)$, can be estimated by the sum of the effort to develop the unique parts of the product (C_{unique}) and the reuse effort to integrate A_k to the product (C_{reuse}), as follows:

$$C(p_k) = C_{unique}(p_k) + C_{reuse}(p_k) \text{ (Equation 41)}$$

Assuming a period of interest t to develop a product p_k, the reuse scenario for product engineering viewpoint, $B(p_k)$ is the difference between the cost to build p_k independently and the cost to build it from a product line infrastructure. The first term is expressed simply by C_{prod}. To estimate the cost of the second term we have to take into account the cost of establishing a product line infrastructure to build p_k. Moreover, the cost to build the core asset base for p_k is also taken into consideration[5] as well as the cost to build p_k itself. So, the reuse scenario for this viewpoint is expresses as follows:

$$B(p_k, t) = C_{prod}(p_k, t) - \left[C_{org}(p_k, t) + C_{cab}(A_k, t) + C(p_k, t)\right] \text{ (Equation 42)}$$

4.4.2.3. Corporate Engineering Viewpoint.

This viewpoint addresses if whether or not to initiate a corporate reuse program thought the adoption of product line engineering approach. It will be achieved

[5] We are assuming the development of the first generation of a product line, so the core asset base is simply build for p_k development.

by measuring the **total benefits for all products within a product line** instantiated to an organization.

Reuse Scenario 1. In this viewpoint, we are interested in estimating all benefits derived from product line engineering adoption within an organization. It implies in taking into account all benefits measured on the product engineering cycle and calculate the potential cost savings or losses for a set of reuse scenarios. InCoME scenarios were clearly influenced by SIMPLE scenarios (Clements et al., 2005) and some of them were used as a guideline to describe the viewpoint for corporate cycle. Again, the strategy here is a mapping between a viewpoint and the cost factors in order to produce a set of reuse scenarios. The first scenario returns the cost savings or losses for a product line adoption. Suppose that an organization wishes to choose between building a set of products as a software product line and building them as a set of stand-alone products that do not share the core asset base. Then, the first scenario function stands for the cost to build *n* products independently less the cost to build the same set of products using a product line. Using *t* as a time period of interest, and equation 42 extended for *n* products, we have the following benefit equation for the product line *P*:

$$B_P(t) = \sum_{k=1}^{n} C_{prod}(p_k, t) - \left[C_{org}(t) + C_{cab}(t) + \sum_{k=1}^{n} \left(C_{unique}(P_k, t) + C_{reuse}(P_k, t) \right) \right]$$

(Equation 43)

In opposition to equation 42, this function captures the total organizational costs as well as the total core asset base costs.

Reuse Scenario 2. Next, we can suppose that an organization wishes to know the benefits in setting up and evolve a software product line. In this case, the reuse scenario addresses the difference between the evolution of *n* products in a stand-alone fashion (C_{evo}) and the costs to evolve the same set of product through a product line regime. The second term of the equation takes into consideration the fact that all organizational costs for a product line were already incurred and only a portion of the costs of build a core asset base were accounted (C_{cabu}). In addition, the costs to update the unique parts of the product are merely a fraction of the original part. So, assuming F_{unique} as the factor to express the percentage of C_{unique} update the equation is expressed as follows:

$$B_P^{'}(t) = \sum_{k=1}^{n} C_{evo}(p_k,t) - \left[\sum_{k=1}^{n} \left(C_{cabu}(p_k,t) + \mathrm{F}_{\mathrm{unique}} * \mathrm{C}_{unique}(p_k,t) + C_{reuse}(p_k,t) \right) \right]$$

(Equation 44)

One point of discussion here is that a product might have um specific number of releases in a period of interest. Despite SIMPLE does not directly take into account this factor (Clements et al., 2005) we are considering it for this scenario. Assuming a number of releases nr for a period of interest t, we can rewrite equation 44 to express the cost savings or losses simply by multiplying nr and $B_P^{'}(t)$.

Another point to discuss is the fact that the benefits accrued from domain engineering cycle were not accounted for corporate engineering viewpoint. According to Mili et al. (Mili et al., 2001), the domain engineering benefits are considered ***values of asset sales to projects*** and in the corporate level this cost transfer is an internal operation, which does not represent a gain or a loss. Accordingly, the benefits for corporate viewpoint are placed only by the potential benefits of product engineering cycle.

4.4.3. Investment Analysis

The investment analysis layer of InCoME has as main objective to evaluate if the benefits accrued across a product line engineering cycle are valuable in an economic point of view. Looking at the literature on engineering economics (Brealey et al., 1996), (Trigeorgis, 1996), (Favaro et al., 1998), (Lim, 1998), it is possible to identify a set of economic functions that can be used to evaluate the worthiness of an investment decision.

In order to perform the investment analysis, the organization must define a set of economic parameters that will reflect the corporate strategy, which is based on an economic rationale. According to Favaro et al. (Favaro, 1998) those parameters can be summarized by the following factors:

- **Investment Cycle**. Denoted by Y, it describes the time period at which the investments will be analyzed. This value is often expressed in years and it is counted from a *Start Date*, denoted by SD.

- **Start Date**. Denoted by *SD*, it describes the date at which the investment cycle starts and the initial costs for the product line are incurred.
- **Discount Rate**. Denoted by *d*, it is an abstract quantity that reflects the time value of money.

Next, a set of economic functions will take these factors in account to perform the calculations.

Net Present Value (NPV). In the context of this dissertation, we are interested in calculate the ***present value*** for a given investment to develop *n* products for a product line. The concept of present value is an essential tool for giving proper weight to all present and future costs and benefits resulting from an investment. Based upon the simple notion that a monetary unit today is worth more than the same unit tomorrow, known as the ***"time value of money"***, engineering economic field define the **Discounted Cash Flow (DCF)**, which "weights" the relative contributions of cash flows that are more or less distant in the future with the application of a discount rate *d* according to the period (e.g. a year) in which the cash flow C_i occur, as follows:

$$PV = \left(\frac{C_1}{1+d}\right) + \left(\frac{C_2}{1+d}\right)^2 + \left(\frac{C_3}{1+d}\right)^3 + \ldots$$

The contribution of each cash flow C_i to the *Present Value (PV)* of the investment is weighted by the compounded discount rate $(1+d)^i$. Since the cash flows are generally preceded by an initial investment *IC*, the ***Net Present Value (NPV)*** adds this to the cash flow as negative value:

$$NPV = -IC + PV \text{ (Equation 45)}$$

Over the years, DCF has become a synonymous of NPV. But in fact it is important to keep the role of each separate: NPV represents the net totally of all contributions to the value of an investment; DCF is a technique used in the calculation of NPV.

For the purpose of this dissertation, we are adopting the assumption in which an investment in a product line is worthwhile if **NPV is greater than zero**. In addition, we must rewrite Equation 45 to have it compatible with the costs and benefits functions defined on the previous sections. First, we can use the vision of Clements et al. (Clements et. al., 2005), which assigns to the investment costs (IC) for a product line the sum of organizational costs and core asset base establishment[6]. As a result, a new term is derived to express initial investment of a product line *P*:

$$IC(P) = C_{org}(P) + C_{cab}(P) \text{ (Equation 46)}$$

Thus, we can assign for each cash flow C_i the values calculated for each reuse scenario function in the InCoME viewpoints, with the period of interest beginning in the start date *SD*, as follows:

- **Domain Engineering Cash Flow**: $B(A_\delta, SD)$, for the core asset base A_δ of a domain δ.
- **Product Engineering Cash Flow**: $B(p_k, SD)$, for the product p_k.
- **Corporate Engineering Cash Flow**: $B_P(SD)$ or $B'_P(SD)$, according to the reuse scenario, to the entire product line.

The next step is to define a new version of equation 46 for a generic viewpoint v, where v can be represented by any of the viewpoints of InCoME. Let assume a cash flow function, denoted by $B_v(SD)$, which returns the cost savings or losses for a reuse scenario in the viewpoint v with the period of interest starting in *SD*, as follows:

$$NPV(v) = -IC + \sum_{z=1}^{Y} \frac{B_v(SD+z)}{(1+d)^z} \text{ (Equation 47)}$$

where *Y* is the investment cycle measured in years and *SD* is the start date of the investment.

Return on Investment (ROI). After the NPV definition, we can now define a function to calculate the ***Return on Investment (ROI)*** achieved by setting up a software product line and using it as basis for product evolution. Despite the ROI is considered a measure for corporate level, each viewpoint can calculate its own ROI values. Moreover, it implies in using all reuse scenarios functions available to calculate the cash flow for present values. As cited in Chapter 2 in Equation 1, the ROI can be expressed by the difference between

[6] It reflects the proactive approach for a product line in which the entire core asset base is built up front.

benefits and costs, divided by investment costs. Accordingly, the ROI equation for a viewpoint v can be expressed by the following function:

$$\mathrm{ROI}(v) = \frac{\mathrm{NPV}(v)}{\mathrm{IC}(P)} \text{ (Equation 48)}$$

ROI values are expressed in a percentage from the investment costs and for the same value of NPV; the investment is more valuable that IC is smaller.

Payback Value (PB). Despite the Favaro et al. (Favaro, 1998) statement in which they emphasize that NPV is an essential approach to analyzing the value of investments in reuse[7], we are interested in evaluating not only *how* much of effort can bring to economic terms, but we also are interested in *when* the investment in a product line will be paid back to the organization. The economic function that performs this analysis is known as **Payback Value**. It meaning is easily intuitive, since it estimates the shortest investment cycle that makes the NPV a positive value for an investment cycle *Y*, i.e. the smallest integer value in Y which satisfies the following equation:

$$-IC + \sum_{z=1}^{Y} \frac{B_v(SD+z)}{(1+d)^z} \geq 0 \text{ (Equation 49)}$$

4.4.4. Simulation Model

Despite the computed results presented by the model can indicate if an investment is worthwhile, it remains unclear how sensitive the results are with respect with to the predicted and chosen values of all input parameters. In practice, the estimation of input variables can differ from the real values and the ROI values for a given product line scenario may lead wrong expectations, or in worst case, it may lead to wrong decisions concerning the investment (Muthig et al., 2006).

As cited in previously, it is possible to manage the uncertainty of input parameters by applying a *simulation model* in order to compute a large number of scenarios based on the random set of input values. One of techniques widely used to perform this is the *Monte Carlo simulation* (Malvin et al., 1986), (Trivedi, 2001).

[7] In fact, Favaro et al. states that NPV is superior to the other economics approaches to evaluate reuse

In order to apply Monte Carlo simulation for InCoME, we follow the approach described by Muthig et al. (Muthig et al., 2006), which defines a sequence of three steps:

- **Step 1**. To identify all input variable for which the stakeholders cannot provide accurate predictions[8].
- **Step 2**. To map each uncertain variable identified in the previous step to a suitable probability distribution. In this step, it is defined the range of acceptable values for each variable and a function specifying how likely a particular value will occur.
- **Step 3**. To generate random input numbers for uncertain variables based on the selected probability distribution. In this step, the economic functions defined on the investment analysis layer will calculate the NPV, ROI and Payback values for each set of random input values.

Next, the computed investment analysis values are put together and a frequency distribution is built, which highlights how likely it is to achieve the targets for NPV, ROI and Payback values through product line engineering.

Uncertain Variables. According to Muthig et al. (Muthig et al., 2006), there are a list of uncertain parameters related with a cost model for product line engineering. This set of variables had its uncertainty studied using SIMPLE (Clements et al., 2005) as cost model. We agree with the vision presented by this work and we are assuming the same list for InCoME added with the *frequency reuse value* for the assets that composes a domain and the *number of estimated products updates*. They can be viewed at Table 4.2, where the column *Uncertain Variable* describes the parameter and the column *Viewpoint Impacted* presents the relationship with the InCoME scenarios.

Table 4.2 – Uncertain Variables and Viewpoints Impacted

Uncertain Variable	Viewpoint Impacted
Number of products derived from a product line infrastructure	Product Engineering and Corporate

[8] In their work, Muthig et al. (Muthig, 2006) define explicitly the uncertain variables as the number of products, the commonality level, the effort for reuse, the change rate of core assets and the evolution rate in traditional (stand-alone) style.

	Engineering
Commonality level	Domain Engineering
Additional effort for making software reusable	Product Engineering
Change rate of core assets	Corporate Engineering
Evolution rate in traditional style	Corporate Engineering
Frequency of reuse	Domain Engineering
Number of Product Updates	Product Engineering and Corporate Engineering

Probability Distribution. Each variable identified as uncertain must be mapped into a probabilistic distribution. For InCoME simulation model, are used the *normal* and *uniform* distributions. Uniform distribution defines a range from a minimal to maximal value within any value is equally likely. For InCoME, the *number of products*, the *frequency of reuse* and *number of product updates* values are adequate for this type of distribution. Normal distribution defines a level that is below or above the average within defined lower and upper boundaries. The remainder of the variables follows this distribution.

Number of Trials. The confidence of the simulated results depends on the number of trials that are executed. According to Muthig et al. (Muthig et al., 2006), when using the uncertain parameters defined previously for a product line cost model it demands at least 20000 execution rounds of the simulation. The simulation model of InCoME was defined to use any number of trials, but we agree with the assumption of Muthig et al. which defines a minimum number of scenarios execution in order to improve the quality of the simulation results.

4.5. Using the Model

In the literature, the use of a cost model for software reuse is intrinsically related with a set of assumptions based on the type of reuse adopted by an organization. For InCoME, we are assuming the organization at least has a certain level of maturity in software development in order to achieve the main objectives for a systematic reuse approach. It implies in establish a data collection policy due the large number of parameters that will feed the model and keep the tracking of these metrics for planning and management purposes.

Despite the use of InCoME in an industrial scale is limited to date, we are coming up with a simple process to allow an organization to apply it when analyzing investments in a product line. In Figure 4.3, the use of InCoME is presented by an activity diagram, denoted using UML notation, through a sequence of activities:

- Establishing an Organizational Scenario;
- Functions Adjustments;
- Model Revision;
- Cost Factors Estimation;
- Model Data Population;
- Benefits Analysis;
- Economic Analysis;
- Product Line Investment Evaluation
- Cost Configuration Establishment

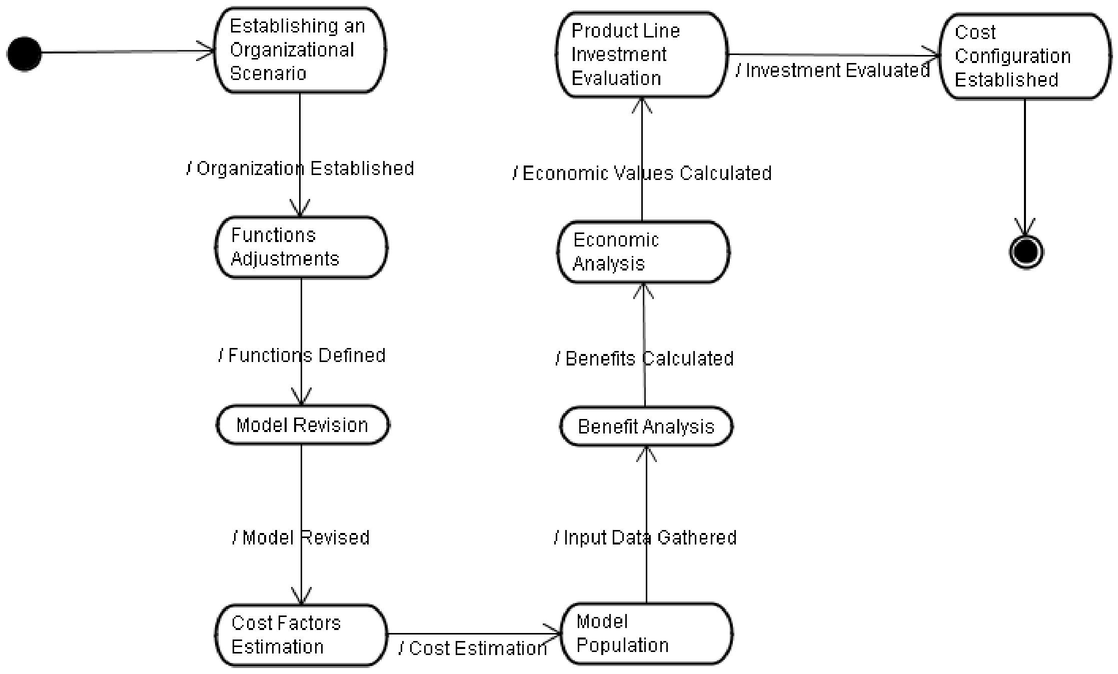


Figure 4.3 – InCoME Activities

In order to exemplify the use of the model, we are assuming the same scenario of a product line described in (Böckle et al., 2004) and (Muthig et al., 2006). In these works, an organization wants to estimate the return on investment (ROI) for fifteen software products with a product line umbrella. All the products have roughly the same size and complexity and reuse scenario in this case must figure out the cost of setting up a software product line to replace the fifteen existing products during one year of interest.

4.5.1. Establishing an Organizational Scenario

This first step gathers people in the organization to establish the organizational vision that must be used as a guideline to the next steps in using the model. It main objective is to find out the following set of information:

- **The product lines of interest**. In this point it must be described the specific product lines that will be evaluated by InCoME. Despite the model can be extended to support several product lines in this initial instance it will be available to work with only one product line at once.
- **The products that are currently involved or planned for the future**. At this point the organization must to define one important factor of cost: the number of products that will be developed through a product line approach.
- **The Investment Factors**. As cited previously in this chapter, InCoME defines the investment cycle (Y) in years, the discount rate (d), and the start date of analysis (SD). These factors are used in the investment analysis and they must reflect the organizational strategy to invest in assets and the amount of money that must be "returned" to the organization when the investment occurs.
- **Product Line scenarios description**. At this point, managers must describe the possible scenarios in which the product line will be evaluated. Each scenario describes the main points of evaluation that can be considered for an investment analysis. It is not necessary to define a scenario throughout mathematical equations, because in the next step we will make some functions adjustments.
- **The relevant time horizon**. It is important to estimate the duration of the period when the product line is to be analyzed. Typically, at least one year of product line engineering activities should be taken into account to allow the calculation of significant values.

To obtain this set of information an organization may have to prepare a questionnaire and apply it internally, or it can use some techniques to find out the business vision that encompasses an investment in a product line.

Example. In order to start the evaluation of the economic aspects of the product line, the hypothetical organization established the following set of information:

- Number of products: fifteen (all with approximately the same size and complexity).
- Investment cycle: 1 year.
- Discount rate: 10% a year.
- Scenario description: Setting up a product line to replace the fifteen existing products during the investment cycle.

4.5.2. Functions Adjustments

To allow an effective accounting for cost savings when applying product line engineering, the organization must have its reuse scenarios well defined. It implies in performing some adjustments in the equations defined by InCoME. As the same way as SIMPLE (Clements et al., 2005), InCoME allows the decomposition of its costs and benefits function in other equations, according to the needs of the organization. For example, if a financial manager wants to explore more the economic analysis he or she could derive functions such as *Profitability Index* or *Internal Rate of Return* (Favaro et al., 1998) from the *NPV*, *ROI* and *Payback* equations defined previously.

Other point to note is the fact that in the economic analysis the unit of measurement rarely is expressed in persons-hour or function points. It is possible to associated a monetary factor (e.g. average cost for one unit of persons-month per function point) and presenting the information in a language familiar to the stakeholder.

Example. In this point, we only have to review our reuse scenarios, since cost and economic functions are the same for this hypothetical scenario. The second reuse scenario for corporate viewpoint is suitable for the needs of the organization with *t=1 year*.

4.5.3. Model Revision

One important aspect of InCoME is the possibility to use it in different areas within an organization, according to the roles played for each stakeholder involved in the product line evaluation.

In this step, the entire model is presented to the stakeholders to make sure that their concerns are addressed and that the constructed model will answer the questions related with their respective viewpoints. In addition, the questions formulated in section 4.2.1 in this chapter must be answered to confirm the effectiveness of the model. If no, the model may have to be reformulated using more finely grained or more sophisticated cost estimates which will result in more precise or detailed data being collected and used to populate the formulas.
Example: Our hypothetical organization wants to establishes the economic benefits from the *corporate viewpoint*. At this scenario, there is no need to evaluate domain engineering and product engineering viewpoints, since the decision will occur only in the corporate level.

4.5.4. Cost Factors Estimation

In this step, the stakeholders must work out all the data needed to feed the cost functions of the model. For estimation purposes, the data gathered from the cost of building past products, domain engineering activities, product engineering activities, as well the measurement of the organizational cost, must be provided as input to the formulas. One point to note here is the maturity of an organization in estimates the low granularity factors, such as the cost to develop an asset by common and unique parts development. For its effectiveness, all cost factors must be fulfilled at the end of this step.

One alternative to provide this level of information is the use of benchmarks available in the case studies of the cost models definition. In the literature, there are a large set of assumptions (Mili et al., 2001) (Böckle et al., 2004), (Peterson, 2004) (Clements et al., 2005) for cost factors, such as values for C_{cabu}, C_{evo} and C_{reuse}. Thus, traditional approach for software size estimation can be used for this step, such as *COCOMO*, *Function Point Analysis* or *Use Case Points Analysis* (Albrecht, 1979), (Karner, 1993), (Boehm et. al., 1995).

***Example*:** At this point we are assuming that the organization gathered the cost estimations through its historical data and it had estimated some other factors. Since we are using the same scenarios from (Böckle et al., 2004) and (Muthig et al. 2006), Table 4.3 presents the cost factors for this hypothetical organization.

Table 4.3 – Cost Factors from the Example

Cost Factor	Value
C_{prod} – Cost of building one product	**12 Persons-Year**
C_{cab} – Cost of the core asset base	**13 Persons-Year**
F_{cab} – Fraction of the core asset base that changes with each new version of the product line	**10%**
C_{reuse} – Cost of using the core assets to build a product	**0,84 Persons-Year**
C_{unique} – Cost to build the unique parts of a product	**0,72 Persons-Year**
C_{org} – Cost of changing the organization to adopt product line engineering	**2,4 Persons-Year**
C_{evo} – Cost of evolving one product the old way	**2,4 Persons-Year**
C_{pl_evo} – Cost of evolving the product line through one evolution	**24,7 Persons-Year**

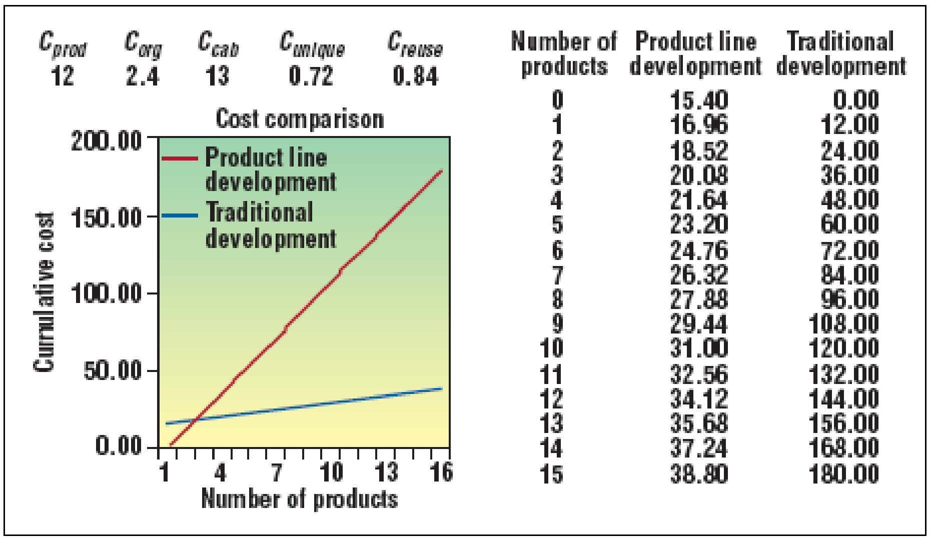
4.5.5. Model Population

In (Mili et al., 1999), Mili et al. define an **archival function** in order to store the data used for viewpoints analysis. In this sense, InCoME has the same approach for the data that has been gathered and inserted into the viewpoint formulas. The stakeholders can customize spreadsheets in order to record, update and track cost information on all four investment cycles.

4.5.6. Benefits Analysis

In this step, all cost savings are calculated and presented for each viewpoint, according to the reuse scenario defined for it. The stakeholders can view at this moment the amount of effort (or a monetary amount of that effort) saved or lost when applying a product line engineering approach.

Example: Using the functions defined for each viewpoint we can extract a set of information from the model computation. According the scenarios defined to the hypothetical organization, the Figure 4.4 presents the results of the cost estimations.



C_{prod}	C_{org}	C_{cab}	C_{unique}	C_{reuse}
12	2.4	13	0.72	0.84

Number of products	Product line development	Traditional development
0	15.40	0.00
1	16.96	12.00
2	18.52	24.00
3	20.08	36.00
4	21.64	48.00
5	23.20	60.00
6	24.76	72.00
7	26.32	84.00
8	27.88	96.00
9	29.44	108.00
10	31.00	120.00
11	32.56	132.00
12	34.12	144.00
13	35.68	156.00
14	37.24	168.00
15	38.80	180.00

Figure 4.4 – Cost Estimations (Böckle et al., 2004)

Using *Equation 44* of InCoME, we can now establish the benefits analysis for the corporate viewpoint scenario. That function expresses the cost savings for evolving n products in a traditional style (i.e., with no adoption of a product line) compared with the cost to evolve the same set of assets through a product line. In our hypothetical organization, the number of products (n) is fifteen, the investment cycle (t) is two years, and the percentage of updates of the unique parts (F_{unique}) is 6%. This configuration generated a loss of -2,8 Persons-Year for the first generation of the product line. Since for the following generations of the product line the organizational cost and the core asset base cost were already incurred, it produces a cost savings of 11.3 Persons-Year for each update of the product line.

4.5.7. Economic Analysis

Based on the cost savings or losses, an organization in this step can perform an economic analysis for the investment cycle defined previously. The results of this analysis are the **Net Present Value, Return on Investment, and Payback Value** for each viewpoint, according to the costs and benefits propagated for one cycle to the next.

According to the step 4.5.2, other appropriate economic functions could be able to run the data gathered across the engineering cycles.

Example: In our hypothetical organization the managers want to know the return on the investment (ROI) for the adoption of a product line with fifteen products during an entire year. To achieve this analysis we are assuming that the products will have four update during a year (i.e., $nr = 4$, according InCoME definition in *Equation 44*). We are interested in calculated the ROI using *Equations 47* and *48*. According the cash flows generated by each update, in the third generation of the product line the ROI can be estimated in 128%.

4.5.8. Product Line Investment Evaluation

This step addresses the evaluation if an investment in a product line is valuable, according to the values calculated in the economic analysis step. If the values expressed are not achieving the targets defined by the organization, an additional step can be performed in order to simulate ROI, NPV and Payback for a random range of cost factors. This step is performed by the elements defined for the simulation model, as presented previously. If it is necessary, a formal study must be made to identify the dependencies among the cost factors that may have some influence in the simulation. In fact, the use of simulation techniques, such as Monte Carlo simulation, can be used as a complement or an extension of the model (Muthig et al., 2006).

Example: According the Monte Carlo simulation method[9], we can estimate the risk to invest in that product line, considering a discount rate of 10% a year. Using the technique with the input parameters gathered from our hypothetical organization we can estimate in 77,3% the probability of *ROI>100%*. It implies in consider more than 20% of risk in losing money when investing in that product line. In another round of simulation we increase the maximum number of products to develop with more five products (i.e. $15 \leq n \leq 20$). The probability grows to the level of 78,6% of *ROI>100%*. Our last attempt to decrease the risk level is to decrease the number of annual updates, e.g. it can have up to two updates a year. With this configuration, the probability of ROI>100% is 92,3%. Similarly, there are a large set of combinations of the input parameters that can affect the result of the simulation. All the estimations were made through the execution of 20.000 instances of the model in a spreadsheet.

4.5.9. Cost Configuration Establishment

The final step in using InCoME is the establishment of a Cost Configuration (Nóbrega et al., 2006). It implies in using the estimates and economic analysis for all viewpoints as a historical data for future projects. According to the

[9] The definition and use of Monte Carlo simulation can be viewed in Appendix A

SIMPLE principles, which we used as guideline for InCoME and organization must keep its configuration of cost as simple and reliable as possible. As a consequence, the decision-making activities regarding the adoption of a product line approach (or its expansion for all products that can be developed) can be planned using that configuration.

The cost configuration for an organization must be flexible to allow its extension to other cost models different from InCoME for comparison purposes. In this sense, an organization can adopt the most suitable model that fits its needs or make adjustments in an existing one to reflect its reuse scenarios (Nóbrega et al., 2006).

Example: For the hypothetical organization it is recommended to establish a product line with number of product varying from fifteen to twenty products and with the maximum of two annual updates for each product. According the simulation values, this cost configuration has a low risk when considering the investment in that product line.

4.6. Chapter Summary

This Chapter presented the definition of the Integrated Cost Model for Product Line Engineering (InCoME), its objectives, foundations and elements. In addition, we proposed a process to apply InCoME when evaluating the adoption of a product line engineering approach.

The model was defined taking as basis the fundamentals of the Integrated Cost Model for Software Reuse (Mili et al., 2001) and SIMPLE (Clements et al., 2005). It defines a set of cost and benefits functions for a set of viewpoints, a vision of costs saving related to a specific set of stakeholders. It also addresses the economic analysis of viewpoints through the calculation of the Net Present Value, Return on Investment and Payback value. The results of the later is then evaluated according to the organization main objectives in which is based on a set of organizational cost factors

Table 4.3 presents a summary of all equations defined for InCoME with a brief description of them. Table 4.4 highlights the set of features from InCoME compared with the product line cost models studied in Chapter 3.

Upon the InCoME definition, the next chapter will present a case study in applying InCoME for product line adoption investment evaluation.

Table 4.4 – InCoME Equations

Description	Equation
Demand Function - returns the size per time unit necessary to develop the set of assets A_k for a product P_k	$D(A_k)=\frac{M(U_k)+M(C_k)}{T_p}$
The cost to build a core asset A_i	$C(A_i)=D(A_i)+CERT(A_i)+INS(A_i)$
Cost to develop a core asset base to a particular scope within a domain	$C_{cab}(\delta)=DA(\delta)+\sum_{\forall A_i \in A_\delta} C(A_i)$
Cost to develop the unique parts of a product that are not based on assets in the core base asset	$C_{unique}(p_k)=D(U_k)$
Cost of building *n* products independently	$C_{ind}=\sum_{k=1}^{n} C_{prod}(p_k)$
Benefit Function for Domain Engineering Viewpoint	$\forall A_i \in A_\delta, B(A_\delta,t)=\sum B(A_i,t)-C_{cab}(\delta,t)$
Benefit Function for Product Engineering Viewpoint	$B(p_k,t)=C_{prod}(p_k,t)- \lfloor C_{org}(p_k,t)+C_{cab}(A_k,t)+C(p_k,t)\rfloor$
First Benefit Function for Corporate Engineering Viewpoint	$B_P(t)=\sum_{k=1}^{n} C_{prod}(p_k,t)- \left[C_{org}(t)+C_{cab}(t)+\sum_{k=1}^{n}\left(C_{unique}(P_k,t)+C_{reuse}(P_k,t)\right)\right]$
Second Benefit Function for Corporate Engineering Viewpoint	$B'_P(t)=\sum_{k=1}^{n} C_{evo}(p_k,t)- \left[\sum_{k=1}^{n}\left(C_{cabu}(p_k,t)+\mathrm{F}_{\mathrm{unique}} * \mathrm{C}_{unique}(p_k,t)+C_{reuse}(p_k,t)\right)\right]$
Net Present Value for a Viewpoint v	$NPV(v)=-IC+\sum_{z=1}^{Y}\frac{B_v(SD+z)}{(1+d)^z}$
Return on Investment for a Viewpoint v	$\mathrm{ROI}(v)=\frac{\mathrm{NPV}(v)}{\mathrm{IC}(P)}$
Payback (the smallest integer value in Y which satisfies the following equation)	$-IC+\sum_{z=1}^{Y}\frac{B_v(SD+z)}{(1+d)^z}\geq 0$

Table 4.5 – A Summary of Features of PL Cost Models (with InCoME)

Cost Model	Features						
	Cost Function	Benefit Function	Predefined Reuse	Viewpoint	Pluggable Function	Investment Analysis	Decision Analysis

			Scenarios				Model
Poulin	X	X	-	-	-	-	-
ABC	X	X	-	-	-	-	-
Convergys	X	X	X	-	-	X	-
SIMPLE	X	X	X	-	X	-	-
COPLIMO	X	X	-	-	-	-	-
SoCoEMo-PLE	X	X	-	X	-	X	-
qCOPLIMO	X	X	-	-	-	-	-
Tomer	X	X	X	-	-	-	-
Schmid	X	X	-	-	-	X	X
InCoME	**X**	**X**	**X**	**X**	**X**	**X**	**(*)**

(*) The model defines a Simulation Model instead a Decision Analysis Model.

5 Case Study

Once InCoME has been described, some experiments must be performed in order to evaluate if the model achieves its proposed objectives. In this chapter, it is presented a case study to evaluate the effectiveness of the model through the application of InCoME in a real software development project of an organization that is studying the adoption of product line engineering. In the context of this case study *effectiveness* can be translated as *accuracy*, which means as *closeness to reality*.

In the literature, the definition of a reuse cost model is intrinsically linked with its formal validation in order to gain confidence for its application in a generic scenario (Wiles, 1999).

In this sense, a case study was performed to analyze if InCoME can effectively help an organization in their decision-making tasks when evaluating if an investment in a product line is worthwhile. For this case study, the model proposed by Wohlin et al. (Wohlin et al., 2000) was used as a guideline to the evaluation activities due to its approach provides a necessary formalism to this case study.

This chapter highlights the context where InCoME was applied, the presentation of the techniques used in this study, the evaluation itself and a discussion concerning the lessons learned in this study.

5.1. InCoME Context

The selected organization to perform the case study is currently the largest governmental information technology provider in Latin America, with its headquarters located in Brasília (DF). This organization has a five years contract to develop a family of products to the Brazilian federal bureau responsible to the public security operation for federal government. The scope of the information systems developed by the organization to its customer is related to the management of the new Brazilian passport and the control of immigration in airports, sea and fluvial ports and other border control offices. The major part of the effort to develop the family of products has been done in Recife (PE), where a group of fifty persons (including software engineers, test engineers, project managers, etc.) are involved since 2005.

During the period of the contract, the organization was demanded to increase the productivity of its development teams in order to deliver new products defined by the costumer, with no overrun in the hired costs, within established schedules agreed by both parts. In 2007, the organization established an internal program to achieve those productivity targets by the creation of a technical division. One of the initiatives proposed by this division is the adoption of a **reuse program** that will be integrated with the software development process adopted by the organization. This program focuses on the definition of a reuse process, including methods, tools and people education. Considering the possibility of adopting product line engineering in the context of the reuse program proposed for the organization, InCoME was applied to evaluate if this approach is economically viable.

Although the fact that there is no a product line formally defined for the customer projects, all products were built from a common set of assets that reflects the ***domain of passport management***. In this sense, InCoME was used to perform an economic analysis for the following scenarios:

The organization has a set of existing stand-alone products undergoing periodic evolutionary updates. Its managers wish to know which scenario has the biggest ROI value: **(i)** converting the products into a product line and continuing their evolution in that form or **(ii)** continue to evolve them separately.

5.2. Evaluation Techniques

For the purposes of this case study, the approach proposed by Wohlin et al. (Wohlin et al., 2000) brings a set of activities to evaluate InCoME:

- **Definition**. This step addresses the experiment definition in terms of its problem, objective and goals.
- **Planning**. In this step, the model presents three main concerns: the design of the experiment, the definition of the instrumentation and the identification of the possible threats.
- **Operation**. In this activity, there are two main sub-activities: the **analysis** and the **interpretation**. In this step, all experiment measurements are collected according to the planned activities.
- **Presentation/Package**. These steps represent the activities for presenting and packaging the set of results generated after the analysis and interpretation activities.

The approach described by Barros (Barros, 2001) for using Wohlin et al. model had also influenced the process to evaluate InCoME.

5.3. InCoME Evaluation

As stated in Chapter 4, the main objective of InCoME is helping an organization in its decision-making tasks when evaluating the investments in a product line from an economic point of view.

Due to the constraints related with time schedule, which can span across months, this evaluation was made in a time frame of two months, starting in November 2007 and finishing in December 2007. The next sections describe the work performed in these two months in order to evaluate InCoME in a formal way.

5.3.1. Definition

To define the experiment we used the ***Goal-Question-Metric approach (GQM)*** (Basili et al., 1994). According to Basili et al., the GQM is based upon the assumption that for an organization to measure in a purposeful way it must first specify the goals itself and its projects. Next, the organization has to trace these goals and provide a framework to interpret the data collected. Basili et al. resumes the GQM in three levels of measurement:

- **Conceptual Level**. Named as ***Goal***, this level addresses the objects of measurement and its reasons with respect to various models of quality, according to various points of view.

- **Operational Level**. Named as ***Question***, this level reflects the questions that characterize the achievement of a specific goal.
- **Quantitative Level**. Denoted by ***Metric***, it expresses the set of data associated with every question in order to answer it in a quantitative way.

In the next sections, the definition of InCoME experiment in terms of its goals, questions and metrics is presented.

5.3.1.1. Goal

G1. To analyze *the InCoME* for the purpose of *validating it* with respect to the *accuracy of the model* from the point of view of *financial managers and project managers* in the context of *product line engineering*.

5.3.1.2. Questions

Q1. How accurate is the model estimations, given accurate input parameter values?

Q2. Can the model indicate accurately the direction of an investment in a product line, according to its reuse potential?

5.3.1.3. Metrics

M1. *Accuracy Variation*. According to Bandinelli et al. (Bandinelli et al., 1996), the accuracy of the results of a reuse economic model are directly related to the quality of the data that is fed into the model. Availability of reliable data is a necessary condition to apply an economic model. The only way to know whether a model is correct is to start using it, validate it against real data and tune it to the organization specific characteristics. As a result, the accuracy of the model is dependent from the accuracy of its input parameters.

According to Clements et al. (Clements et. al., 2005), the input parameters gathered from organization historical data are the simplest and the most reliable source of effort. We assume here that these parameters can be considered accurate enough to provide good estimations. On the other hand, if such parameters are available only by expert judgment or market benchmarks, they can carry out a possible lack of accuracy due the fact that future products are possibly dissimilar to past products. In this sense, we defined an indicator for Q1, *Accuracy Variation* (ΔA), that is the sum of the number of input parameters gathered by historical data (P_h) less the sum of parameters gathered by other less-than-accurate sources (P_l), all divided by the number n of parameters, as follows:

$$\Delta A = \frac{\sum P_h - \sum P_l}{n} \text{ (Equation 50)}$$

The parameters that have its accuracy measured are restricted by the cost factors defined in Chapter 4: C_{org}, C_{cab}, C_{unique}, C_{reuse}, C_{evo}, C_{prod} and C_{cabu}. A positive value of ΔA indicates the model estimations has some level of accuracy (the opposite means the estimations is less accurate). As ΔA increases during the evaluation, the accuracy of the model values increases in the same proportion.

M2. *Homogeneity Degree.* In order to answer Q2, we first based our investigation in the work performed by Muthig et al. which indicates that the return on investment in a product line is associated with the number of products released in a specific period and the *degree of reuse* applied in its implementation (Muthig et al., 2006)[10]. In addition, Clements et al. (Clements et. al., 2005) state that to evaluate the capacity of a product line in generate a specific number of products, it is necessary to calculate its *homogeneity degree*. This metric measures how homogeneous the set of products are, and it is based on the set of requirements that apply for each product. It indicates the *"reuse potential"* for a product line by analyzing the percentage of *unique requirements* that satisfy a family of products. The values obtained with this metric are in the range of zero and one, with values near zero indicating a low reuse potential and values near one indicating a high reuse level. This metric are measured by the expression *($100\%-F_{commonality}$)* which assigns the value of 70% of commonality for all products (Muthig et. al., 2006). In this sense, to answer Q2 we defined an indicator that measures the correlation between the NPV estimated for a product line and its degree of homogeneity. The metric, denoted *r*, is presented below:

$$r = \frac{S_{rh}}{S_r S_h} \text{ (Equation 51)}$$

[10] There is a rule of thumb that defines in three the number of product releases that can be considered for a product line payback.

where S_r is the standard deviation for NPV measures, S_h the standard deviation for homogeneity degree measures and S_{rh} is defined by the equation

$$S_{rh} = \frac{\left(\sum_{i=1}^{n} r_i h_i\right) - n\bar{r}\bar{h}}{n-1} \quad \text{(Equation 52)}$$

where $\bar{r}$ is the mean of NPV measures $\bar{h}$ is the mean of homogeneity degree measures and *n* the number of measures. The *r* value is always in the range of -1 and 1 and a null value indicates that there is no correlation between NPV and homogeneity degree. A positive value implies that there is an association among the two measures in the same proportion. Negative values indicate that there is a correlation between them in an inverse proportion. There is a classification of the linear correlation where values greater or equal **0.90** are considered "strongly correlated".

5.3.2. Planning

After the definition of the experiment, a planning for its conduction is necessary. Here it is defined the context of the experiment, the subjects associated with it, the training activities, a pilot project, its instrumentation, the experiment criteria, null and alternative hypothesis, dependent and independent variables, the qualitative analysis, an internal and external validity, and a construction and conclusion validity.

5.3.2.1. Context

The objective of this study is the validation of the accuracy of InCoME based on a real reuse scenario within an organization. The project where this scenario occurred has been developed since 2005 and it was conducted in organization office located on Recife by a team with more than fifty engineers.

5.3.2.2. Subjects

The staff of managers involved with the project, playing the roles of financial or project manager, represents the subjects of the study. The first role is responsible to approve an investment for a hired project, and the later is responsible to entry the data for the model. Additionally, a subject representing a quality engineer plays a role that is responsible to collect the organization historical data. Eventually, a manager can plays the roles related with both financial and project management activities. The effort estimation and the model input parameters entry were performed by one project manager that fed the financial staff with the data gathered with the study.

5.3.2.3. Training

The training of the subjects was divided in two phases. In the first phase it was performed a study on the financial formulas available in Microsoft Excel[11] spreadsheet in order to develop the tool to represent the model. This activity took eight hours and it was performed by one subject representing the project manager staff. The second phase started when the spreadsheet was already developed and it included the data population with a product line scenario defined by Böckle et al. (Böckle et al, 2004) with a set of random values. This phase took two days or sixteen hours and it also was performed by the same subject of the previous training phase.

5.3.2.4. Pilot Project

The experiment itself was considered a pilot for the purpose of measuring the economic viability to adopt a product line. Due to the time constraints of the project a formal pilot project was not performed. The information needed for the experiment operation was gathered from the organization historical data and the estimation for a specific group of cost factors.

5.3.2.5. Instrumentation

The subjects received a spreadsheet with all cost factors equations defined for the scenarios defined by the model.

5.3.2.6. Criteria

The study focuses in evaluating the accuracy of the model. In this sense, the criteria demanded must be evaluated quantitatively through the effort necessary to convert the products into a product line (and continuing its evolution in that form) and the effort to build the same set of products in a stand-alone fashion. Moreover, the accuracy variation and homogeneity degree correlation will be evaluated qualitatively in order to answer Q1 and Q2.

5.3.2.7. Null Hypothesis

This hypothesis is that the experimenter wants to reject with a high significance as possible. In this study, the null hypothesis express that InCoME has no

[11] http://office.microsoft.com/excel

accuracy for its use in an organization and the economic analysis performed cannot indicate accurately if an investment in a product line is worthwhile. Thus, according to the criteria defined in previous section, the null hypotheses for this experiment are:

$H_0^{'}$ **:** $\mu_{\Delta A} < 0$

$H_0^{''}$ **:** $\mu_r > 0.9$ **and the investment is not indicated**

5.3.2.8. Alternative Hypothesis

This is the hypothesis in favor of which the null hypothesis is rejected. In this study, the alternative hypothesis reflects is accurate and it can indicate if an investment is worthwhile. The set of alternative hypothesis is:

H_1 **:** $\mu_{\Delta A} >= 0$

H_2 **:** $\mu_r > 0.9$ **and the investment is indicated**

5.3.2.9. Independent Variables

These variables will be manipulated and controlled along the study. For this study they are the model itself, the number of products derived, the commonality level among the products, the additional effort for making software reusable, the change rate of core assets, and the evolution rate in stand-alone development.

5.3.2.10. Dependent Variables

In this experiment the objects of the study is the dependent variables. They are the *ROI* value for the reuse scenario, the product line homogeneity degree, and the cost factor accuracy. *ROI* will be measured by the *Net Present Value* for the viewpoints divided by the investment costs for the reuse scenario. Product line homogeneity will be measured by the products requirements commonality (Clements et. al., 2005), and the accuracy for each cost factor will be measured through its identification as an organization historical value.

5.3.2.11. Qualitative Analysis

This step has the intention to evaluate the accuracy of the model in providing an investment analysis for a product line. This analysis will be performed through the spreadsheet with the costs and benefits functions. After the input of all cost parameters the spreadsheet will indicate if the estimation can be considered accurate enough to support the investment analysis.

5.3.2.12. Internal Validity

According to Wohlin et al. (Wohlin et al., 2000), the internal validity of the study is the capability to repeat its behavior into a new study. For this study, the internal validity is strongly dependent of the number of products derived from a product line infrastructure. According to Muthig et al. (Muthig et al., 2006) at least three products must be built (or planned to be built) in order to evaluate accurately the investments in a product line and this guideline was followed in this study.

5.3.2.13. External Validity

This step aims to measure the capability of the study to be affected by generalization. It implies in considering the capability to repeat the same study in other research groups (Wohlin et al., 2000). For this study, its external validity can be considered sufficient, since it aims to evaluate the accuracy of the model in a large organization through a big development project. Additional studies can be planned with the same profiles of subjects and reuse approach.

5.3.2.14. Construct Validity

This validation aims to measure the relation between the theories that is to be proved and the instruments and subjects of the study (Wohlin et al., 2000). In this study, a well-known domain and the legacy resultant of its implementation was chosen. In addition, the estimation values have a meaningful use in real investment analysis activities.

5.3.2.15. Conclusion Validity

This validation determines the capability of the study to generate conclusions (Wohlin et al., 2000). The conclusion of the study will be described by the use of descriptive statistic.

5.3.3. Project Description

The project used in the study was performed to build nine products related with the domain of passport management. As a basis for product development, a framework previously built in Java platform was available to derive the new products. The first product attempts to create an integration platform between the passport management system and an Automated Fingerprint Identification System (AFIS), and it was developed by a five persons team. The second product is an attendance control system to be used integrated with the main passport system, and seven persons compose the development team. The third product is a new version of a workflow management system, responsible to keep track of all passport emission tasks and a six person team developed it. Six additional products were constructed based on the same base and they completed the family of domain products. All of these products were based on an internal framework, which contains the reusable assets, including the source-code, requirements, user interface templates, architecture, test plans and other shared artifacts. Table 5.1 presents the summary of the project numbers.

Table 5.1 – Project Numbers

Experiment Data	Value
Days (01/11/2007 - 21/12/2007)	51
Number of Participants	2
Products Assessed	9
Set of Core Asset Base	10
Number of Reuse Scenarios	2
Total Development Effort (Stand-Alone)	29.610 Persons-Hour
Core Asset Base Development Effort	4.986 Persons-Hour
Domain Analysis Effort	476 Persons-Hour
Organizational Cost	3.256 Persons-Hour
Product Evolution Effort (annual mean)	900 Persons-Hour
Core Asset Evolution Level (annual mean)	10% of Core Asset Base

5.3.4. Instrumentation

5.3.4.1. Selection of the Subjects

The subjects selected to the study were composed by one project manager, responsible to the project activities for each product and a senior manager, which plays the role of the financial manager. The group of subjects was selected by convenience sampling (Wohlin et al., 2000), representing the nearest and most convenient people related with the experiment.

5.3.4.2. Data Validation

The data used in this study was validated by descriptive statistics that provide simple summaries about the sample and the measures. It is used to present quantitative descriptions in a manageable form in order to help the analysis of a large amount of data in a sensible way.

5.3.4.3. Instrumentation

Before the beginning of the experiment, all instruments must be ready for use, including the cost factors spreadsheet.

5.3.5. Operation

5.3.5.1. Experimental Environment

The case study was conducted during November-December 2007, at the organization office in Recife. It was performed by the project manager allocated for each product and by the senior manager. The quality team indirectly supported the data collection task.

5.3.5.2. Training

The training concerning Excel economic formulas and the spreadsheet tool took twenty-four hours in the beginning of November 2007.

5.3.5.3. Costs

The subjects involved in the study were hired by the organization and according to the activities cited in the planning phase they spent two months in general, including the instrumentation preparation, the training activities, the operation as well as the analysis tasks.

5.3.6. Analysis and Interpretation

5.3.6.1. Training Analysis

The subjects involved in the study were trained in order to evaluate the results of the experiment. The training to analyze the data gathered was performed in four hours and it consisted of the study of statistical functions (e.g. linear correlation) and Monte Carlo simulation with the intention to use the spreadsheet with the model definition.

5.3.6.2. Quantitative Analysis

The analysis had compared two distinct scenarios for a product line composed by nine different products. The first scenario ***(SC1)*** addresses the situation where the organization develops the nine products in a traditional style, i.e. out of the umbrella of a product line, and continues evolving them in this way. The second scenario ***(SC2)*** reflects the situation where the organization wishes to convert all products using product line engineering approach and continue to evolve them in that form.

Each product has some similar features, including the JEE platform[12], the passport management domain, among other non-functional requirements. The engineers allocated for each product development have a good experience in the development platform and environment and they are considered as a group of experienced software engineers. All the development teams have a good experience in using a systematic process to develop software, since the organization was CMMI Level 3 certified[13].

For each scenario, the subject that played the role of a project manager retrieves the effort estimations and fulfills the spreadsheet with the data. Next, for the second scenario, he performed the domain analysis in order to establish the core asset base for all products development. The variability of each product was analyzed and the unique parts were identified and its effort fulfilled in the spreadsheet. Moreover, for both scenarios the data related with organizational costs, core asset base development, reuse level, product and assets evolution, was estimated or retrieved according to its availability as historical data. The Table 5.1 presents a summary of the effort and the estimated number of updates for each product. The effort to develop the products is measured in persons-hour.

Table 5.2 – Products, Effort and Number of Annual Updates

Product	Effort	#Annual Updates
P1	5346 PH	8
P2	3276 PH	4
P3	3096 PH	2
P4	5094 PH	8
P5	2700 PH	2
P6	2070 PH	4
P7	3060 PH	2

[12] http://sun.java.com
[13] http://www.sei.cmu.edu/cmmi/

P8	3492 PH	4
P9	1476 PH	4

In Table 5.2 is presented the remaining relevant cost factors to all products over a product line engineering approach. Again, the effort unit is expressed in persons-hour.

Table 5.3 – Products Cost Parameters

Product	Unique Parts Effort	Reuse Level	Product Evolution Effort (Annual)[14]
P1	1134 PH	144 PH	900 PH
P2	1260 PH	112 PH	900 PH
P3	1116 PH	96 PH	900 PH
P4	486 PH	144 PH	900 PH
P5	882 PH	96 PH	900 PH
P6	828 PH	48 PH	900 PH
P7	288 PH	112 PH	900 PH
P8	576 PH	96 PH	900 PH
P9	234 PH	48 PH	900 PH

The Core Asset Base cost values are presented in Table 5.3 and the domain analysis effort is described in Table 5.4. For each asset it was assigned a constant effort to certify it and insert it into a repository. This cost was estimated in 8 Person-Hour for each factor.

Table 5.4 – Core Asset Base Cost Parameters

Core Asset	Effort	Frequency of Reuse	Asset Evolution Effort (Annual)[15]
A1	594 PH	8	59,4 PH
A2	360 PH	3	36,0 PH
A3	954 PH	5	95,4 PH
A4	648 PH	9	64,8 PH

[14] Due to a lack of reliable data, the effort necessary to evolve the products in a traditional style was normalized according the historical average of products evolution

[15] For the same reason, the effort to evolve the core asset base is normalized as a rate of 10% of each asset. In (Böckle et. al., 2004) the same fraction is used and it was considered as a reasonable value.

A5	540 PH	5	54,0 PH
A6	198 PH	1	19,8 PH
A7	180 PH	3	18,0 PH
A8	360 PH	3	36,0 PH
A9	738 PH	6	73,8 PH
A10	414 PH	2	41,4 PH

Table 5.5 – Domain Analysis Cost Parameters

Cost Factor	Effort
Commonality and Variability	160 PH
Product Line Scope Definition	96 PH
Generic Architecture Design	160 PH
Development Environment	40 PH
Testing Architecture Definition	40 PH
Other Development Artifacts	80 PH

Finally, the Table 5.5 presents the effort to change the organization in order to support the adoption of a product line.

Table 5.6 – Organizational Cost Parameters

Cost Factor	Effort
Internal Reorganization	640 PH
Process Improvement	320 PH
Training	400 PH
Repository Purchase and Installation	1800 PH
Product Line Operational Costs	96 PH

Using descriptive statistic, the data collected in the case study were grouped in three different perspectives: the *Viewpoints Analysis*, the *Investment Analysis* and the *Simulation Analysis*.

5.3.6.3. Viewpoint Analysis

The Viewpoint analysis was performed by the execution of the model through the spreadsheet calculations. The data obtained for the estimation was separated into two distinct groups: data gathered from historical data and data gathered from expert judgment. In the first group we found the products effort, product evolution effort, the most of organizational effort factors, and the unique parts. The second group includes the reuse level, the core asset base effort and the asset evolution effort. This distribution implies in an **accuracy variation (ΔA) of 14,2%**. This measure **rejects** the null hypothesis $H_0^{'}$ **: μΔA < 0**, which **confirms** the alternative hypothesis H_1 **: μΔA > 0**. Despite the value can be considered low to the organization purposes, it can grow if the factors related with core asset base cost can be retrieved in future project from the organization historical database. Since the organization does have not a domain analysis processes instantiated these group of factors can only be estimated by expert judgment.

The data collected from the model computation indicates that the Domain Engineering activities have a positive balance in its benefits when compared to the costs of building the assets. Figure 5.1 presents the values collected from the benefits and cost equations. The final balance is 20534 Persons-Hour of effort savings.

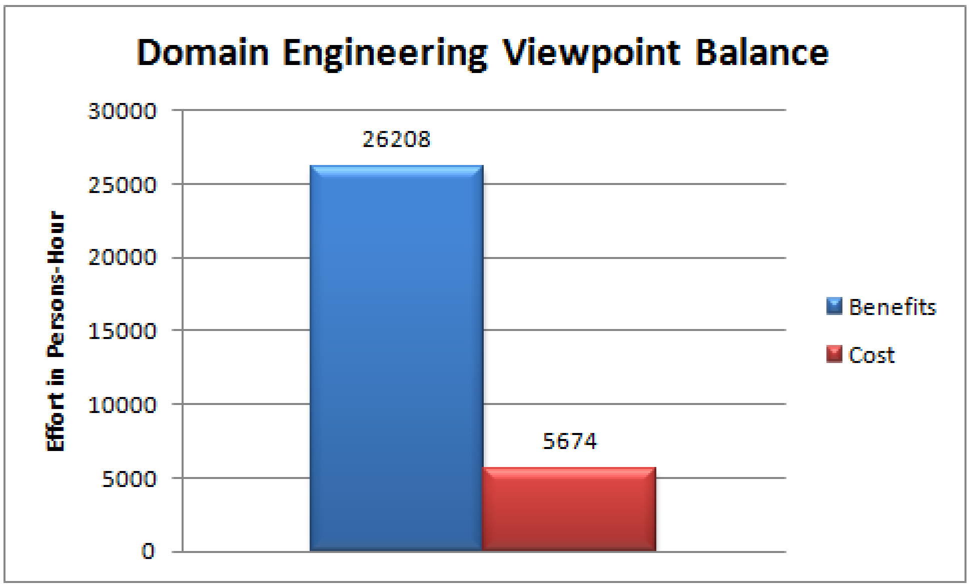


Figure 5.1 – Domain Engineering Viewpoint Balance

The Product Engineering Viewpoint was analyzed by the estimation of the cost savings or losses for each product defined previously within the two reuse scenarios. In general *SC1* presents the best results considering that 77% of products produce cost savings when comparing the traditional development approach and the product line engineering approach. In SC2 all products produces negative balances which implies in a higher cost to produce the first generation of products for a product line. Figure 5.2 presents the balance for the first generation of a product line for SC1 and SC2. Despite this negative scenario, SC2 cannot be completely discarded due to the evolution of the products along a specific time period. When we consider all subsequent releases for a period of a year the scenario balance became positive. This is explained due to the fact that organizational costs and core asset base cost were already incurred and the most significant factor for cost savings is the reuse level associated with the core asset base evolution effort. Figure 5.3 presents the balance for subsequent generations of a product line during a year.

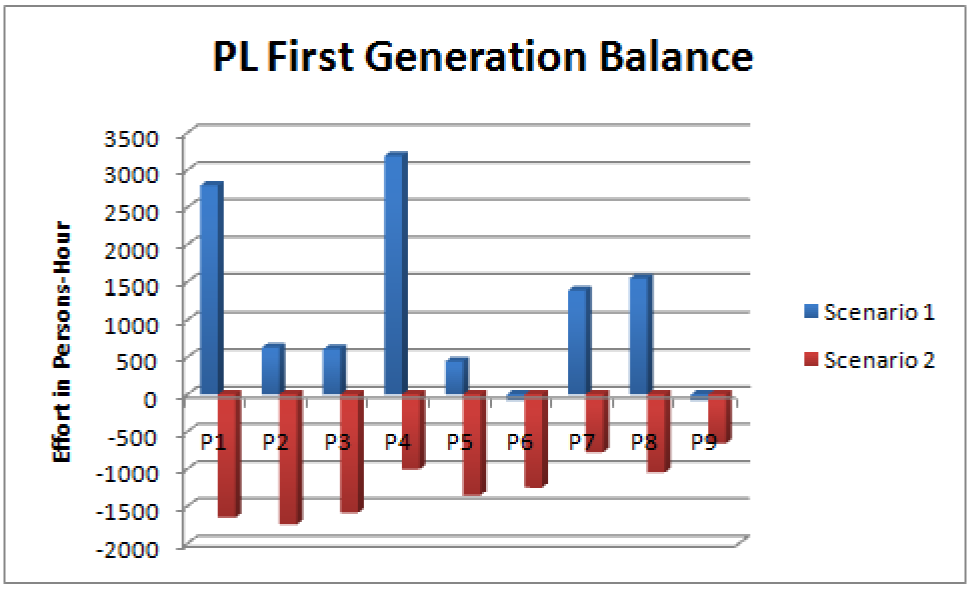


Figure 5.2 – PL First Generation Balance

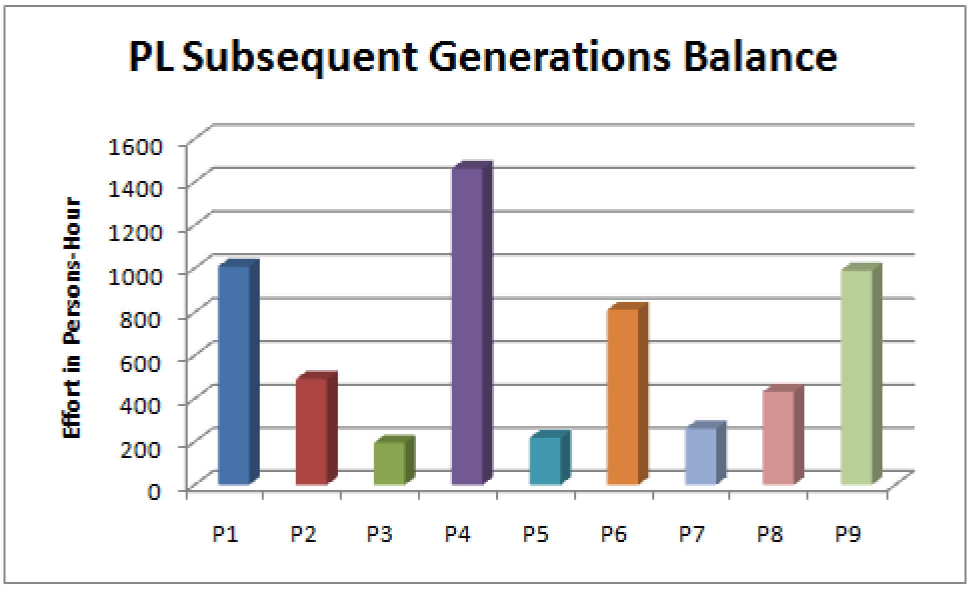


Figure 5.3 – PL Subsequent Generations Balance (within one year)

The last viewpoint analyzed was the Corporate Engineering which presents a balance as the sum of all products, considering *SC1* and *SC2*. The cost savings for this viewpoint can be analyzed using the same assumptions of the individual products. The cost savings to build a product line from scratch instead converting it are impressive for the first generation of products. The subsequent generation presents positive values for both scenarios, but the upfront costs are strongly favorable to SC1. Figure 5.4 shows the balance for Corporate Engineering Viewpoint.

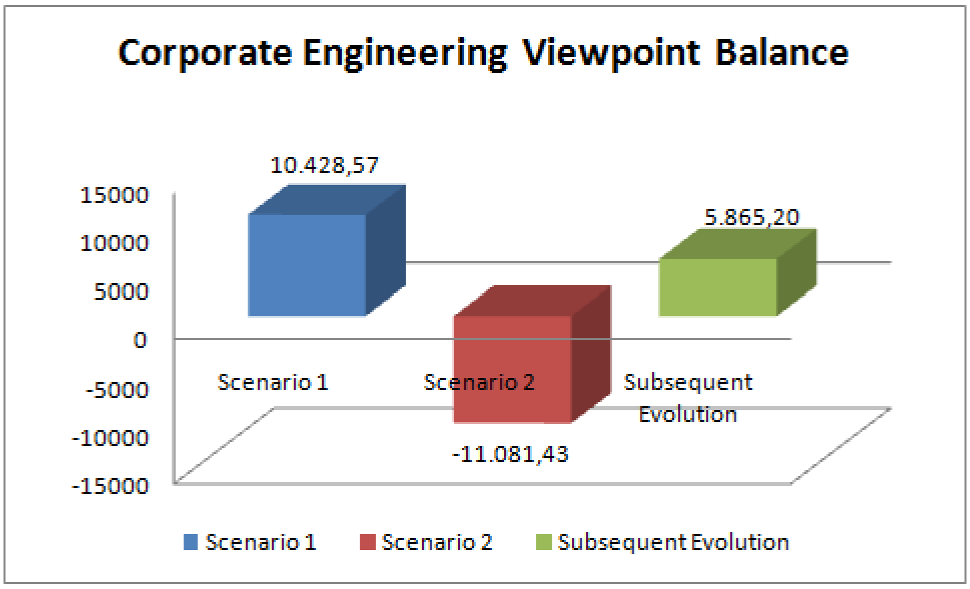


Figure 5.4 – Corporate Engineering Balance

Conclusion: The experiment indicates that the SC1 saves more effort than SC2. If we consider that SC1 is a situation associated with the *proactive approach* for product lines (Clements et. al, 2002) where the core asset base is entirely built from scratch, the scenario is compatible with the study of Clements (Clements, 2002) which states that this strategy can predict well the costs for future developments. On the other hand, SC2 stands for the extractive *approach* (Clements, 2002) which can help in the transition from conventional to software product line engineering, but with a higher cost than the other strategy.

5.3.6.4. Investment Analysis

After execution of the experiment for all viewpoints, the set of results had fed the economic functions in the spreadsheet in order to perform the analysis from an economic point of view. According to the data obtained from the previous section, seven of nine products of SC1 presented positive cost savings, which is an indicator that they can be considered to develop. But when we consider an economic analysis for the present values gathered from products benefits, we can have another direction of decision.

In this step of the experiment it was calculated the values of three economic functions for each viewpoint: Net Present Value *(NPV)*, Return on Investment *(ROI)* and Payback Value *(PB)*. It was taken into account a *period of five years* of investment analysis over a discount rate of *10% a year*. The results indicated that in a long term P6 and P9 can get the money invested back, in opposition to vision established in the viewpoints analysis. It can be explained by the high cost savings achieved for subsequent versions of those products (811, 80 PH and 990 PH, respectively). On the other hand, P3 and P5 had its investment not recommend, since NPV is negative for the period analyzed. Again, the behavior can be explained by the low values achieved for subsequent versions (193,80 PH and 217,20 PH, respectively) which are not sufficient to make this investment worthwhile[16]. For the same reason the investment in SC2 can be considered now for three products - P1, P4, and P9 - in opposition to the

[16] It is important to remember that an investment is indicated only if the NPV is greater than zero. Otherwise, it is recommended to invest in other type of assets.

viewpoint analysis where for the first generation there was losses. The Figure 5.5 shows the values obtained after the NPV analysis.

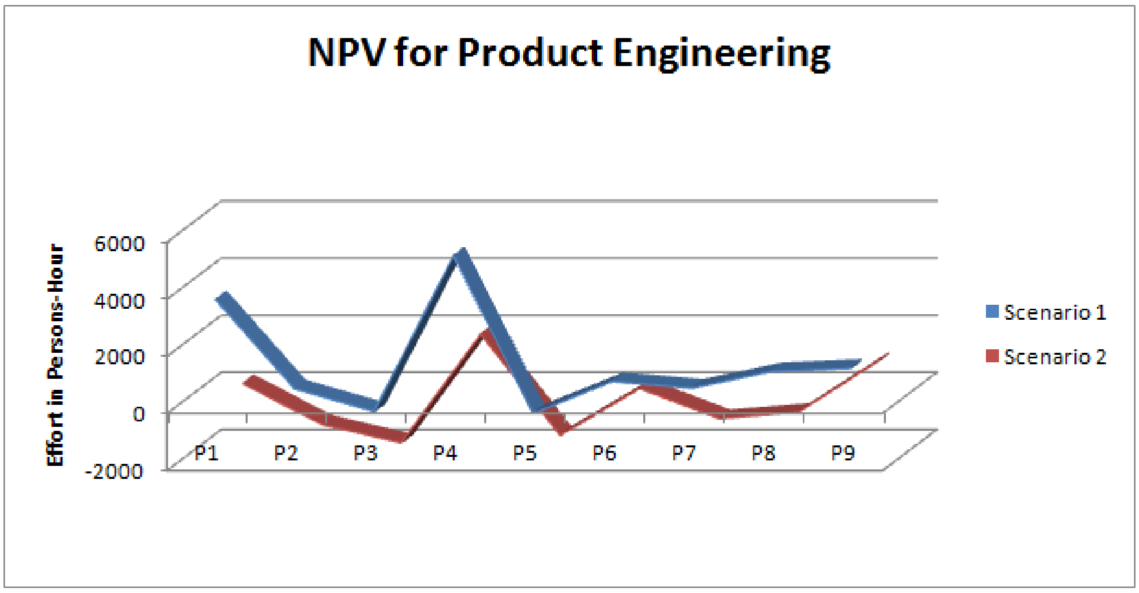


Figure 5.5 – NPV for Product Engineering

The investment analysis for the Domain Engineering Viewpoint demands a strong recommendation to invest on it, since its NPV had indicated a **payback** in only **three months**, which can be justified by the higher frequency of reuse of the core asset base.

Next, it was collected the results of the Return on Investment (*ROI*) function, which presented another set of interesting results. For SC1, the *ROI* achieved indicated that despite seven of nine products has the investment indication, only three of them have a *ROI* value greater than 100%[17], respectively P1, P4 and P9. Through the *ROI* analysis SC2 has a worse evaluation than the *NPV* analysis due only one product (P4) has been presenting a *ROI* greater than 100%. Figure 5.6 presents a summary of the *ROI* estimations for Product Engineering Viewpoint, with the green line indicating the desired level of the measure.

[17] The ROI value indicates the totality of the investment that returns to the organization. In this sense, values greater than 100% are a good indicator for an investment.

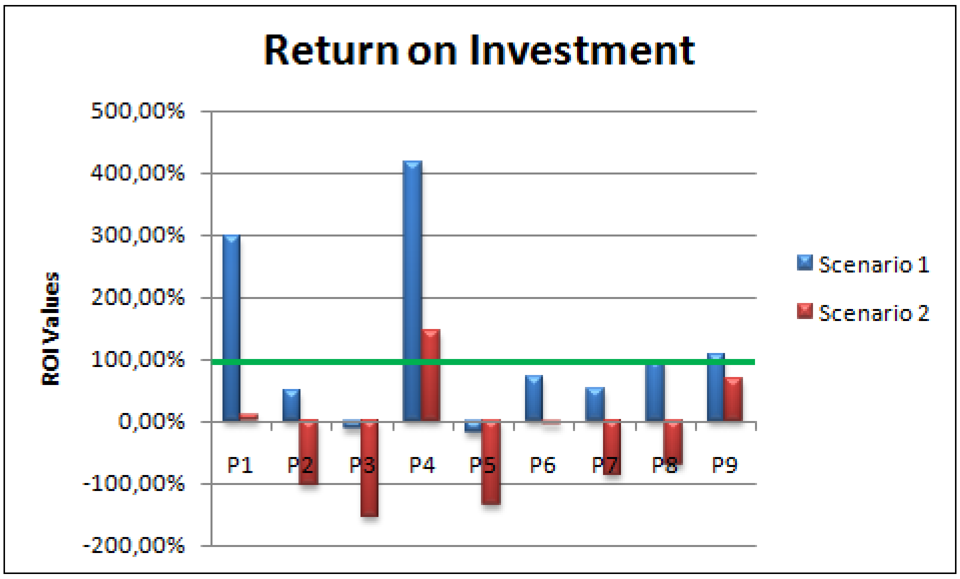


Figure 5.6 – ROI for Product Engineering Viewpoint

One point to note here is the behavior of the Payback value measurements which indicates when the investment will return to the organization. The Table 5.6 presents a summary with the payback values for SC1 within Product Engineering Viewpoint.

Table 5.7 – Payback Values for Product Engineering Viewpoint

Product	Payback Value
P1	5 months
P2	2 years and 3 months
P3	4 years and 5 months
P4	4 months
P5	No Payback
P6	2 years and 8 months
P7	11 months
P8	9 months
P9	2 years and 4 months

One point to note is even some products presents negative values for NPV and ROI less than 100%, there is a payback value associated with them. Only one product (P5) has a payback outside the investment period (five years)[18].

Finally, for the Corporate Engineering Viewpoint, the investment analysis was also performed using the sum of all products values. In this sense, the NPV for all products of SC1 presented a positive value, indicating that the investment is valuable. For SC2, the values are negative and the investment on it is not recommended. At the same way, ROI for Corporate Engineering Viewpoint of SC1 presented a value greater than 100% which implies in a good indicator to invest in that scenario. The payback value for SC1 is estimated in 10 months.

$Conclusion$: Despite the viewpoint analysis indicated interesting relations for cost savings, the investment decision only can be made after the economic analysis of all reuse scenarios. According to Favaro et al. (Favaro et. al., 1998), the NPV is the more reliable technique to evaluate investments in software assets. ROI and $Payback$ were used to reinforce the investment decision as a “second level “of decision.

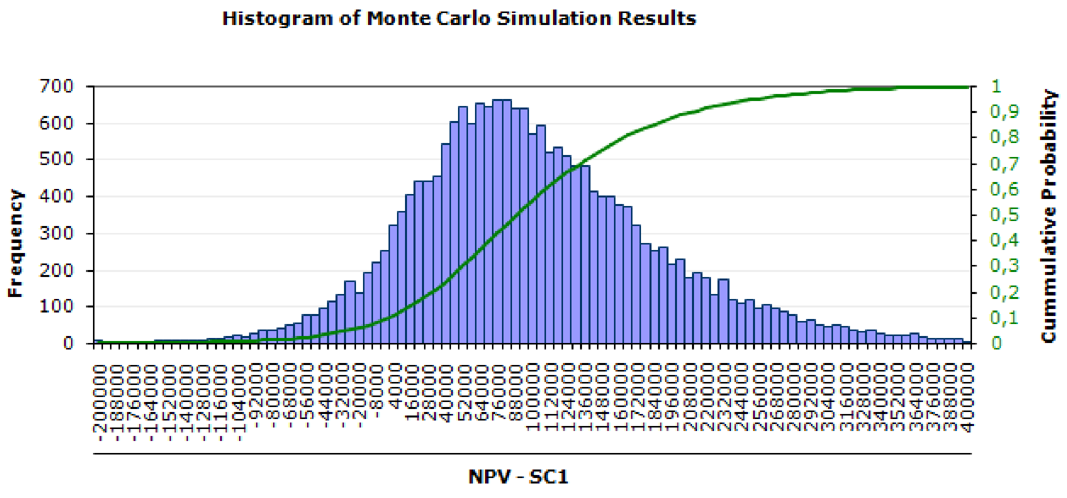
5.3.6.5. Simulation Analysis

The investment analysis was used to indicate if an investment in a product line scenario is worthwhile from an economic point of view. In other words, the model must indicate accurately if the investment has a certain $\boldsymbol{level\ of\ risk}$. According to the data gathered from investment analysis, we investigated the correlation between the reuse potential of the product line and the indication of investment presented by the model.

To avoid the use of a unique scenario for this investigation, we execute the same experiment for a set of random scenarios, using the Monte Carlo simulation technique. The first step was to create a large number of input parameters for SC1 and SC2. In the spreadsheet, it was defined twenty thousand (20.000) variation of both scenarios, each of them with a different set of input parameters. According to the studies performed by Böckle et al. (Böckle et. al., 2004), Clements et al. (Clements et al., 2004) and Muthig et al. (Muthig et al., 2006), a reasonable homogeneity degree of a product line has the $F_{commonality}$ factor in average of 70%. The data gathered from the commonality analysis

[18] The Payback technique used here is not considering an amortization schedule through a discount rate. This approach is known as “Discounted Payback”

indicated a value of **75,59%** for the homogeneity degree of the product line in this case study. In addition, we found a standard deviation of **12,43%** for this metric. After the execution of the twenty-thousand scenarios, the correlation between the homogeneity degree and the model probability to generate positive values of NPV for SC1 is **0,993** and **0,995** for SC2. For SC1, the probability of generating a NPV value greater than zero is **90,4%** and for SC2 the probability is **82,3%,** as presented in Figure 5.7. We can assume these probability values can indicate that there is a low risk in invest in that product line. This implies in **reject** the null hypothesis $H_0^{''}$**: μr > 0.9 and the investment is not indicated** and **confirms** the alternative hypothesis H_2**: μr > 0.9 and the investment is indicated.**



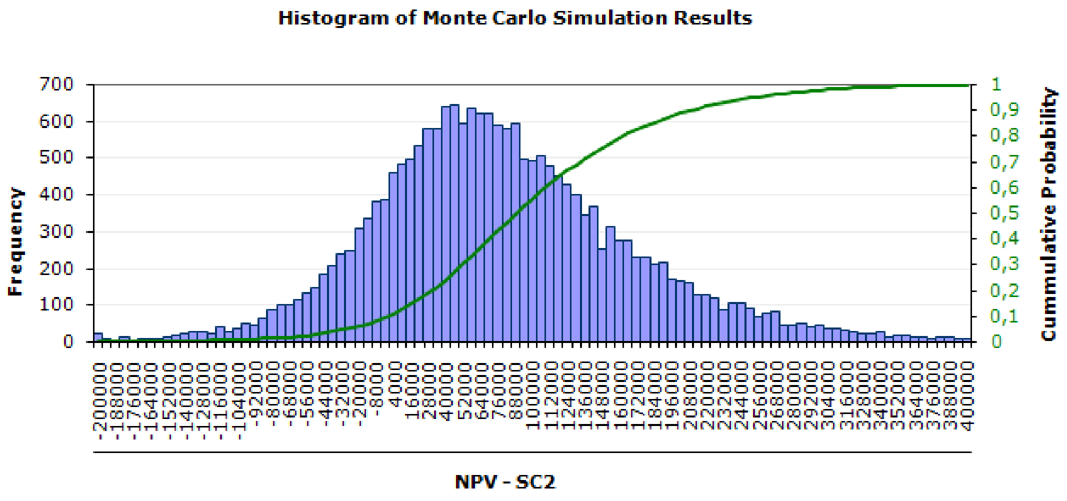


Figure 5.7 – Simulation Results (NPV Probability for SC1 and SC2)

Conclusion: The model can indicate accurately if an investment in a product line is valuable. This fact was proved by the correlation between the risk to invest in a product line and its reuse potential, measured by the homogeneity degree metric (Clements et. al., 2005). The risk of NPV to be negative is less than 10% for SC1 which indicates that the product line may have a **positive** $\boldsymbol{ROI}$ for the most situations of the scenario. SC2 presented more risk involved with product line adoption, and as result, a positive ROI for this scenario could not be achieved with a probability of 18%.

5.3.6.6. Qualitative Analysis

After concluding the quantitative analysis for the case study, the qualitative analysis was performed. This activity focused in analyzing the quality of the material used in the experiment. Here, the most of the difficulties lies on the use of the spreadsheet to perform the estimations. Because of the large number of reuse scenarios generated in a random way, it took more than three minutes to perform an entire simulation round. Each change performed in the spreadsheet implied on a full calculation of the random scenarios. This fact contributed for making the study a low productive experience.

This low level of productivity can be explained by the use of a spreadsheet with no refinements to support the simulation activities. It was studied the use of commercial packages in order to perform the Monte Carlo simulation experiment, but due to the lack of a formal budget associated with this case study it was not possible to acquire such software licenses.

5.4. Lessons Learned

After concluding the case study, there are some points that should be considered in order to repeat the experiment in a general way. The points that should be improved are:

- **Reuse Effort Data**. One crucial aspect of InCoME is its application by an organization that has a systematic reuse process in use by its development team or has the intention to adopt product line engineering approach. It is important that the historical data concerning the reuse effort might be easily gathered, providing more confidence and accuracy to the estimations. The organization where the study was performed does not have a formal reuse process in use and some cost factors for reuse were estimated using market and other research benchmarks.

- **Management Feedback**. The final results of the study were presented to the managers as forecast of a product line adoption by the organization. For the next experiments, the relevance of the economic analysis should be evaluated through the application of a questionnaire fulfilled by the managers.
- **Pilot Project**. To reduce the time to start the experiment it is fundamental the application of the model in a pilot project. In this sense, it is recommended that an organization must have to define a set of random product line scenarios in order to calibrate the spreadsheet used in the experiment. A suggestion of such scenarios is given by Böckle et al. (Böckle et al, 2004) e it was followed by Muthig et al. (Muthig et al., 2006).

5.5. Chapter Summary

This chapter presented a case study using InCoME within an organization that has been evaluating the economic aspects in adopting a product line approach to develop a family of products. It was covered the context of the study, the techniques used in the evaluation, the definition of the study, a formal planning, the description of the project, the experiment instrumentation, the operation, the analysis and interpretation and the lessons learned in the experiment. The study analyzed the possibility of the subjects in using InCoME to evaluate a real reuse scenario for an economic point of view. It approaches the model accuracy and the correct indication for a viable investment in a product line.

The analysis has shown that InCoME can be accurately calibrated if the input parameters provided for it have as a source the organization historical data. It also indicated that the homogeneity degree implies in a certain level of reuse potential, which can indicate the viability of an investment.

Finally, the study identified some improvements and directions for future experiments, concerning the measurement of reuse effort and management feedback.

The next chapter will present the conclusions of this work, its main contributions and directions for future works.

6 Conclusions

Product line engineering appears to be a solid and consistent approach to start a reuse program within an organization. In this sense, the use of a cost model to evaluate whether or not to invest in that approach can be considered a key aspect to improve the level of confidence in decision-making tasks. As discussed in Chapter 2 and 3, there are a large number of models available for use by software development community. According to the survey presented in those chapters, to be effective a cost model for software product line has to focuses in both cost estimation and investment analysis. In addition, such models must be the most flexible as possible in order to allow the definition of dynamic reuse scenarios that can occur within an organization. However, the models studied in this work lack in providing the flexibility to define new scenarios and an integrated viewpoint of costs, benefits and investment analysis from the point of view of different stakeholders.

In this sense, in order to solve the issues related with the available cost models for product lines, this dissertation proposed the **Integrated Cost Model for Product Line Engineering (InCoME)**, which defines a set of cost and benefits functions, a set of reuse scenarios and a set of economic functions to perform investment analysis for different viewpoints. The foundations of the model are based on an extensive survey of existing cost models, their failures and good practices.

6.1. Research Contributions

The main contributions of this work can be summarized in the following aspects:

- **The Key Developments in the Field of Reuse Cost Models**. This aspect focused in investigating the origins of cost models for software reuse, including its definition, main features, classifications and state-of-the-art. In addition, a comparison of nineteen models found in the literature was made, presenting its most relevant features.
- **A Survey on Software Product Line Cost Models**. After the reuse cost model definition, this dissertation focused in investigating the most important cost models for software product line engineering, presenting its main features and a discussion of the most important aspects that can be considered in the definition of an effective model. The weakness and strengths of nine cost models were studied in order to establish a basis to solve the issues found.
- **The Integrated Cost Model for Product Line Engineering (InCoME)**. Next, with the issues of the models identified, we defined the InCoME, which is a model to integrate the various elements that can be considered relevant to evaluate investments in a product line. Its foundations are based on the fundamentals of two different but complementary models, coming up with a new approach to evaluate the economic aspects of a product line. In addition a discussion on how to apply the model into an organization was presented.
- **The Case Study**. In order to evaluate the accuracy of the model, a case study was performed for a real reuse scenario. The study comes up with two new metrics to evaluate the accuracy and the relevancy of the estimations generated by the model. In its operation, the study analyzed the model as quantitatively as well as qualitatively, and it indicates that the model can be accurately calibrated to produce sounds estimations. At the end, some improvements and directions were presented for future experiments.

We can highlight the main contribution of this work as the definition of an integrated model to evaluate the economics aspects of a product line, with the description of a set of mathematical functions to estimate cost and benefits, and the results propagation to different viewpoints of an organization. Finally, we highlight the approach to deal with investment analysis for product line engineering.

6.2. Related Work

In the literature, some related work could be identified during this research. In Chapter 3 nine models were presented, a few of them with some level of similarity with this work. We can state that the key difference of this work and the others is the definition of a model that integrates the most significant aspects relative to an effective cost model for product line engineering. These aspects are related to the integration among cost estimation (Mili et al., 2001), investment analysis (Favaro et al., 1998) and reuse scenarios (Clements et al., 2005). Another point to help distinguishing between InCoME and the other models is a new way to analyze the worthiness of an investment in a product line by the use of simulation (Muthig et al., 2006).

6.3. Future Work

According to the contributions obtained during this dissertation, some directions for future work can be proposed, as an extension of the study performed for InCoME definition. The directions are:

- **InCoME Validation**. According to Böckle et al. (Böckle et al., 2004), *constructing economic models is one thing, but building practical ones is another*. In this sense, new studies in different reuse scenarios should be performed in order to calibrate the model and improve its level of confidence. The initial evaluation of InCoME was performed in an organization that has no systematic reuse program and additional product line scenarios cannot be tested as well. A possible direction is to repeat the experiment in a set of organizations that have one or more product lines and investigate the real benefits achieved by product line adoption, establishing new benchmarks for reuse community.
- **Product Line Approach**. Modeling the evolutionary cycles for product lines implies in the interpretation of the cost incurred during a period of interest. The need to account for time periods is true for all cost functions (Clements et al., 2005). This aspect is relevant when comparing reactive product line development with proactive approach. In this context, one line of investigation could be established in order to compare the reuse

scenarios associated with both approach and the investment analysis of its viewpoints. This study can suggest the most suitable approach (reactive or proactive) for an organization from an economic point of view.

- **Dependency and Sensitivity Analysis**. Defining a cost model in terms of its input parameters it is not a trivial task due to the dependencies that can exist among them. Uncovering and quantifying these dependencies is a key factor to help the evaluation of model accuracy (Peterson, 2004). Moreover, it may exists a particular scenario that is more sensitive to input changes than others and the identification of the set of variables that apply in this case can help an organization in calibrating the model. A possible direction here is to perform a sensitivity analysis for a large set of results provided by the model and investigates its correlation[19].
- **Decision Model**. According to the survey performed on Chapter 3, one relevant aspect to define an effective cost model for software product line is the use of a decision model (Schmid, 2003). Even with its relevancy recognized by the models studied, only one of them has this aspect formally defined. Applying a decision model allows an organization to address the risks associated with an investment in a product line. The direction here relies in investigating the potential options for reuse scenarios using decision analysis techniques, such as decision trees (Harrison et al., 2002). Decision trees can be used in this case to evaluate the future benefits to take uncertainty into account. Schmid (Schmid, 2003) was already investigated the use of decision trees to find out the influence of a certain strategy in determine the appropriated path of action for product lines investment.
- **Tool Implementation**. The application of a reuse cost model in an organization may be a difficult task if no automated support is provided (Krueger, 2007). Some directions in this sense have been approaching requirements definition and tool implementation to support the model

[19] Clements et. al. (Clements, 2005) suggests that this investigation can be performed through the “What-If” technique.

estimations (Mili et al, 2001[20]), (Clements et al., 2005[21]), (Lamine et al., 2005). In the case study a spreadsheet was used to calculate the cost and benefits factors. This tool was particularly important to automate the investment analysis activity, but it lacks on retrieving the historical data from the organization internal systems in an integrated way. A possible direction can explore the development of an integrated tool that acquires effort information from historical data and provide more usability to the users of the model. Clements et al. (Clements et al., 2005) suggest a list of features for a potential tool, including the display of graphs for each reuse scenario, the possibility to execute simulation over the input parameters, the possibility to accept a list of assumptions inherent to model (or to the scenarios, or yet, to the input parameters), and, the possibility to allow users to propose new scenarios and formulations that reflects those scenarios.

6.4. Academic Contributions

The knowledge acquired during this dissertation can be shared with the product line research community through the following publication:

- (Nóbrega et al., 2006) Nóbrega, J.; Almeida, E. S.; Meira, S. R. L., "**A Cost Framework Specification for Software Product Lines Scenarios"**, in the Sixth Workshop on Component-Based Development (WDBC), Recife, Brazil, 2006.

Besides the published paper, there are two additional papers in evaluation during the period when this dissertation was written:

- (Nóbrega et al., 2008a) Nóbrega, J.; Almeida E. S.; Meira, S. R. L., "**An Integrated Cost Model for Product Line Engineering**", in 34th Euromicro Conference on Software Engineering and Advanced Application, Parma, Italy, September 3-5, 2008 (in evaluation).
- (Nóbrega et al., 2008b) Nóbrega, J.; Almeida E. S.; Meira, S. R. L., "**An Industrial Case Study with an Integrated Cost Model for Software Product Lines**", in Simpósio Brasileiro de Componentes,

[20] http://www.csee.wvu.edu/reuseroi
[21] http://simple.sei.cmu.edu/

Arquitetura e Reutilização de Software 2008 (SBCARS 2008) (in evaluation).

6.5. Concluding Remarks

Product line engineering is not a new concept, **since Eli Whitney revolutionized the manufacturing of rifles using interchangeable** parts and Henry Ford did the same for automobiles, integrating this idea with an assembly line. For software development community, the product line engineering is emerging as a practical and important paradigm to solve the problems related with cost and schedule overruns. One key aspect to successful this approach is to identify costs and benefits that are associated with product family development.

In this sense, this work presented the Integrated Cost Model for Product Line Engineering (InCoME), which was based on an extensive review of available cost models by addressing their main features, weakness and strengths.

This model aims to perform an investment analysis to a product line through the estimation of costs and benefits associated with it. In addition, the model was evaluated in a real software development scenario, where the findings had indicated that it can generates cost estimations in an accurately way.

References

(Albrecht, 1979) Albrecht, A. J. **Measuring Application Development Productivity**, IBM Applications Development Symposium, Monterey, CA, 1979.

(Almeida et al., 2004) Almeida, E. S., Alvaro, A., Lucrédio, D., Garcia, V. C., Meira, S.R.L. **RiSE Project: Towards a Robust Framework for Software Reuse**, IEEE International Conference on Information Reuse and Integration (IRI), 2004, Las Vegas, USA, p. 48-53.

(Almeida, 2007) Almeida, E.S. **The RiSE Process for Domain Engineering**, Ph.D. Thesis, Federal University of Pernambuco, Recife, March, 2007.

(Alvaro et al., 2006) Alvaro, A., Almeida, E.S., Meira, S. R. L. **A Software Component Quality Model: A Preliminary Evaluation**, 32nd IEEE EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), Component-Based Software Engineering Track, Cavtat/Dubrovnik, Croatia, p. 28-35.

(Bandinelli et al., 1996) Bandinelli, S., Sagarduy, G., **A Unifying Framework for Reuse Economic Models**, Technical Report ESI-1996-REUSE03, European Software Institute, Bilbao, Spain, November, 1996.

(Barnes et al., 1991) Barnes, B., Bollinger, T. **Making Software Reuse Cost Effective**. IEEE Software (1 1991), pp. 13-24**.**

(Barros, 2001) Barros, M.O. **Project Management based on Scenarios: A Dynamic Modeling and Simulation Approach** (in Portuguese), Ph.D. Thesis, Federal University of Rio de Janeiro, December, 2001, pp. 249.

(Basili et al., 1994) Basili, V.R., Caldiera, G., Rombach, H.D. **The Goal Question Metric Approach**, Encyclopedia of Software Engineering, Vol. II, September, 1994, pp. 528-532.

(Bayer et al., 1999) J. Bayer, Flege, O., Knauber, P., Laqua, R., Muthig, D., Schmid, K., Widen, T., DeBaud, J. **PuLSE: A Methodology to Develop Software Product Lines**, Proc. 5th Symp. Software Reusability (SSR'99), ACM Press, New York, 1999, pp.122–131.

(Böckle et al., 2004) Böckle, G., Clements, P., McGregor, J.D., Muthig, D., Schmid, K. **Calculating ROI for Software Product Lines**, IEEE Software, Vol. 21, No. 03, May/June, 2004, pp. 23-31.

(Boehm, 1981) Boehm, B.W. **Software Engineering Economics**, Prentice-Hall, Englewood Cliffs, NJ.

(Boehm et al., 1995) Boehm, B.W., Clark, B., Horowitz, E., Westland, C., Madachy, R. Selby, R. **Cost Models for Future Software Lifecycle Processes: COCOMO 2.0**, Annals of Software Engineering 1, 57–94.

(Boehm et al., 2003) Boehm, B., Huang, L.G. **Value-based Software Engineering: a Case Study**, IEEE Computer, Vol. 36, No. 03, March, 2003, pp. 33-41.

(Boehm et al., 2004) Boehm, B., Winsor, B.A, Ray, M., Yang, Y. **A Software Product Line Life Cycle Cost Estimation Model**, 156-164. Proceedings of the 2004 International Symposium on Empirical Software Engineering. Redondo Beach, CA, August 19-20, 2004. Los Alamitos, CA: IEEE Computer Society, 2004.

(Boehm, 2006) Boehm, B., Hoh, P., Baik, J., Kim, S., Yang, Y. **A Quality-based Cost Estimation Model for the Product Line Life**

Cycle, Communications of the ACM, December 2006/Vol. 49, No. 12.

(Bollinger et al., 1990) Bollinger, T.B., Pfleeger, S.L. **Economics of Reuse: Issues and Alternatives**, Information and Software Technology, Volume 32, Issue 10 (December 1990) Pages: 643 – 652.

(Brealey et al., 1996) Brealey, R., Myers, S. **Principles of Corporate Finance**, 5th Edn., McGraw-Hill, New York, NY.

(Brito et al., 2007) Brito, K.S., Garcia, V.C., Lucrédio, D.A., Almeida, E. S., Meira, S.R.L. **LIFT: Reusing Knowledge from Legacy Systems**, Brazilian Symposium on Software Components, Architectures and Reuse, Campinas, Brazil, 2007.

(Brooks, 1995) Brooks, F.P. **The Mythical Man-Month: Essays on Software Engineering, 20th Anniversary Edition**, Addison-Wesley Professional, 1st edition (August 2, 1995).

(Brownsword et al., 1996) Brownsword, L., Clements, P. **A Case Study in Successful Product Line Development**, Tech. Report CMU/SEI-96-TR-016, Software Eng. Inst., Carnegie Mellon Univ., Pittsburgh, 1996.

(Burégio, 2006) Burégio, V.A.A. **Specification, Design, and Implementation of a Reuse Repository**, Msc. Dissertation, Federal University of Pernambuco, Recife, August, 2006.

(Caldiera et al., 1991) Caldiera, G., Basili, V. **Identifying and Qualifying Reusable Software Components**, IEEE Computer 24, 2, pp 61–70.

(Chulani, et al. 1999) Chulani, S., Boehm, B., Steece, B. **Bayesian Analysis of Empirical Software Engineering Cost Models**, IEEE Transactions on Software Engineering 25, 4 (1999), 573–583.

(Clements et al., 2001) Clements, P., Northrop, L.M. **Software Product Lines**, Addison-Wesley, 2001.

(Clements et al., 2002) Clements, P., Krueger, C. **Initiating Software Product Lines - Point/Counterpoint: Being Proactive Pays Off**, IEEE Software 19, 4 (July/August 2002).

(Clements, 2002) Clements, P. **Being Proactive Pays Off**, IEEE Software, July/August 2002, pp 28-30.

(Clements et al., 2004) Clements, P., Northrop, L. **A Framework for Software Product Line Practice**, Version 4.2. Consulted in November 2007 in http://www.sei.cmu.edu/productlines/framework.html (2004).

(Clements et al., 2005) Clements, P.C., McGregor, J.D., Cohen, S.G. **The Structured Intuitive Model for Product Line Economics (SIMPLE),** Technical Report, CMU/SEI-2005-TR-003, ESC-TR-2005-003.

(COCOTS, 1999) **COCOTS Technical Report**, Center for Software Engineering, University of Southern California, Los Angeles, CA.

(Cohen, 2003), Cohen, S. **Technical Note** CMU/SEI-2003-TN-017, 2003.

(Coriat, 2000) Coriat, M. **The SPLIT Method**, Proc. 1st Software Product Line Conf, 2000.

(Coulange, 1998) Coulange, B. **Software Reuse**, Springer, London, UK, 1998.

(Devanbu et al., 1996) Devanbu, P., Karstu, S., Melo, W., Thomas, W. **Analytical and Empirical Evaluation of Software Reuse Metrics**, In Proceedings of International Conference on Software Engineering, Berlin, Germany, IEEE Press, New York.

(Erdogmus et al., 2004) Erdogmus, H., Favaro, J. M., Strigel, W. **Introduction: Return on Investment**, IEEE Software, Vol. 21, No. 03, May/June, 2004, pp. 18-22.

(Ezran et al., 2002) Ezran, M., Morisio, M., Tully, C. **Practical Software Reuse**, Springer, 2002, pp. 374.

(Fafchamps, 1994) Fafchamps, D. **Organizational Factors and Software Reuse**, IEEE Software 11, 5, 31–41.

(Favaro, 1996) Favaro, J. **A Comparison of Approaches to Reuse Investment Analysis**, The Fourth International Conference on Software Reuse, IEEE Computer Society Press, Orlando, USA, April, 1996, pp. 136-145.

(Favaro et al., 1998) Favaro, J., Favaro, K., Favaro, P.F. **Value Based Software Reuse Investment**, Annals of Software Engineering 5, 5–52.

(Feynman, 1985) Feynman, R., **Surely You're Joking Mr. Feynman!**, Bantam, 1985.

(Frakes et al., 1994a) Frakes, W.B., Terry, C. **Reuse Level Metrics**, In Proceedings of the 3rd International Conference on Software Reuse, Rio de Janeiro, Brazil, November, pp. 139–148.

(Frakes et al., 1994b) Frakes, W.B., Isoda, S. **Success Factors of Systematic Software Reuse**, IEEE Software, Vol. 12, No. 01,September, 1994, pp. 15-19.

(Frakes et al., 1996) Frakes, W.B., Terry, C. **Software reuse: Metrics and Models**, ACM Computing Surveys, Vol. 28, No. 02, ACM Press, Jun, 1996, pp. 415-435.

(Gaffney et al., 1992) Gaffney, J.E., Cruickschank, R.D. **A General Economics Model of Software Reuse**, In Proceedings of the International Conference on Software Engineering, Melbourne, Australia, May, pp. 327–337.

(Garcia et al., 2007) Garcia, V.C., Lucrédio, D., Durão, F.A., Santos, E.C.R., Almeida, E.S., Fortes, R.P.M., Meira, S.R.L. **From Specification to the Experimentation: A Software Component Search Engine Architecture**, 9th International Symposium on Component-Based Software Engineering (CBSE), Sweden, Lecture Notes in Computer Science (LNCS), Springer-Verlag, p. 82-97.

(Gibbs, 1994) Gibbs, W.W. **Software's Chronic Crisis**, Scientific American 271 (3), 86-95, 1994.

(Guerrieri et al., 1988) Guerrieri, E., Lori A.L., Theodore, B.R. **An Acquisition Strategy for Populating a Software Reuse Library**, National Conference on Software Reusability, Washington D.C., July 19-20, 1989.

(Harrison et al., 2002) Harrison, W., Erdogmus, H., Sullivan, K., Boehm, B., Reifer, D., **Software Engineering Economics: Background, Current Practices and Future Directions**, Tutorial 3 at the International Conference on Software Engineering, 2002.

(Hartmann et al., 2006) Hartmann, S., Frigg, R. **Models in Science**, In The Stanford Encyclopedia of Philosophy. 02/2006. Edited by Zalta, E. N. Stanford University, 2006.

(Kain, 1994) Kain, B.J. **Measuring the ROI of Reuse**, Object Magazine 4, 3, pp. 48–54.

(Karner, 1993) Karner, G. **Metrics for Objectory**, Diploma thesis, University of Linköping, Sweden, No. LiTHIDA-Ex-9344:21. December 1993.

(Kazman et al., 2002) Kazman, R., Asundi, J., Klein, M. **Making Architecture Design Decisions: An Economic Approach**, (CMU/SEI-2002-TR-035, ADA408740). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2002.

(Krueger, 1992) Krueger, C.W. **Software Reuse**, ACM Computing Surveys, Vol. 24, No. 02, June, 1992, pp. 131-183.

(Krueger, 2007) Krueger, C.W. **The New Generation of Software Reuse Tools for Software Product Line Lifecycle Engineering and Management**, RISS 2007, 1st RiSE Summer School on Software Reuse, Recife, Brazil, 2007.

(Lamine et al., 2005) Lamine, S., Jilani, L., Ghezala H. **A Software Cost Estimation Model for Product Line Engineering: SoCoEMo-PLE**, Proceedings, The 2005 International

MultiConference in Computer Science and Computer Engineering, Software Engineering Research and Practice conference SERP 2005. Las Vegas Nevada USA (2005).

(Learch, 1997) Leach, R.J. **Software Reuse**, McGraw Hill, New York, 1997.

(Leung et al., 2001) Leung, H., Fan, Z. **Software Cost Estimation**, Handbook of Software Engineering and Knowledge Engineering, Vol. II, 2001, pp. 14.

(Lim, 1994) Lim, W.C. **Effects of Reuse on Quality, Productivity and Economics**, IEEE Software 11, 5, 23–30.

(Lim, 1996) Lim, W.C. **Reuse Economics: A Comparison of Seventeen Models and Directions for Future Research**, The 4th International Conference on Software Reuse, Orlando, USA, April, 1996, pp. 41-51.

(Lim, 1998) Lim, W.C. **Managing Software Reuse: A Comprehensive Guide to Strategically Reengineering the Organization for Reusable Components**. Prentice-Hall, Upper Saddle River, NJ, 1998.

(Lisboa et al., 2007) Lisboa, L.B., Garcia, V.C., Almeida, E.S., Meira, S.L. **ToolDAy: A Process-Centered Domain Analysis Tool**, 21st Brazilian Symposium on Software Engineering, Tools Session, João Pessoa,Brazil, 2007.

(Malan et al., 1993) Malan, R., Wentzel, K. **Economics of Reuse, Revisited**, Technical Report HPL-93-31, Hewlett Packard Laboratories.

(Malvin et al., 1986) Malvin, K., Whitlock, PA. **Monte Carlo Methods**, New York: John Wiley & Sons, 1986.

(Manhattan, 2004) History – Los Alamos – Oversight Committee Formed **The Manhattan Project Heritage Preservation Association**, http://www.childrenofthemanhattanproject.org/HISTORY/H-06c12.htm, 2004. URL acessed in 10/12/2007.

(Margano et al., 1992) Margano, J., Rhoads, T.E. **Software Reuse Economics: Cost Benefit Analysis on a Large Scale Ada**

Project, In Proceedings of International Conference on Software Engineering, Melbourne, Australia, May, 1992, pp. 338–348.

(Mascena, 2006) Mascena, J. **ADMIRE: Asset Development Metric-based Integrated Reuse Environment**, M.Sc. Dissertation, Federal University of Pernambuco, Recife, May, 2006.

(McGregor et al., 2002) McGregor, J., Northrop, L., Jarrad, S., Pohl, K. **Initiating Software Product Lines**, IEEE Software, July-August 2002, pp. 24-27.

(McIlroy, 1968) McIlroy, M.D. **Mass Produced Software Components**, NATO Software Engineering Conference Report, Garmisch, Germany, October, 1968, pp. 79-85.

(Mili, 1996) Mili, R. **Return on Investment of Reusable Components: Analytical and Empirical Approaches**, Technical Report, University of Ottawa, Ottawa, ON, Canada, 1996.

(Mili et al., 1999) Mili, A., Fowler, S., Gottumukkala, R., Zhang, L. **Software Reuse Cost Estimation**, Technical Report, CSEE Department, West Virginia University.

(Mili et al., 2000) Mili, A., Chmiel, S. F., Gottumukkala, R., Zhang, L. **An Integrated Cost Model for Software Reuse**, The 22nd International Conference on Software Engineering, Limerick, Ireland, ACM Press, June, 2000, pp. 157-166.

(Mili et al., 2001) Mili, A., Chmiel, S.F., Gottumukkala, R., Zhang, L. **Managing Software Reuse Economics: An Integrated ROI-based Model**, Annals of Software Engineering 11, 175–218, 2001.

(Mili et al., 2002) Mili, H., Mili, A., Yacoub, S., Addy, E. **Reuse-Based Software Engineering**, Willey, 2002, pp. 636.

(Morizio et al., 2002) Morisio, M., Ezran, M., Tully, C. **Success and Failure Factors in Software Reuse**, IEEE Transactions on Software Engineering, Vol. 28, No. 04, April, 2002, pp. 340-357.

(Muthig et al., 2006) Muthig, D., Ganesan, D., Yoshimura, K., **Predicting Return-on-Investment for Product Line Generations**, 10th International Software Product Line Conference (SPLC'06).

(Naur et al., 1969) Naur, P., Randell, B. **Software Engineering: Report of a Conference Sponsored by the NATO Science Committee**, Garmisch, Germany, 7-11 October 1968, Brussels, Scientific Affairs Division, NATO. (Eds.). 1969.

(Nazareth et al., 2004) Nazareth, D.L., Rothenberger, M.A. **Assessing the Cost-Effectiveness of Software Reuse: a Model for Planned Reuse**, Journal of Systems and Software, Vol. 73, No. 02, October, 2004, pp. 245-255.

(Nóbrega et al., 2006) Nóbrega, J.P. Almeida, E.S., Meira. S.R.L. A Cost Framework Specification for Software Product Lines Scenarios, WDBC 2006, Recife, pp. 30-37.

(Nóbrega et al., 2008a) Nóbrega, J.; Almeida E. S.; Meira, S. R. L., **An Integrated Cost Model for Product Line Engineering**, in 34th Euromicro Conference on Software Engineering and Advanced Application, Parma, Italy, September 3-5, 2008 (in evaluation).

(Nóbrega et al., 2008b) Nóbrega, J.; Almeida E. S.; Meira, S. R. L., **An Industrial Case Study with an Integrated Cost Model for Software Product Lines**, in Simpósio Brasileiro de Componentes, Arquitetura e Reutilização de Software 2008 (SBCARS 2008) (in evaluation).

(Northrop, 2002) Northrop, L.M., **SEI's Software Product Line Tenets**, IEEE Software, July/August 2002, pp. 32-40.

(Peterson, 2004) Peterson, D.R. **Economics of software Product Lines**, Lecture Notes in Computer Science, Springer Berlin / Heidelberg Volume 3014, pp. 381-402, 2004.

(Philips, 2000) Philips America **CoPAM: A Component-Oriented Platform Architecting Method Family for Product**

Family Engineering, Proc. 1st Software Product Line Conf, 2000.

(Pllana, 2000) Pllana, S. **History of Monte Carlo Method**, http://www.geocities.com/CollegePark/Quad/2435/index.html, August 2000. URL acessed in 10/12/2007.

(Poulin et al., 1993) Poulin, J.S., Caruso, J.M., Handcock, D.R. **The Business Case for Software Reuse**. IBM Syst. J. 32, 4, 567–594.

(Poulin, 1997a) Poulin, J.S. **Measuring Software Reuse: Principles, Practices, and Economic Models**, ISBN 0-201-63413-9, Addison-Wesley, Reading, MA.

(Poulin, 1997b) Poulin, J.S. **The Economics of Product Line Development**, Available in URL: http://home.stny.rr.com/jeffreypoulin/Papers/IJAST97/ijast97.html, Consulted in October, 2007.

(Poulin, 2006) Poulin, J.S. **The Business Case for Software Reuse: Reuse Metrics, Economic Models, Organizational Issues, and Case Studies**, Tutorial Notes, Torino, Italy, June, 2006.

(Pressman, 2004) Pressman, R.L., **Software Engineering: A Practitioner's Approach**, McGraw-Hill, ISBN 007301933X / 9780073019338, 2004.

(Rothenberger et al., 2004) Rothenberger, M.A., Nazareth, D.L. **A Cost-Benefit Model for Systematic Software Reuse**, ECIS 2002, June 6–8, Gdańsk, Poland.

(Sametinger, 1997) Sametinger, J., **Software Engineering with Reusable Components**, Springer- Verlag, 1997, pp.275.

(Schmid, 2002) Schmid, K. **Reuse Economics from a Product Line Point of View**, Seventh International Conference on Software Reuse, International Workshop on Reuse Economics, 2002.

(Schmid, 2003) Schmid, K., **Integrated Cost and Investment Models for Product Family Development**, Kaiserslautern, 2003, VIII, 80 pp. : Ill., Lit.IESE-Report, 067.03/E.

(Schimsky, 1992) Schimsky, D. **Software Reuse – Some Realities**, Vitro Tech. Journal 10, 1, 47–57.

(Sharp, 2000) Sharp, D. **Component-Based Product Line Development of Avionics Software**, Proc. 1st Software Product Line Conf., pp. 353–369.

(SPLC1, 2000) Kluwer Academic Publishers, Boston, 2000, pp. 147–166.

(Thiel et al., 2000) Thiel, S., Peruzzi, F. **Starting a Product Line Approach for an Envisioned Market**, Proc. 1st Software Product Line Conf., 2000.

(Toft, 2000) Toft, P. **A Cooperative Model for Cross-Divisional Product Development for a Software Product Line**, Proc. 1st Software Product Line Conf, 2000.

(Tomer et al., 2004) Tomer, A., Goldin, L., Kuflik, T., Kimchi, E., Schach, S.R. **Evaluating Software Reuse Alternatives: A Model and Its Application to an Industrial Case Study**, IEEE Transactions on Software, Vol. 30, No. 9, September 2004.

(Trigeorgis, 1996) Trigeorgis, L. **Real Options**, The MIT Press, Cambridge, MA, 1996.

(Trivedi, 2001) Trivedi, K.S. **Probability and Statistics with Reliability,Queuing, and Computer Science Applications**, John Wiley and Sons, New York, 2001. ISBN number 0-471-33341-7.

(Vanderlei et al., 2006) Vanderlei, T.A., Durão, F.A., Martins, A.C., Garcia, V.C., Almeida, E.S., Meira, S.R.L. **A Classification Mechanism for Search and Retrieval Software Components**, 22nd Annual ACM Symposium on Applied Computing (SAC), Information Retrieval Track, Seul, Korea.

(Verhoef, 2005) Verhoef, C. **Quantifying the Value of IT Investments**, Science of Computer Programming 56 (2005), pp. 315–342.

(Wiles et al., 1998) Wiles, E., Bott, F. **Eight Steps to Your Own Economic Model of Software Reuse**. In Proceedings of the European Reuse Workshop 98 (Madrid, Spain, Nov. 1998), pp. 123-127

(Wiles, 1999) Wiles, E. **Economics Models of Software Reuse: A Survey, Comparison and Partial Validation**, Technical Report, version 2.1, April, 1999, pp. 49.

(Wohlin et al., 2000) Wohlin, C., Runeson, P., Host, M., Ohlsson, M.C., Regnell, B., Wesslén, A. **Experimentation in Software Engineering: An Introduction**, Kluwer Academic Publishers, 2000, pp. 204.

Appendix A. Monte Carlo Simulation

The Monte Carlo Simulation method provides approximate solutions to a variety of mathematical problems by performing statistical sampling experiments on a computer (Malvin et al., 1986). The method applies to problems with absolutely no probabilistic content as well as to those with inherent probabilistic structure.

This method is largely used for iteratively evaluating a deterministic model using sets of random numbers as inputs. It is often used when the model is complex, nonlinear, or involves more than just a couple uncertain parameters. It also can be described as a method to analyze the ***uncertainty propagation***, where the goal is to determine how *random variation*, *lack of knowledge* or *error* affects the *sensitivity*, *performance* or *reliability* of the system that is being modeled. Monte Carlo simulation is categorized as a sampling method because its inputs are randomly generated from *probability distributions* to simulate the process of sampling from an actual population. To perform simulation using this method is necessary to choose a distribution for the inputs that most closely matches data we already have, or best represents our current state of knowledge. The data generated from the simulation can be represented as probability distributions, i.e. *histograms*, or converted to errors bars, reliability predictions, tolerance zones, and confidence intervals.

Among the frequently used distribution in Monte Carlo simulation, we can highlight the following:

- **Normal/Gaussian Distribution**. Continuous distribution applied in situations where the mean and the standard deviation are given and the mean represents the most probable value of the uncertain variable. It is symmetrical around the mean and is not bounded.

- **Lognormal Distribution**. Continuous distribution specified by mean and standard deviation. This is appropriate for a variable ranging from zero to infinity, with positive skewness and with normally distributed natural algorithm.
- **Triangular Distribution**. Continuous distribution with fixed minimum and maximum values. It is bounded by the minimum and maximum values and can be either symmetrical (the most probable value is equals to the mean and to the median) or asymmetrical.
- **Uniform Distribution**. Continuous distribution bounded by known minimum and maximum values. In opposition to the triangular distribution, the likelihood of the occurrence of the values between the minimum and maximum is the same.
- **Exponential Distribution**. Continuous distribution used to illustrate the time between independent occurrences when the rate of them is known.

Consider that we have a real-valued function g(X) with probability frequency function P(x), if x is discrete, or probability density function f(x), if x is continuous. Then we can define the expected value of g(X) in discrete and continuous terms, as the following:

$$E(g(X)) = \sum_{-\infty}^{+\infty} g(x)P(X), \text{ where } P(x) > 0 \text{ and } \sum_{-\infty}^{+\infty} P(x) = 1$$

$$E(g(X)) = \int_{-\infty}^{+\infty} g(x)f(x)dx \text{, where } f(x) > 0 \text{ and } \int_{-\infty}^{+\infty} f(x)dx = 1$$

Next, we make *n* random drawings of *X* $(x_1, \ldots, x_n)$, called trial runs or simulation runs, calculate $g(x_1), \ldots, g(x_n)$ and find the mean of *g(x)* of the sample:

$\overline{g_n(x)} = \frac{1}{n}\sum_{i=1}^{n} g(x_i)$, which represents the final simulated value of *E(g(X))*.

As $n \to \infty$, $\overline{g_n(X)} \to E(g(X))$, thus we are now able to compute the dispersion around the estimate mean with the unbiased variance of $\overline{g_n(X)}$:

$$Var\left(\overline{g_n(X)}\right) = \frac{1}{n}\frac{1}{(n-1)}\sum_{i=1}^{n}\left(g(x_i) - \overline{g_n(x)}\right)^2$$

In summary, to perform a simulation using the Monte Carlo method is necessary to follow the five simple steps below:

Step1. Create a parametric model, $y = f(x_1, x_2, \ldots, x_n)$.

Step2. Generate a set of random inputs, $x_{i_1}, x_{i_2}, \ldots, x_{i_n}$.

Step3. Evaluate the model and store the results as y_i.

Step4. Repeat steps 2 and 3 for *i=1 to n*.

Step5. Analyze the results using histograms, summary statistics, confidence intervals, and so on.

Dissertação de Mestrado apresentada por **Jarley Palmeira Nóbrega** à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título **"An Integrated Cost Model for Software Product Line Engineering",** orientada pelo **Prof. Silvio Romero de Lemos Meira** e aprovada pela Banca Examinadora formada pelos professores:

[illegible]

Prof. Alexandre Marcos Lins de Vasconcelos
Centro de Informática / UFPE

[illegible]

Prof. Jones Oliveira de Albuquerque
Deptº de Estatística e Informática / UFRPE

[illegible]

Prof. Silvio Romero de Lemos Meira
Centro de Informática / UFPE

Visto e permitida a impressão.
Recife, 14 de março de 2008.

Francisco de Assis Tenório de Carvalho

Prof. FRANCISCO DE ASSIS TENÓRIO DE CARVALHO
Coordenador da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.