2005

Generalization of an integrated cost model and extensions to COTS, PLE and TTM

Lin Yang
West Virginia University

Generalization of an Integrated Cost Model and Extensions to COTS, PLE and TTM

Lin Yang

Dissertation submitted to the
College of Engineering and Mineral Resources
at West Virginia University
In partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science

Ramana Reddy, Ph.D., Chair
Ali Mili, Ph.D.
Jim Mooney, Ph.D.
Ming-jeng Hwang, Ph.D.
Sumitra Reddy, Ph.D.

Lane Department of Computer Science and Electrical Engineering

Morgantown, West Virginia
2005

Keywords: Reuse, COTS, PLE, TTM, NPV, and ROI

Abstract

Generalization of an Integrated Cost Model and Extensions to COTS, PLE, and TTM

Lin Yang

There have been existing software reuse cost models related to estimating costs of software reuse, for example, COCOMO II, COCOTS, and so on. Chmiel's model [Chmiel 2000] is a generalization of these cost models. This model is different from others in that the decisions are composed of four levels and treats reuse projects from a point of view of long term run. Each level corresponds one engineering cycle. Each level is a decision making process based on calculation of NPV, ROI, and other economic indices. Reuse investment decisions are made on different levels from the corporate to the programming.

Chmiel's model doesn't cover Commercial-Off-The-Shelf (COTS), Product Line Engineering (PLE) and benefits due to shortened Time-To-Market (TTM) and can only deal with internal traffic since the model assumes all of the reusable components are built from scratch in house. For example, Extra efforts caused by the use of COTS, assessment, tailoring, glue code and COTS volatility, are not covered in this model. And this model doesn't treat the benefits of TTM, for example, business performance.

By extending Chmiel's model, the new model is applied to CBSE (Component-Based Software Engineering), COTS reuse systems and PLE. In addition, attempts of quantifying benefits of shortened TTM are made within this study and a TTM submodel is developed to cover this issue.

This study also addresses the issue to analyze and optimize corporate Return On Investment (ROI). The rational is that optimizing (maximizing) the corporate ROI under the condition that all other ROI's are positive. By designing an algorithm and applying it to the data, this study discovers the method how to make the maximized value of corporate ROI.

And this model is supported through a tool based on the model rationale. Users to this tool are corporate management and development engineers. The supporting tool has user-friendly interface and allows users to input values of related parameters. A detailed report is produced, which is composed of details about costs and benefits, final Net Present Value (NPV) and ROI of each cycle.

Acknowledgements

It'the eighth year in this country since I came to US in 1997. After obtaining degrees of bachelor and master in computer science at WVU, I began the Ph.D. program from January 2001.

First I would like to say thanks to Dr. Ali Mili. He assigned me this topic and continued to support me after he left WVU for NJIT. Without his advising, I can not go through it.

I also want to express my thanks to Dr. Ramana Reddy. After Dr. Mili's leaving, Dr. Reddy took the position being my committee chair so that I could finish this project properly.

I would like to thank my father Weixi Yang. He suggested me to continue to work on Ph.D. program after I got my master five years ago. He always give me great support and help.

Lastly, I would like to say my thanks to Dr. Kurishinkal Cleetus. He has been funding me since 2004 so I don't have to worry about the funding issue.

Thanks all for great support though I know these simple words cannot express my appreciation and feeling.

Table of Contents

Table of Figures

Table of Tables

Chapter 1 Introduction

1.1 Software reuse

Nowadays, software reuse has been playing a more and more important role in software industry. At the same time, new challenging issues have come up. How to estimate the costs and benefits in the fields of COTS and PLE? How to quantify the real world benefits due to these reuse activities? For example, besides the gains in quality and productivities, how to quantify the business performance such as shortened TTM is another primary concern in software reuse research.

Software reuse is the process of implementing or updating software systems using existing software assets [DOD 1996]. Assets can be architectures, source code, data, designs, documentation, estimates, human interfaces, plans, requirements, and test cases. These assets can be reused in different contexts. Software reuse may occur within a software system across similar systems, or in widely different systems [Hudson 2001]. Basically, the goal of reuse is to use as much software data as possible from previous development efforts in order to reduce time, cost, and risks associated with redevelopment.

The definition of software reuse from SEI is as followed:

The purpose of software reuse is to use existing software assets in new contexts, either in existing systems or in new applications, to increase development productivity and increase product quality. Cumulative, long-term benefits of reuse to an organization require the evaluation of an organization's reuse potential, creation or acquisition of reusable assets, asset management, and asset use. An organization's software development process contains reuse practices and activities aimed at exploiting the organization's software assets and ensuring the maintenance of existing assets and the addition of new assets with high reuse potential. Reusable software assets may consist of any software artifacts such as requirements, design, code, documentation, plans, procedures, and manuals [SEI 2004].

Software reuse is a consequence of good product design. Full and effective software reuse can only be achieved if much attention is paid to reuse at the requirements analysis stage of the component developing life cycle. Domain analysis needs to consider both the reusability of components in later stages and the reuse of existing analysis deliverables. In order to make a component reusable in the future, more efforts are needed, for example, more efforts on design, domain analysis, and implementation. The design phase of the component should be object-oriented, focusing in particular on encapsulation to ensure that objects can be reused. Standard good programming practices, such as maximal cohesion, modular design, and minimal coupling between code modules, should be employed to help make software components more reusable.

There are three kinds of improvements by taking software reuse. These advantages that software reuse can bring are recognized by software engineering community.

- Productivity: That means less code needs to be written by using reusable components instead of developing the same functionalities from scratch. Under this assumption, that also means development time is shortened and development costs are decreased.

- Quality:

  (1) Reusable assets proven in one project can be carried through to the next.
  (2) In addition, less code means more opportunities for testing. Less code makes possible for software engineers to spend more efforts to test and validate.

- Business performance: Software reuse can also result in lower costs, shorter time to market, and customer satisfaction. That makes corporation get more market gains in market share and market sales by delivering the products to market earlier. Especially, there are no competing products due to earlier TTM than competitors. In addition, by satisfying customer and industry, the possibility of rising on stock market has greatly increased.

So far most existing cost models only cover the first two issues and give their solutions to estimate the costs and benefits in productivity and quality. There is no such a model, which formally and completely deals with the third one: business performance. Business performance is the most concern for corporate managers. Besides the gains in productivity and quality obviously, as the decision makers, they are more interested in how much benefit using reusable components can bring.

After reviewing and analyzing the literature of software reuse, this research finds that the definition of software reuse has been widening over the past decades. On technical level, software reuse not only deals with the use of existing reusable components, but also the implementation and creation of them. On a higher level of reuse economics which corporate management is more interested in, software reuse also addresses the decision making based on economics of the whole life cycle.

Based on the current situation of cost models, besides covering the main software reuse fields nowadays: CBSE and COTS, this study also gives solution to analyze and quantify the benefits due to shortened TTM.

1.1.1 Advantages of software reuse

General improvements incurred by software reuse are discussed in last section. Based on these, there are detailed advantages by taking software reuse, which can describe these improvements.

- Initial Development Savings
- Reduced cost to design the component
- Reduced cost to implement the component
- Reduced cost to test the component

- Reduced cost to document the component
- Ongoing maintenance savings
- Reduced cost to fix defects in the component
- Reduced cost to enhance the software product of next generation
- Shortened time-to-market

One of the hidden benefits of reuse is that development must follow good software engineering practices and that prototyped projects tend to be designed and developed sooner due to the wide usage. This earlier development allows the application to enter the market sooner and reap the additional benefits that result from this earlier marketing of the application. If the application was built without reuse it would run the risk of having another application take over the market and losing the additional benefits gain with early marketing [Neilsen 2000].

Software reuse can greatly reduce development costs. Toshiba Software Factory has a set of statistics to show a comparison of cost reductions realized by different methods.

Methods to achieve productivity and quality	Cost reduction
Reuse	52.1%
Improve procedures and environments	18.0%
Application of new software tools	9.7%
Improvement on functional descriptions	7.2%
Use of higher level languages	6.3%

Table 1: Comparison of cost reductions [Neilsen 2000]

Reuse can save costs on testing, verification and validation. Especially when purchasing COTS components, the development team knows the components are tested and has functionalities as specifications indicate. Also, cost on documentation decreases due to the reusable documentation. The Product Line Engineering makes reduced cost to enhance the software product of next generation, since some components have property to be reused and they are validated and verified. These make these components be used in the next generation [SEI 2003].

Reuse improves productivity because the life cycle now requires less effort to obtain the same outcome and reduces the amount of time and labor needed to develop and maintain software products. Furthermore, because work products are used multiple times, the defect fixes from each reuse accumulate**,** resulting in higher quality. The defining characteristic of "good reuse is not the reuse of software per se but the reuse of human problem solving [Barnes 1991]." Reuse allows for this expertise to be extended to projects/companies that cannot afford such expertise, thereby improving productivity. The producer and consumer of reusable assets can avoid duplication of effort by centrally maintaining the reuse components, managing their evolution,

and propagating upgrades using reuse technology. Furthermore, using reusable black box components allows further enhancement and correction quicker and at a lower cost [Malan 1993].

The following corporations reported the improvements by taking reuse. DEC had cycle time 67-80% lower at reuse rate of 50-80%. Fujitsu experienced a proportion of projects on schedule increased from 20-70% and effort to customize package reduced from 30 person months to 4 days. Hewlett-Packard had defects up to 76% lower, productivity up to 57% higher and time to market 42% lower. NEC made productivity 6.7 time higher and quality 2.8 time better. Raytheon had a productivity of 50% higher at reuse level 60%. Toshiba had defects 20-30% lower at reuse level 60% [Neilsen 2000].

Another big benefit is that software reuse can improve a product's time to a market. If the product reaches the market earlier, then there will be more revenue. Software reuse can save development time and thus the product can be delivered earlier than if all the software is written from scratch.

Here are some statistics that can show the comparison between earlier and later TTM.

- Projects on time and 50% over budget earn 4% profit over 4 years [Market Forecasters].
- Projects 6 months late and on budget earn 33% less profit over 5 years [McKinsey].

Herbsleb and Anita Carleton states that the time to market actually suffered for the first year or so as the reusable components were being developed. During this period, the development time is delayed due to extra efforts spent to pay more attention to design and domain analysis and implement reusable components. After two-three years, the time to market dropped very rapidly since it’s the time to gain benefits from reuse. These reusable components can be used in application engineering and thus shorten time to market. In the end, the time to market will be shortened. The yearly reduction in time to market has a range of 15%-23% and a median of 19%. For all software products, improved quality through reuse reduces costs of maintenance [Griss 1993].

1.1.2 Disadvantages of software reuse

However, software developers have to pay extra costs and risks that software reuse brings besides the advantages. There are also existing disadvantages.

- More upfront investment
- More risk on the future
- It can end up costing more, if not done properly
- It can induce errors, if not done properly
- It must be used cautiously in safety-critical domains

Software reuse usually results in more initial costs. For example, purchasing COTS can increase costs. In order to get the reusable components, the project managers probably have to purchase,

for example, COTS. That means more upfront investment. And if the corporation decides to develop reusable components from scratch instead of buying or using existing, the development process will cost more than that of developing normal application.

There is also safety issue in safety-critical system. In this kind of system, safety is the most important issue. Sometimes inappropriate use of reusable components can result in system crash. Using reusable components should be cautious since development team has to rely on some components that are not implemented by itself but through a purchase. For example, usually the source code of COTS components is not accessible to users. In this situation, using reusable assets could also bring more risk in the future. If the development team doesn't have the source code of the components, it could take more time and cost to do preliminary steps for sure of safety.

And if the reusable components are not used properly, it can result in more errors. For example, if a component is modified and then used, that could probably cause more problems if validation and verification are not conducted after the modification.

Another disadvantage is the high costs of the reuse itself. For example, library insertion and acquisition, and development of reusable components make reuse more expensive comparing not using or developing reusable components. If these cost factors are not processed properly, the costs of using reusable components can outweigh the benefits.

1.2 Problem statement and objectives

There have been existing software reuse cost models related to estimating costs of software reuse, for example, COCOMO II, COCOTS, and Chmiel's. This research has found that there does not exist an effective and validated one which incorporates both the technical and non-technical facets of software reuse and showing its impact on time to market. Some of these models are immature and only in experiment phase. And some of them can only cover partially. For example, COCOTS can only give solutions to the initial phase and cannot treat a long term running reuse project. This study discovers that the critical problem in today's practice of software reuse is there is no such a model to conceptualize, define and develop necessary details to support a valid COTS software use process mode and PLE based on software economics. In addition, there is no such a model, which covers COTS, PLE, and TTM properly and completely.

Chmiel's model [Chmiel 2000] is a generalization of these existing reuse cost models. The idea of Chmiel's model is that the decisions are composed of four levels and treats reuse projects from a point of view of long term run. Each level corresponds one engineering cycle. Each level is a decision making process based on calculation of NPV, ROI, and other economic indices. Reuse investment decisions are made on different levels from the corporate to the programming.

- At the corporate level, management determines if the reuse investment will be worthwhile.

- At the managerial level, domain engineering managers determine which assets that have good reuse potential and create them. And the application managers determine the worthiness of incorporating reusable assets into a project.
- At the programming level, named Component Engineering Cycle, programmers implement reusable assets based on the decision from its upper level, domain level.

These decisions of different levels are reflected in the following 4 investment cycles:

- Corporate investment cycle.
- Domain engineering investment cycle.
- Application engineering investment cycle.
- Component engineering investment cycle.

This model doesn't cover COTS and PLE and benefits due to shortened TTM and can only deal with internal traffic since the model assumes all of the reusable components are built from scratch in house. For example, Extra efforts caused by the use of COTS, assessment, tailoring, glue code and COTS volatility, are not covered in this model. And this model doesn't treat the benefits of TTM, for example, business performance.

Based on Chmiel's model, this research adds new features that cover COTS-based system and PLE completely with detailed quantification. In addition, benefits of shortened TTM are also addressed in this model. After analyzing the challenging issues resulted from software reuse nowadays, this research focuses on these issues and gives solutions to them. By adding new features to Chmiel's model, the approach proposed by this research is applied to COTS reuse systems and PLE. In addition, attempts of quantifying benefits of shortened TTM are made within this study and a TTM submodel is developed to cover this critical issue.

Chapter 2 COTS, PLE and TTM

2.1 COTS (Commercial-Off-The-Shelf)

2.1.1 COTS definition

A COTS software product is a product with the following properties [Meyers 2001].

- sold, leased, or licensed to the general public
- offered by a vendor trying to profit from it
- supported and evolved by the vendor, who retains the intellectual property rights
- available in multiple, identical copies
- used without modification of the internals

Vigder defines that a COTS software component is software that is acquired from a commercial source and is integrated into a working system [Vigder 1998]. The driving force behind making using of COTS has been economics. COTS based systems are constructed by integrating large-scale components acquired from third parties. One of advantages of the COTS approach is flexibility. Flexibility arises from the ability to replace the COTS components used at a fraction of the cost normally required to develop those components from scratch [Erdogmus 2000].

The rationale for building COTS-based systems is that they will involve less development time by taking advantage of existing, market proven, vendor supported products, thereby reducing overall system development costs [Abts 1999]. There are two defining characteristics of COTS software, and they drive the whole COTS usage process:

- COTS product source code is not available to the application developer, and
- The future evolution of the COTS product is not under the control of the application developer.

There are several groups of COTS products that have been successfully used in software development [Vidger 1998]:

- Geographic Information Systems (GIS)
- Graphics User Interface (GUI) builders
- Office automation software, such as calendars, word processors, spreadsheets, etc.
- E-mail and messaging systems
- Databases
- Operating systems, including low-level software such as device drivers, window systems, etc.

Following is a table to show some examples of COTS software and brief description to them.

COTS name	Domain/ Functionalities
STK: Satellite Tool Kit	Orbit determination and mission planning
Oracle	Database administration and management
Labview	Data acquisition, analysis and visualization
Autocon	Orbit determination
Altair Control System Mission Control	Mission control
Microsoft Office	Office application
Probe	Data analysis
Crystal Reports	Report generator
GensAa	Spacecraft monitoring, commanding, fault detection and isolation
GTDS	Orbit determination
PhotoShop	2 dimension graphic design
Builder Xcessory, X-Software, Shared-X, Visual Optimization package, X Runner	GUI, GUI builders
Matlab	Computing environment, data visualization, application development
Windows XP	Operating system

Table 2: COTS examples

There is a spectrum of COTS software. Packaged software or solution software, for example, Office applications in the table 1, is designed with well-understood and specific tasks. There is little need for customization and the integration process is usually defined as COTS-solution System. What customers do is simply buy it and use it. On the other hand, information management software, for example, ERP, has a large and complex scale. Data is also large and complex. There is great temptation for customization. After purchasing, customers must implement based on their own specific situations before use these COTS software and this is usually define as COTS-Intensive or COTS-Integrated system. Both of the two approaches belong to COTS-Based System.

For the purposes of this study, the term COTS means a software product, supplied by a vendor, which has specific functionality as part of a system and whose source code is not accessible, which is composed of COTS components and extra efforts on tailoring and glue code are needed. In projects the COTS component is a piece of pre-built software that is integrated into the system, for example, COTS components such as GUI and report generator.

2.1.2 COTS-Based System (CBS)

A COTS based software system is a system that has been built primarily by assembling a set of COTS software. The integrator responsible for building the system does so by buying the components and assembling them into a complete system. This involves minimal code development as most of the components are not developed from source but are purchased off-the-shelf [Vigder 1998].

COTS-based systems include COTS products besides newly written (in-house) software, for example, glueware. The most important features of COTS products are their suitability for integration into different systems and commercial availability. These aspects allow COTS products to provide high quality prepackaged functionality. Therefore, the use of COTS products as components of a new system can reduce development effort and increase system quality. As a result, CBS development can significantly aid developers in building a better product in a shorter time.

For example, a COTS finance package could take as much as five years to develop with a team of 15 to 20. That would be a cost of well over $15 million. Using a COTS package could be accomplished in two years or less with about 10 staff. That would be a cost of about $3 million. The savings is potentially substantive, and the time saving is 60 percent [Whitemarsh 1997]. The main benefit is that a CBS involves the minimum of code implementation since most of components are not developed from scratch but purchased off-the-shelf. There is still some coding coming from:

- Tailoring a commercial software to a specific requirement.
- Building functional components that are not being supplied through commercial sources.
- Integrating the customized components into the CBS.

COTS-based systems comprise a spectrum, ranging from COTS-solution systems at one extreme, to COTS-integrated systems at the other extreme. COTS-solution systems are pre-integrated systems that are customized and deployed for use; examples include enterprise resource management packages and payroll packages. In COTS-solution systems, few customizations are needed on client side. COTS-integrated systems are assembled from (frequently many) COTS components provided by different vendors [Wallnau 1998]. This type of COTS-based system needs more extra efforts on tailoring and glue code.

Similar to Wallnau, Carney identifies three types of COTS-based systems based on the number of COTS used and their influence on the final system [Carney 1997].

1. Turnkey systems are built around a (or a suite of) commercial products, such as Microsoft Office or Netscape Navigator. Only one COTS package is used, and customization does not change the nature of the initial COTS.
2. Intermediate systems are built around one COTS (e.g. Oracle) but integrate other

components, commercial or developed in house.
3. Finally other systems are built by integrating several COTS, all on the same level of importance.

In this study, COTS projects deal with the third category. It means a project may use several COTS. The corporation and project management needs to determine whether or not to purchase them instead of developing from scratch.

2.2 PLE (Product Line Engineering)

A software product line is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way.

The key to successful software product lines is systematic management of planned variations across the product line, while exploiting the commonalities. This commonality permits reuse of a multitude of shared assets, including everything from architecture and reusable components, schedules and budgets, test cases and performance modeling, training and documentation, to marketing plans and literature. When building a new product in a product line, programming is de-emphasized.

The vision of PLE is that:

- Product line development is a low risk/high return proposition.
- Techniques for finding and exploiting system commonalities and for controlling variability are standard software engineering practice in DoD, government, and industry.

A product line involves core asset development and product development using the core assets, both under technical and organizational management. Core asset development and product development from the core assets can occur in either order: new products are built from core assets, or core assets are extracted from existing products. Often, products and core assets are built in concert with each other. Core asset development has also been called domain engineering. Product development from core assets is often called application engineering.

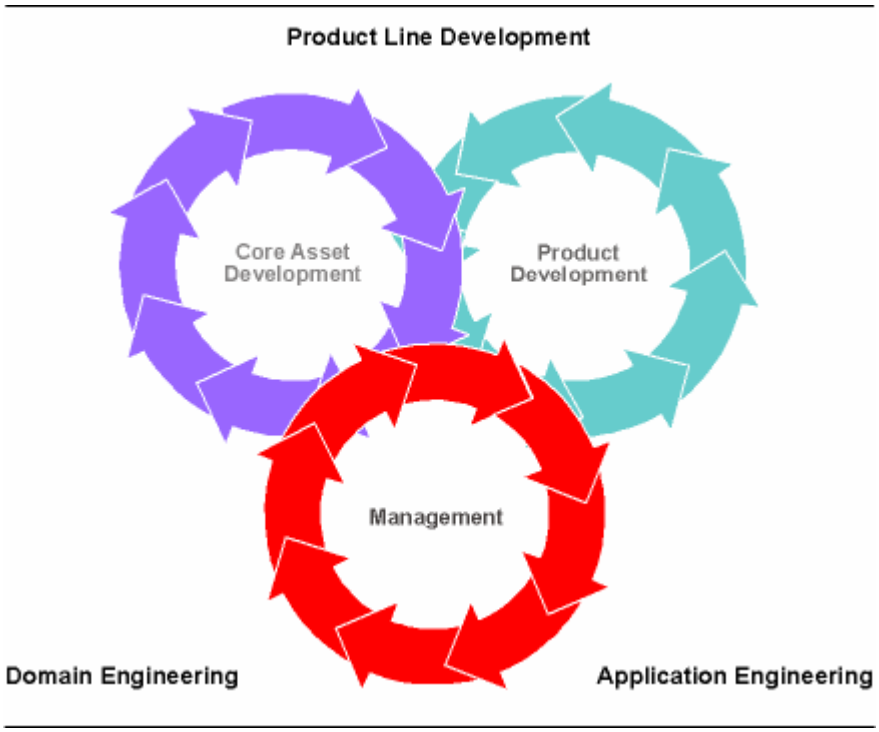

Figure 1: Software Engineering Institute: A Framework for Software Product Line Practice

Software product lines are rapidly emerging as a viable and important software development paradigm. A software product line is a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way [SEI 2003].

The basic idea of PLE is that the application engineering utilizes core assets from domain engineering. The product line is composed of products currently, probably more in the future, and the development of these similar products which share same architecture.

The reusable core assets include requirements and domain model, software architecture that the products will share, software components, design documentation, and etc. there are mainly three ways that these assets are imported into domains:

- Develop from scratch
- Mine legacy software inside corporation
- Buy from market, COTS

All of the three ways are covered in this study.

2.3 TTM (Time-To-Market)

Time is money and essence for the payoff of a software product. Besides technical benefits, such as higher quality and lower maintenance, the use of COTS or other reusable core assets can reduce time to market. The effects of shortened TTM come from more sales revenues and increase of stock value of a corporation.

2.3.1 Sales revenues

2.3.1.1 Quantify sales revenues

Software reuse can improve a product's time to a market. Higaki states that buying software saves development time and thus the product can be delivered earlier if all of the software are written from scratch [Higaki 1995]. Earlier TTM can result in increased profits from two effects: added profits from delivering the product earlier to marketplace, and added benefits from increase market share of over the life of the product [Smith and Reinertsen 1991]. Therefore, the direct effects of shortened TTM are more market share and more sales revenues. For example, if the product reaches the market two months earlier with the same functionality, then there will be an additional two months' revenue. To estimate this effect, Patterson [Patterson 1993] uses the projected annual sales and estimates the improvement in TTM:

$$S_I = S_V T_M$$

Where

S_I: sales increase resulting from earlier TTM
S_V: original projected volume per month
T_M: improvement in TTM in month

Nakano have the same ideas as Patterson and suggests a formula to calculate TTM benefits [Nakano 2000].

TTM Benefit = (schedule acceleration) x (achieved revenue/year)

Where

Schedule acceleration is how much faster a product goes to marketplace.

For example, suppose achieved revenue is 12 million/year and the schedule acceleration is two months. The TTM benefit is calculated as followed:

TTM Benefit = (schedule acceleration) x (achieved revenue)

= (2/12 year) x ($12 million/year) = $2 million

2.3.1.2 Earlier TTM resulting in a positive NPV

This study proposes the rationale that more market sales revenues are caused by the earlier TTM due to integrating COTS and other types of reusable components instead of implementing from scratch.

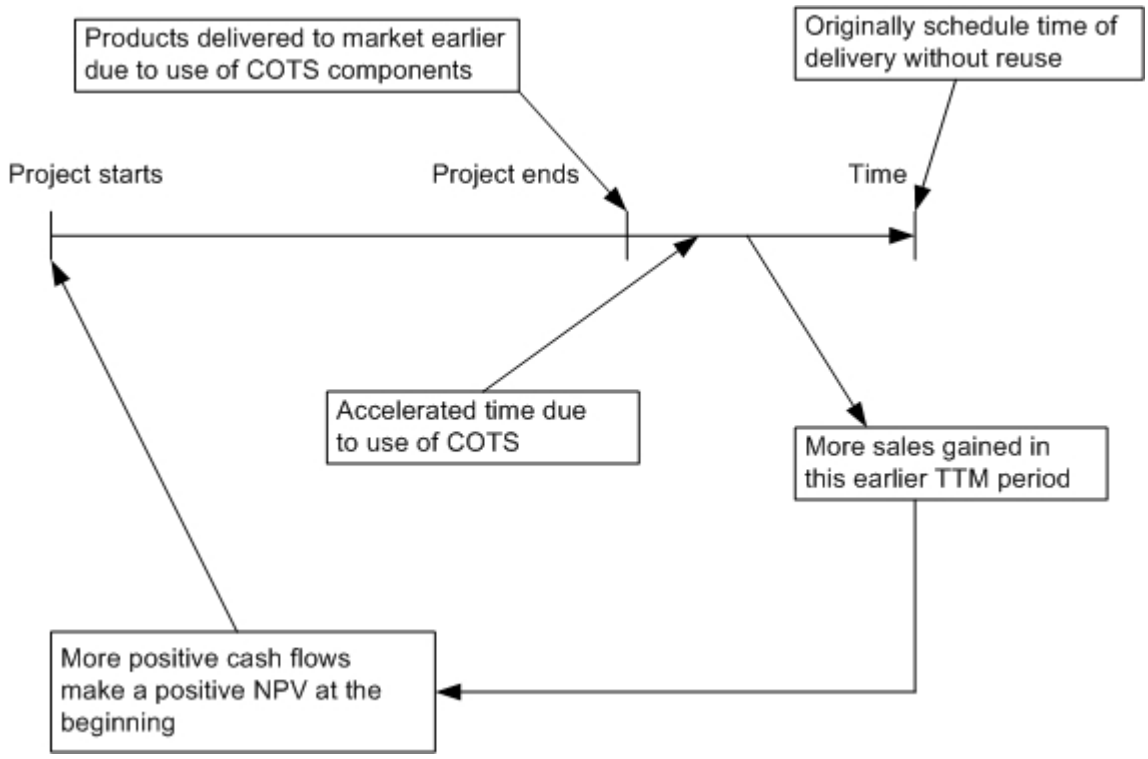

Figure 2: More sales revenues caused by earlier TTM

The TTM submodel in this study quantifies how much time earlier that product is delivered to market.

2.3.2 Stock value

2.3.2.1 Why stock value determining

Management's basic, overriding goal is to create value for stockholders. Managers seek to maximize the values of their firm's stocks [Brigham 1999]. Therefore, stock maximization should be the primary concern and goal of corporation management. Managers want to know if the stock value of the firm increases or not after the product is delivered to market. Earlier TTM

can result in increased profits and shows R&D capabilities of a corportaion. Consequently, by increasing the market value, earlier TTM causes increase on stock value. The consequence can also definitely satisfy customers, industry, and stockholders. When a company delivers its products to market sooner than its competitors, not only more market sales and sales revenues are gained, but also the increase on stock value by showing public its innovation capabilities and ability on product development.

In fact, the incentive of maximizing stock value is not only the economic concern from corporation managers, but also concern related to corporation control. The question of control has become a central issue in recent years. Managers who don't have majority control (more than 50 percent of their firm's stock) are very much concerned proxy fights [Brigham 1999]. Stockholders of a firm don't want an incompetent and inefficient management. The majority of the stockholders have the right to reelect new manages to take place of the inefficient ones. If the management can not bring stock price up, the stockholders are more likely to fire the management. This makes managers have to focus on stockholder concerns, which means the maximization of stock price.

Brigham states that 45 percent of U.S. adults own stocks directly, and 80 percent own stocks through retirement programs. Thus, most members of society have stake in the sock market [Brigham 1999]. A product with earlier TTM can attract more attention from public. Stock analyzers and investment bank would like to give more credits on this earlier delivery. Earlier TTM brings more market sales revenues and makes the project positive NPV. Brigham states that if a firm takes on a project with positive NPV, the wealth of the stockholders is improved [Brigham 1999]. As with any asset, the market value of a share of common stock is determined by the present value PV of its expected future cash flows [Vetzal 2003].

2.3.2.2 Factors influencing share price

A number of factors influence stock prices. These include internal factors and external factors. Internal factors can be:

- earnings growth
- sales growth
- product release
- leadership changes
- lawsuits pending

External factors can be:

- new market competition
- economic news such as interest rates and inflation
- government policy changes

A report about the increased profits, earnings growth, and sales growth of a company usually indicates that the company has ability to gain profits and may increase its dividends. This may draw investors and increase the price of the stock.

If a corporation postpones its products delivery time to market, its stock price usually drops down. For example, over the past years, Apple Computer Inc. postponed its products several times due to technical difficulties and its stock prices dropped down, since public doubted its capability to earn profits. On the contrary, if a corporation delivers its products earlier to market, public will give more credits on the capability of earning profits and R&D. For example, there has been a well-known competition between AMD and Intel in IT industry. Before 2000, AMD was left behind by Intel since AMD CPU clock speed couldn't compete with Intel. In March 2000, AMD began shipping a microprocessor that achieved an industry milestone with a clock speed of one gigahertz. At the same time, Intel didn't have the same product in market. It's the first time Intel was beat by AMD. The fact is that since AMD beat industry leader Intel for the fastest microprocessor with earlier TTM and also took market shares away from Intel, share price of AMD increased.

A corporation is owned by its shareholders and the shareholders' equity is the portion of total assets that belong to the shareholders. The value of shareholders' equity as recorded on the balance sheet is the book value of equity; dividing the book value of equity by the number of outstanding shares gives the book value per share [Vetzal 2003]. Stock share price is based on the investment returns or cash flows that the investor expects to receive from owning the share (depends on the ability of the company to earn a profit), As proved in last section, earlier delivery to market can result in more sales revenues and the positive cash flows increase the wealth of the corporation and thus the increased profits on balance sheet increase the stock value per share.

The market value of each share is the price required to purchase a share in the firm from a trade on a stock exchange; multiplying the share price by the number of outstanding shares gives the market value of equity. In general, these two equity values (book vs. market) are seldom equal, for most healthy firms, market value exceeds book value.

2.3.2.3 Economic Value Added (EVA) theory

Makelainen states that the real profit that is of interest to investors is the profit after deducting the capital costs. This profit figure is often called Economic Value Added, EVA (or Economic Profit or Residual Income) [Makelainen 1998]. EVA is operating profit (after taxes) of a company subtracted with the total cost of capital. The definition is depicted in the equation as followed:

EVA = Net operating profits after taxes – Cost of capital

Financial theory suggests that the market value of a company depends directly on the future EVA-values. EVA is a measure of surplus value created on an investment [Damordoran 2002].

The market value of a company
= Book value of equity + Present value of future EVA [Makelainen 1998]

The equation above is based on discounted cash flows. An essential component of EVA is the Weighted Average Cost of Capital (WACC).Generally speaking, the assets of a company are financed by either debt or equity. WACC is the average of the cost of each of these sources of financing weighted by their respective usage in the given situation. By taking a weighted average, we can see how much interest the company has to pay for every dollar it borrows.

EVA is estimated based on a NPV technique as followed:

$$EVA = \sum_{i=1}^{n} \frac{EVA_i}{(1+WACC)^i}$$

Where

EVA: Economic value added
n: number of years
WACC: Weighted Average Cost of Capital

There are three situations based on the value of EVA.

1. **EVA = 0 market value keeps the same.**

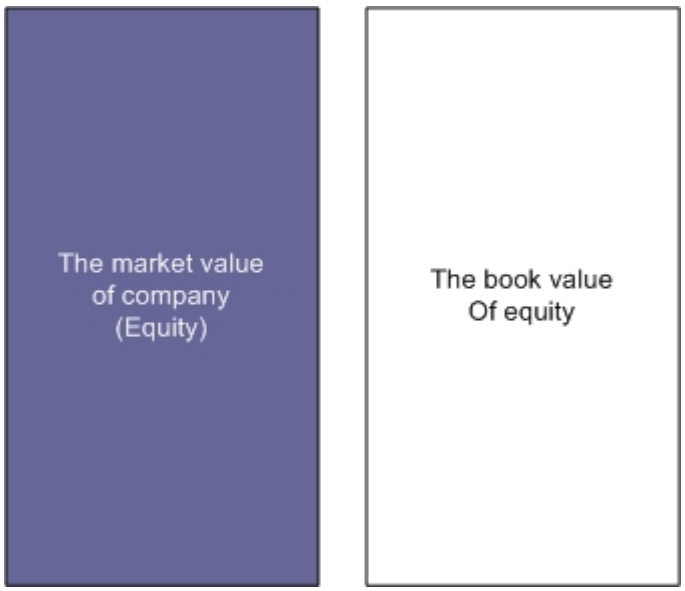

Figure 3: EVA = 0

$$EVA = \frac{EVA_1}{(1+WACC)^1} + \frac{-EVA_2}{(1+WACC)^2} + \ldots + \frac{EVAn}{(1+WACC)^n} = 0$$

In this situation, the company produces a return that is equal to capital costs. That means there are no profits gained and market value does not increase. Then the market value of the company will equal the current book value of equity (no premium or discount).

2. **EVA > 0 market value increases.**

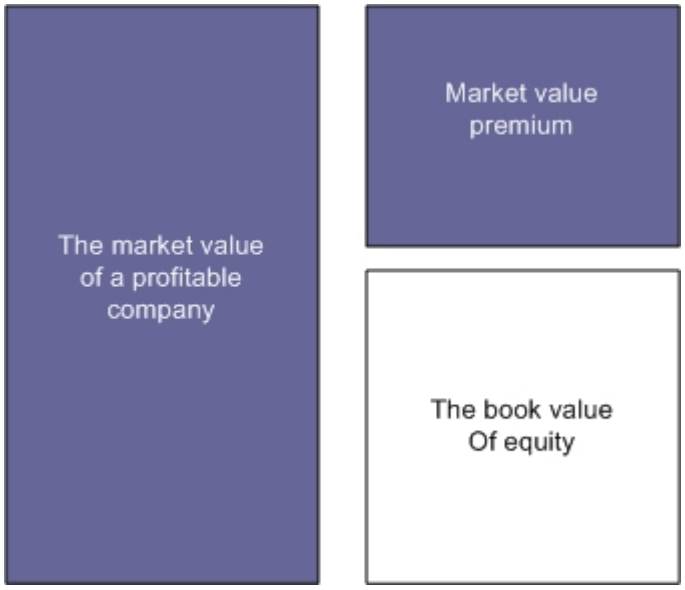

Figure 4: EVA > 0

$$EVA = \frac{EVA_1}{(1+WACC)^1} + \frac{EVA_2}{(1+WACC)^2} + \ldots + \frac{EVAn}{(1+WACC)^n} > 0$$

Positive EVA adds a premium to the market value of equity of a corporation. The positive EVA comes from a positive NPV based on future cash flows.

3. **EVA < 0 market value decreases.**

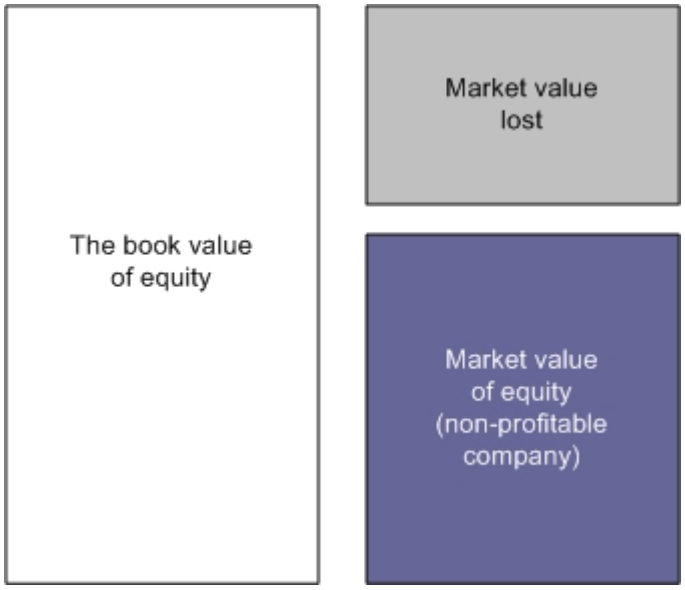

Figure 5: EVA < 0

$$EVA = \frac{-EVA_1}{(1+WACC)^1} + \frac{-EVA_2}{(1+WACC)^2} + \ldots + \frac{-EVAn}{(1+WACC)^n} < 0$$

Negative EVA causes discount to the market value of equity of a corporation. The negative EVA comes from a negative NPV based on future cash flows.

Stock prices reflect the future EVA expectations. Makelainen states that the bigger expected EVA the company has, the bigger is the market value of the company and the stock price. Especially profitable growth (growth in EVA) gears up stock prices [Makelainen 1998]. This study treats the NPV of a project using COTS and other types of reusable components as EVA. The extra benefits due to use of all of reusable components are the positive added value to the project.

2.3.2.4 Earlier TTM resulting in share value increase

There are so many factors who can affect stock price. Based on the factors related to earlier TTM, such as corporation wealth and cash flows, this study states that earlier TTM can result in more sales revenues from market. The more sales revenues make corporation more positive cash flows and thus a positive NPV in the end. Obviously, the book value will increase. And from external factors, product with earlier TTM can beat competitor and gain more credits from stock market analyzer and investment banks. This rationale proposed by this study based on earlier TTM can be depicted as followed:

more credits from stock market analyzer and investment banks. This rationale proposed by this study based on earlier TTM can be depicted as followed:

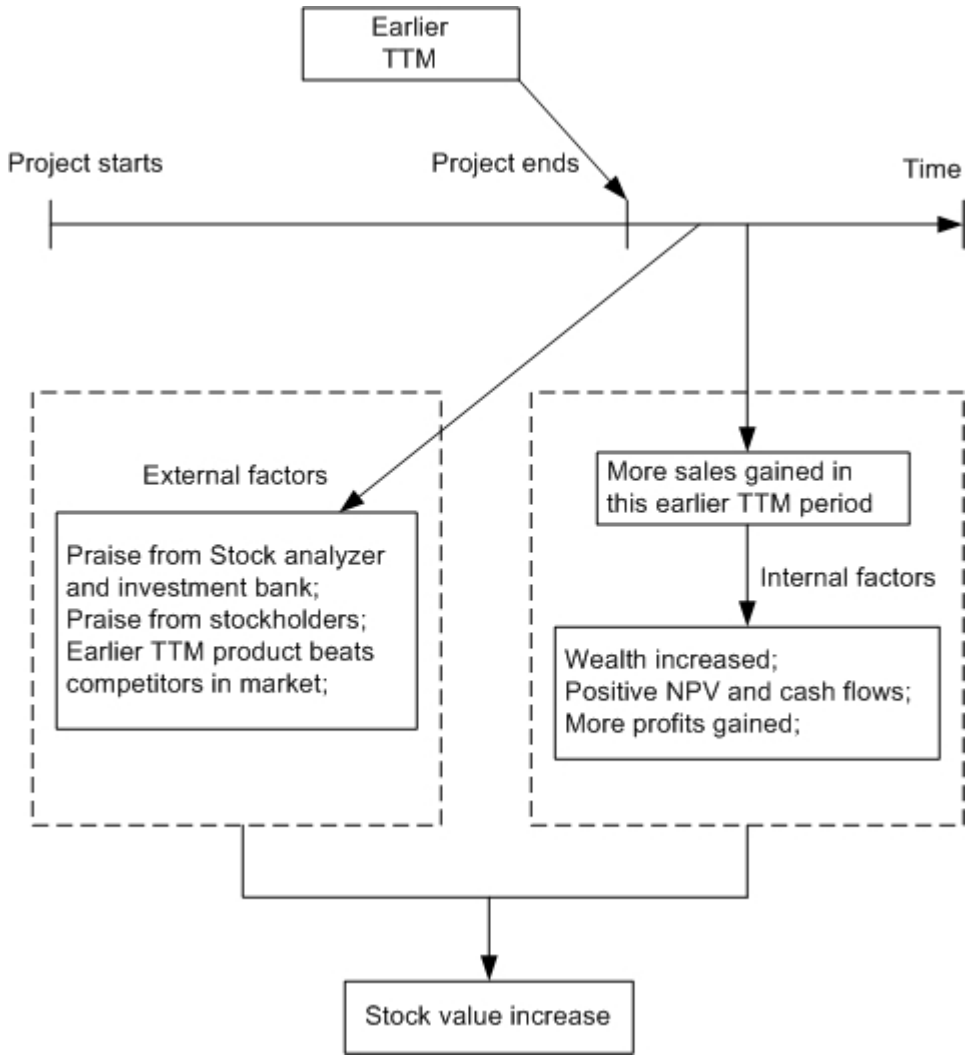

Figure 6: stock value increase caused by earlier TTM

Similar to Makelainen, Damordoran also proves the firm value is determined by NPV of future projects, which is named economic value added in EVA theory. The equation from Damordoran is as followed:

Value of Firm = Value of Assets in Place + Value of Future Growth
= Value of Assets in Place + NPV of all future projects [Damordoran 2002]

As shown in the equation above, a positive NPV of a project can increase a firm's market value. This study deals with the second item, NPV of all future projects.

This study treats the increase on stock value as a further benefit after more sales revenues occur due to earlier delivery to market. Basically, the earlier TTM results in more market share and helps corporation make the project a positive NPV. At the same time the positive cash flows increase the wealth of the corporation and stockholders. As proven in the EVA theory and Damordoran's equation, therefore, the stock market value of the corporation also increases.

Is it a 100% certainty that stock price must increase? The answer is that if the project has a positive NPV, then the certainty is higher since if a firm takes a project, which yields a positive NPV. That means the wealth of stockholders increases and firm value also increases. However, the stock price is not only determined by internal factors, such as firm value, positive NPV. As discussed in previous section, external factors, such as governmental policy, industry competition, can also affect the variations of the stock price. And if the project has a negative NPV, at a higher rate of expectation, we can draw a conclusion that the stock price will decrease.

The benefit of stock value increase belongs to non-technical issue and so far there is no such a mode formally validates and covers this issue. A few models mentions and recognizes the benefit due to shortened TTM caused by using reusable components but doesn't put more efforts on quantifying this. Most other models simply ignore it. This study more focuses on internal factors such as profit, earning growth, and sales growth. Based on these internal factors, this study attempts to resolve this challenging issue by proposing a TTM model. The quantification part is covered in TTM submodel section.

Chapter 3 Software Economics Cost models

3.1 COCOMO (COnstructive COst MOdel)

The most fundamental calculation in the COCOMO model is the use of the Effort Equation to estimate the number of Person-Months required developing a project. Most of the other COCOMO results, including the estimates for Requirements and Maintenance, are derived from this quantity. The COCOMO calculations are based on estimates of a project's size in estimated thousand delivered source instructions (KDSI).

Intermediate COCOMO estimates for the time of development and the amount of effort needed for software development. It was later discovered that the schedule and effort are influenced by certain factors related to the difficulty of the project. The level of difficulty (or familiarity) is broken down into 3 modes [COCOMO 1994]:

1. Organic mode is used to calculate the effort for a project where constraints upon development are mild. In addition, the given project has been pre-dated by a number of similar projects that could assist in defining the agenda of development.
2. Semi-detached mode is used for a project where the constraints on the project are greater than organic mode, but there still remains some flexibility. The project may only be pre-dated by a few similar projects.
3. Embedded mode is used for a project that has very tightly defined constraints. The project as a whole is a trailblazer and therefore cannot rely upon previous projects completed.

Development Mode	**Project Characteristics**			
	Size	**Innovation**	**Deadline/Constraints**	**Dev. Environment**
Organic	Small	Little	Not tight	Stable
Semi-detached	Medium	Medium	Medium	Medium
Embedded	Large	Greater	Tight	Complex Hardware/customer interface

Table 3: COCOMO development mode

3.1.1 COCOMO Equations

There are two main equations in COCOMO.

1. Development effort

$$PM = a \times KDSI^{b}$$

This equation is based on PM - person month / staff-month. COCOMO defines there are 152 hours per Person month. According to organization this values may differ from the standard by 10% to 20%.

2. Effort and development time (TDEV)

$$TDEV = 2.5 \times PM^{c}$$

This equation estimates how much time to spend to develop with a unit in month.

The values of *a, b*, and *c* are based on different modes.

Mode	***a***	***b***	***c***
Organic	3.2	1.05	0.38
Semi-detached	3.0	1.12	0.35
Embedded	2.8	1.20	0.32

Table 4: COCOMO parameters

Here is an example to show how these two equations work. Development team plans to implement a database system for an office automation. The project is in organic mode (a=3.2, b = 1.05, c=0.38) and there are 4 modules to implement:

- Data entry: 0.6 KDSI
- Data update: 0.6 KDSI
- Query: 0.8 KDSI
- Report generator: 1.0 KDSI
- Total System SIZE: 3.0 KDSI

$PM = 3.2 \times 3^{1.05} = 3.2 \times 3.17 = 10.14$ (PM)

$TDEV = 2.5 \times PM^{c} = 2.5 \times 10.14^{0.38} = 6.03$ (Calendar Months)

The value of TDEV indicates that the development time for this project will last more than 6 months. The TDEV equation is used in this study to estimate how much time can be saved to deliver to market.

3.1.2 Equivalent delivered source instructions (EDSI)

Equivalent delivered source instructions (EDSI) is calculated from the following estimated adaptation quantities [COCOMO 1994]:

- Adapted Delivered Source Instructions (ADSI). The number of delivered source instructions adapted from existing software used in developing the new product.
- Percent of Design Modification (DM). The percentage of the adapted software's design that received modification to fulfill the objectives and environment of the new product.
- Percent of Code Modification (CM). The percentage of the adapted software's code that receives modification to fulfill the objectives and environment of the new product.
- Percent of Integration Required for Modified Software (IM). The percentage of effort needed for integrating and testing of the adapted software in order to combine it into the new product.

The equations for calculating EDSI involve an intermediate quantity, the adaptation adjustment factor (AAF), as followed:

$$AAF = 0.4(DM) + 0.3(CM) + 0.3(IM)$$

$$EDSI = (ADSI)\frac{AAF}{100}$$

Following is an example to show how EDSI works. Suppose we are converting a 50-KDSI organic-mode Fortran electronic circuit analysis program from a Univac 1110 computer to an IBM 3033. Typically, for this situation, we would have

DM = 0	(no change in the program's design)
CM = 15	(perhaps 15% of the lines of code will change because of compiler change, operating system interfaces, job control language changes, and so on)
IM = 5	(a small amount of effort required to integrate the above changes)

The resulting adaptation calculations would be

$$AAF = 0.4(0) + 0.3(15) + 0.3(5) = 6$$

$$EDSI = (50{,}000)(\frac{6}{100}) = 3000$$

Using the COCOMO organic-mode estimation equation, the resulting conversion effort would be

$$MM = 2.4(KEDSI)^{1.05} = 2.4(3)^{1.05} = 7.6MM$$

The result shows that 7.6 MM effort will be added to integrate this software component into a new environment.

3.1.3 Summary

The original COCOMO model has been successful, but it doesn't apply to newer software development practices as well as it does to traditional practices. In addition, the other problem with this model is that it assumes no reuse and few changes in the requirements after they are identified. Thus, this model does not reflect the assumptions of reuse. The first and primary approach modeled by COCOMO is the use of system components that are built from scratch, that is, new code. Hence, the original COCOMO doesn't apply to CBSE and COTS. COCOMO is good for a traditional approach, and uses a 3GL (third generation language), such as C, FORTRAN, or COBOL and the original COCOMO will give good results. If development tools and processes haven't changed much in recent years, COCOMO might be the right model.

This study utilizes these basic equations in COCOMO as a tool to estimate the development efforts and time of developing from scratch without software reuse.

3.2 Chmiel's Integrated Cost model

3.2.1 Model introduction

This model is different from others in that the decisions are composed of four levels and treats reuse projects from a point of view of long term run. Each level corresponds one engineering cycle. Each level is a decision making process based on calculation of NPV, ROI, and other economic indices. Reuse investment decisions are made on different levels from the corporate to the programming.

- At the corporate level, management determines if the reuse investment will be worthwhile.
- At the managerial level, domain engineering managers determine which assets that have good reuse potential and create them. And the application managers determine the worthiness of incorporating reusable assets into a project.
- At the programming level, named Component Engineering Cycle, programmers implement reusable assets based on the decision from its upper level, domain level.

These decisions of different levels are reflected in the following 4 investment cycles:

- Corporate investment cycle.
- Domain engineering investment cycle.
- Application engineering investment cycle.
- Component engineering investment cycle.

Corresponding to these four cycles, there are four stakeholders in the software reuse lifecycle. Each stakeholder is responsible for a key decision in the corresponding cycle. The basic structure of this model is that it utilizes a cascade pattern. For example, the costs and benefits in Component Engineering Cycle are cascaded into Domain and Application Engineering Cycles. In the end, the costs and benefits of the latter two cycles are cascaded into Corporate Engineering Cycle and the decision is made based on the cascaded costs and benefits.

These investment cycles are characterized analyzed by the following cost factors:

- Investment costs, IC measured in person months.
- Periodic costs at year y, C(y) measured in person months at year y, including the costs of creating, retrieving, and using reusable assets.
- Periodic benefits at year y, B(y) measured in person months at year y, gained by creating, using, and classifying a reusable asset.
- Discount rate, d measured in person months at year y
- Duration, Y measured in number of years and ranging typically 3 to 5.

The four decisions are based on the economic rationale.

- Net Present Value (NPV),
- Payback Period (PB),
- Average Return on Book Value (ARBV),
- Internal Rate of Return (IRR),
- Return on Investment (ROI),
- Profitability Index (PI).

For each cycle, these values are calculated and the decision is based on the values. Here is a figure showing the structure of the model.

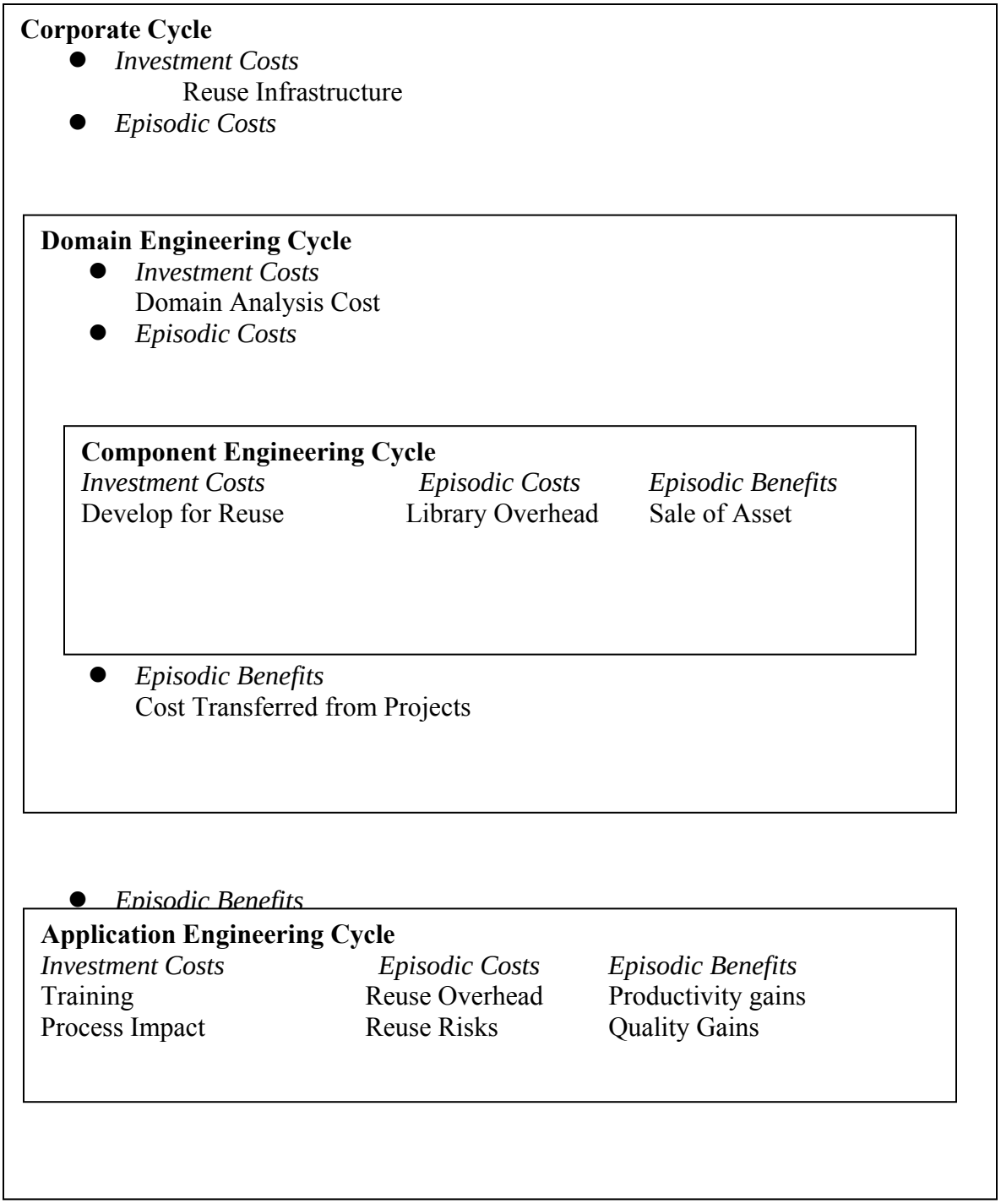

Figure 7: Software Reuse Model [Chmiel 2000]

This model assumes reusable components are analyzed and developed in Component Engineering Cycle and Domain Engineering Cycle. In Application Engineering, application engineers get benefits by using these reusable assets.

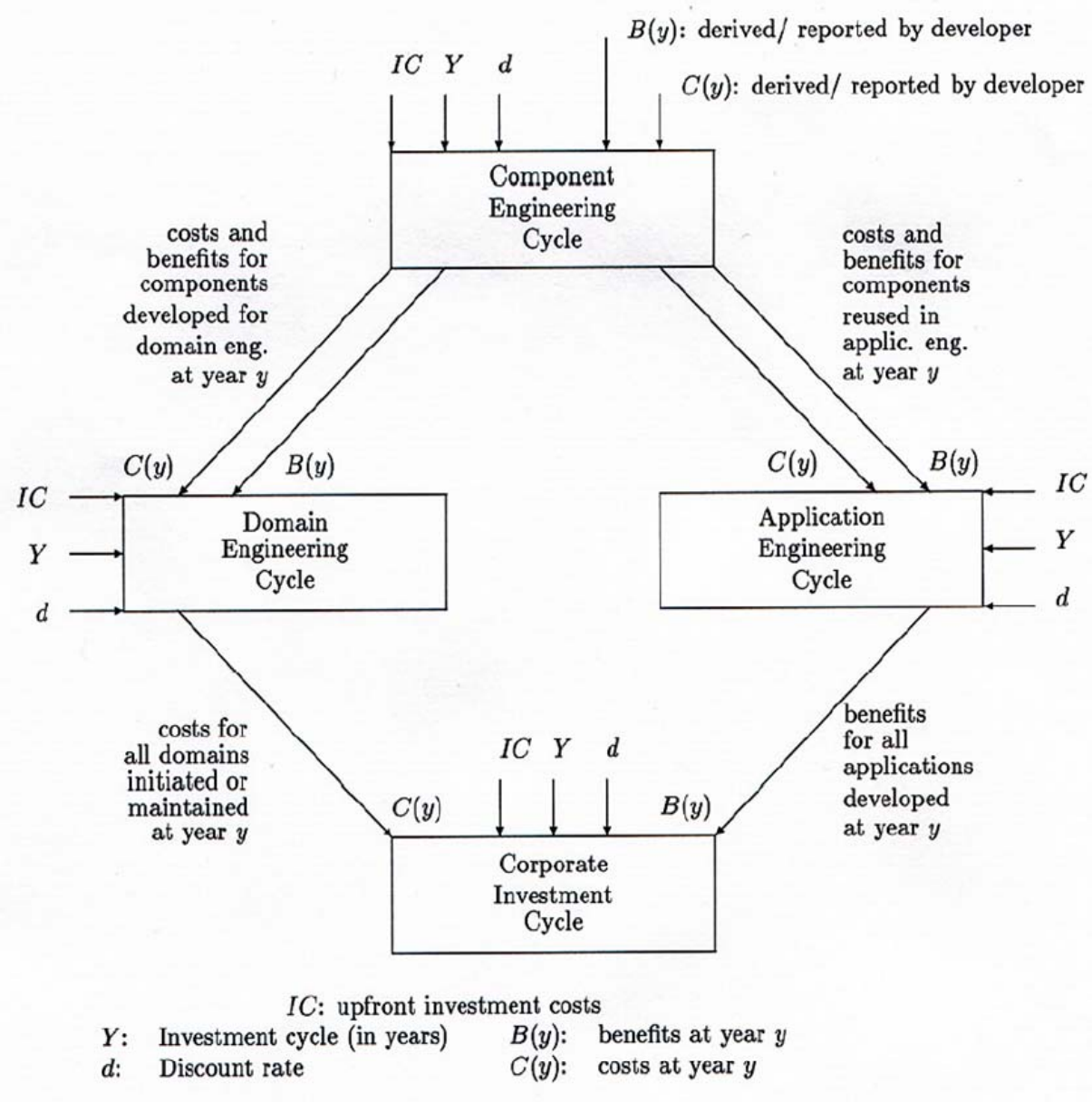

Figure 8: Cascade of Costs through Investment Cycles [Chmiel 2000]

The costs and benefits of components developed for domain engineering comprise the periodic costs of the domain engineering cycle. These costs plus the initial costs of domain analysis during the domain engineering cycle propagate to the corporate engineering cycle. Thus, the component engineering cycle is a subset of the domain engineering cycle which is a subset of the corporate engineering cycle. By minimizing the costs of component engineering and domain

engineering, the costs of the corporate engineering cycle can therefore be minimized. The benefits of application engineering follow a similar pattern [Chmiel 2000].

The figure above shows the basic cost and benefit structure of Chmiel's model. Besides initial investment costs, periodic costs and benefits of Corporate Engineering Cycle are cascaded from Domain and Application Engineering Cycles. On a lower level, the costs and benefits from Component Engineering Cycle are cascaded into Domain and Application Engineering Cycles.

The rationale of this model is that to maximize the ROI at the corporate level, one must maximize the ROI at each of the intermediate levels (domain, application, and component) [Chmiel 2000].

3.2.2 Summary

Chmiel's model is based on time value of money and has its advantages on quantifying productivity and quality gains in much more details than other models. The classification of the four engineering cycles and the cascade pattern in Chmiel's model can represent the basic process and cost structure of COTS and PLE. This study inherits this classification and the cascade pattern.

This model doesn't cover COTS and PLE and benefits due to shortened TTM and can only deal with internal traffic since the model assumes all of the reusable components are built from scratch in house. For example, Extra efforts caused by the use of COTS, assessment, tailoring, glue code and COTS volatility, are not covered in this model. And this model doesn't treat the benefits of TTM, for example, business performance.

3.3 COCOTS (COnstructive Commercial-Off-The-Shelf)

3.3.1 Introduction

COCOTS is a model complementary to COCOMO II, capturing costs that traditionally have been outside the scope of COCOMII. The model focuses on estimating the cost, effort, and schedule associated using COTS components in a software development project.

A definition of COTS components is: commercial software product - sold, leased, licensed at advertised prices, source code unavailable, usually periodic releases with feature growth (obsolescence), future development not under control of application developer [Clark 2003].

There are some basic risks inherent using COTS components: immaturity of the product and/or the vendor, inexperience of integrators and/or users with the product, incompatibility of the product with the larger application, platform, or other COTS components in the system, lack of control over the product's current and future functionality (COTS is evolutionary (volatility)),

customer/user view that COTS integration is Plug-and-Play (for example, the assumption that use of COTS means zero development time).

COCOTS analyzes the benefits and costs of initial integration of COTS. OCOCOTS is composed of four related submodels (Assessment, Tailoring, Glue Code, and Volatility), each addressing individually the four primary sources of COTS software integration costs.

The deficiency of COCOTS is that COCOTS currently deals only with initial integration efforts.) Initial integration costs are due to the effort needed to perform as followed [Boehm 1998]:

- **Assessment Effort**: Candidate COTS component assessment,
- **Tailoring Effort**: COTS component tailoring
- **Glue code Effort**: The development and testing of any integration or "glue" code needed to plug a COTS component into a larger system
- **Volatility Effort**: Increased system level programming and testing due to volatility in incorporated COTS components. It is the additional effort that results from the impact on the larger system of the effects of swapping COTS components out of the system with newer version of those components that have been released by the COTS vendors.

By collecting data and analyzing COTS-Based projects, Madachy [Ray 2004] gives COTS activity efforts percentage.

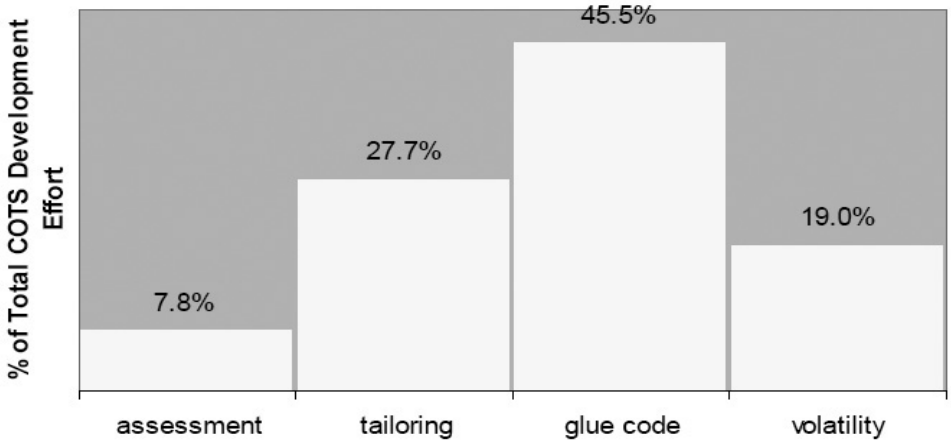

Figure 9: COTS efforts distribution over 4 phases [Ray 2004]

The following equations are used in COCOTS to apply to initial integration of COTS [Boehm 1998].

3.3.2 Efforts estimation and calculation in COCOTS

3.3.2.1 Initial filtering effort

$$Total\,Effort = (\#COTS\,Candidates)(Average\,Filtering\,Effort\,/\,Candidate)$$

3.3.2.2 Final selection effort

$$Total\,Effort = \sum_{\substack{Assessement \\ attributes}} (\#COTS\,Candidates)\left(\frac{Average\,Assessment\,effort\,for\,attribute\,in\,given\,domain}{Candidate}\right)$$

The assessment attributes are listed as below:

1. Correctness
2. Availability/Robustness
3. Security
4. Product Performance
5. Understandability
6. Ease of Use
7. Version Compatibility
8. Inter-Component Capability
9. Flexibility
10. Installation/Upgrade
11. Portability
12. Functionality
13. Price
14. Maturity
15. Vendor Support
16. User Training
17. Vendor Concessions

Assessment Effort = Filtering Effort + Final Selection Effort

3.3.2.3 Tailoring effort calculation

$$Total\,Effort = \sum\left(\begin{matrix}COTS\,Candidates \\ tailored\ at\,complexity\,level\end{matrix}\right)\left(\begin{matrix}Average\,effort\,at\,tailoring\,complexity \\ level\,in\,domain\end{matrix}\right)$$

The sum is over the complexity levels that are rated based on the following tailoring activities:

1. Parameter specification
2. Script Writing
3. I/O Report and GUI Screen Specification and Layout
4. Security/Access Protocol Initialization and Setup
5. Availability of COTS Tailoring Tools

3.3.2.4 Glue code development and test effort calculation

$$Total\ Effort = A \times [Size]^B \times \prod_{i=1}^{14} EM_i$$

Where

$$A = 12.0$$
$$B = 1.00 + (0.04 \times SF)$$
$$Size = KSLOC \times \left(1 + \frac{Brak}{100}\right)$$

A: Constant, provisionally set to 12.0
Brak: Breakage: Percentage of COTS glue code thrown away due to requirements volatility
KSLOC: Size of COTS component glue code expressed in
EM: Effort Multipliers: ACIEP, ACIPC, AXICP, APCON, ACPMT, ACSEW, APCPX, ACPPS, ACPTD, APVOL, ACREL, AACPX, ACPER, ASPRT
SF: Scale Factor, AAREN

The effort multipliers are:

1. ACIEP: COTS Integrator Experience with Product
2. ACIPC: COTS Integrator Personnel Capability
3. AXCIP: Integrator Experience with COTS Integration Processes
4. APCON: Integrator Personnel Continuity
5. ACPMT: COTS Product Maturity
6. ACSEW: COTS Supplier Product Extension Willingness
7. APCPX: COTS Product Interface Complexity
8. ACPPS: COTS Supplier Product Support
9. ACPTD: COTS Supplier Provided Training and Documentation
10. APVOL: COTS Product Volatility
11. ACREL: Constraints on System/Subsystem Reliability
12. AACPX: Application Interface Complexity
13. ACPER: Constraints on System/subsystem Technical Performance
14. ASPRT: System Portability

The nonlinear scale factor is:

AAREN: Application Architectural Engineering

How adequate/sophisticated were the techniques used to define and validate the overall systems architecture?

Rating	Driver													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
	ACIEP	ACIPC	AXICP	APCON	ACPMT	ACSEW	APCPX	ACPPS	ACPTD	APVOL	ACREL	AACPX	ACPER	ASPRT
VL	1.34	1.60		1.58	1.45				1.20	0.71				
L	1.16	1.27	1.12	1.26	1.20	1.07	0.82	1.14	1.09	0.84	0.88	0.84		
N	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
H	0.86	0.79	0.89	0.80	0.83	0.94	1.22	0.88	0.91	1.19	1.14	1.19	1.11	1.07
VH	0.75	0.62	0.79	0.63	0.69	0.88	1.48	0.77	0.84	1.33	1.30	1.42	1.22	1.14

Linear Scaling Factor
A
12.0

Nonlinear Scaling Factor				
AAREN				
VL	L	N	H	VH
4.00	3.00	2.00	1.00	0.00

Table 5: Value ranges of effort multipliers and nonlinear scale factor [Abts 1997]

3.3.2.5 Application Effort Due to COTS Volatility

Approximate model

$$Total\ Effort = (Application\ Effort)\left[\frac{BRAK\ COTS}{100}\right](EAF)_{COTS}$$

Detailed Model with COCOMO II Parameters

$$Total\ Effort = (Application\ Effort)\left[\left(1+\frac{BRAK\ COTS}{1+BRAK}\right)^{1.01+\sum} - 1\right](EAF)_{COTS}$$

Where

- *BRAK COTS*: % application code breakage due to COTS volatility
- *BRAK* : % application code breakage otherwise
- *S* : COCOMO II scale factor
- *EAF* : Effort Adjustment Factor (product of effort multipliers)

The application Effort Due to COTS Volatility is based on the following scale factor:

- Precedentedness
- Development Flexibility
- Architecture/Risk Resolution

- Team Cohesion
- Process Maturity

Hence

Total Integration Effort (in Person-Months) = Assessment Effort
+ Tailoring Effort
+ Glue Code Effort
+ Volatility Effort

Where

Total integration Cost = (Total Integration Effort) ($$/Person-Month)

3.3.3 Summary

COCOTS primarily focuses on initial development since COCOTS views costing as a one-time activity. However, CBS is an ongoing activity based on the whole life cycle of a project. For example, volatility cost happens every year and should be estimated based on time value of money in a long run project, but COCOTS treats it as a one-time activity. COCOTS more focuses on integration costs and does not support continuous development efforts estimation from a whole-life viewpoint. There are other costs and benefits that COCOTS doesn't cover. For example, COCOTS doesn't consider the costs of reusable library and other episodic operation costs. By using volatility cost factor, COCOTS solves the maintenance costs, but it treats it as a one-time activity. In fact, maintenance costs happen every year of a project and should be estimated based on time value of money. Therefore, COCOTS cannot deal with a reuse project lasting years.

COCOTS only estimates the efforts and costs of assessment, tailoring, glue code, and volatility. COCOTS doesn't count into other costs and benefits due to the use of COTS, for example, the cost of maintenance and the benefits of earlier time to market. In COCOTS, time value of money is not considered. COCOTS does not treat the long term operation and maintenance. In addition, COCOTS is not fully formulated and validated since COCOTS only deals with initial integration. COCOTS is still immature.

Chapter 4 Economic Techniques

This integrated empirical model with TTM submodel is based on NPV and ROI. Followings are introductions to the involved economic techniques in this study.

4. 1 Net Present Value (NPV)

The value today of a future cash flow or series of cash flow is called the present value. Net Present Value is the cumulative present worth of income (positive values) and a series of future payments (negative values) from a series of investment cash flow using a discount rate to handle time values of money [Brigham 1999].

$$NPV = CF_0 + \frac{CF_1}{(1+k)^1} + \frac{CF_2}{(1+k)^2} + ... + \frac{CFn}{(1+k)^n} = \sum_{t=0}^{n} \frac{CF_t}{(1+k)^t}$$

Where

CF_t: the expected net cash flow at period t
k: discount rate
n: project life

NPV is present value of all cash inflows is compared to the present value of all cash outflows. The investment project is acceptable if NPV > or = Zero [Agrizzi 2003]. The rationale for NPV is that an NPV of zero signifies that the project's cash flows are just sufficient to repay the invested capital. If a project has a positive NPV, then it is generating more cash that is needed to service its debt and to provide the required return to shareholders, and this excess cash accrues solely to the firm's holders [Brigham 1999].

The advantages of NPV are:

- Based on cash flow
- Considers all cash flows
- Incorporates the time value of money

The disadvantages of NPV are:

- must forecast all cash flows
- considers all cash flows, sometime it can be hard to estimate discount rate

NPV is the most widely accepted criterion for project evaluation in corporate finance [Ross 1996]. In corporate finance, NPV is the standard for making capital budgeting decisions. The NPV of a project is given by the sum of its discounted future cash flows. The discounting takes

into account time value of money, or the notion that money that is yet to be expended or received is worth less today than it is in the future. The NPV rule states that only projects with positive NPV are worth undertaking, and a project with a higher NPV is preferable to a project with a lower NPV.

This study uses NPV as a main estimation technique to calculate COTS strategy and helps corporate management make decision to take the CBS strategy or not.

4.2 Return On Investment (ROI)

This study deals with a project that can last long life cycle, for example, several years. The long length time of a software development project makes capital expenditures more risky than other investments. Before beginning a capital expenditure program that involves a large outlay of investment funds on purchasing COTS, which will probably last several years, corporation management should seek assurance that they will receive an acceptable return on investment.

In order to make the decision of whether or not to purchase COTS components instead of developing from scratch, the predicted cash flows must be compared to the required investments to determine if the return generated from using COTS meets or exceeds what management considers acceptable.

Return On Investment (ROI) analysis allows decision makers to determine the financial return by comparing net program benefits (benefits minus costs) to costs [Careertools 2000]. The purpose of this ROI evaluation technique is to assist corporation managers in evaluating whether or not to pay price for use of COTS components. The basic equation of ROI is as followed:

$$ROI = \frac{Benefits - Costs}{\mathrm{Pr}ogram\ \mathrm{cos}ts} \times 100$$

For example, a project results in $1,500 in benefits and $1,000 in total costs. The NPV of this project is (1500-1000)/1000 = 0.5. That means the project yields a ROI of 50%.

This calculation of ROI, also known as the book value rate of return, is commonly used because it is based on the accrual method of financial statement preparation, and is easy to apply. Its weakness is that is fails to consider the time value of money. Therefore, this study uses a combination of NPV and ROI, since NPV is based on time value of money. This combination assures the accuracy of calculating and evaluating the benefits and costs of COTS components.

For ROI calculations, rationale is that the higher the percentage, the more desirable the project. When ROI is equal to zero, that means there is no net benefits gained back. When ROI value is greater than zero, that means the project is worth to undertake.

This study also uses ROI to determine if COTS strategy is worth to carry out. For example, only if the final ROI value of analysis of COTS strategy is greater than zero, the COTS strategy would give corporate management more confidence to take it.

4.3 Relative Cost of Writing for Reuse (RCWR)

RCWR is the ratio of the effort it takes to develop reusable software to developing it from scratch.

$$RCWR = \frac{\text{Cost of developing reusable asset}}{\text{Cost of developing single - use asset}}$$

The value of RCWR is based on environmental factors like experience, reuse organization, and software complexity. Here is a table showing summary of RCWR values.

Organization	RCWR
Caldwell	1.23-1.3
Favaro	1.0-1.2
Gaffney and Cruickshank	1.5
IBM	1.25-2.0
Jones	1.5
Lim	1.11-1.8
Margano and Rhoads	2
Reifer	1.1-1.36
Tracz	1.6
Bardo, et al	1.15-1.25
Lockheed Martin Federal Systems	1.86
Pant, et al	1.55

Table 6: RCWR Values over 12 organizations [Poulin 2002]

This study takes value of 1.5 as default. However, users to the supporting tool have privilege to set up a RCWR value as they wish.

Chapter 5 Extensions of Chmiel's Model to COTS, PLE, and TTM

5.1 Model architecture

Based on inheriting the basic structure of Chmiel's model, the integrated cost model with TTM submodel is composed of four engineering cycles and one TTM submodel:

- Corporate Engineering Cycle, which reflects the corporate management point of view and determines the long-term benefits of a reuse program. Based on the costs and benefits in the project and plus the benefits of shortened TTM, the corporate makes final decision to initiate reuse activities.
- Domain Engineering Cycle, which reflects the domain manager point of view. The domain engineers in this cycle commit domains analysis initiative and analyze candidate COTS components or existing reusable components for assessment.
- Component Engineering Cycle in which the benefits and costs of the existing reusable components or the COTS component are calculated. For each reusable component or COTS component, it corresponds to a Component Engineering Cycle.
- Application Engineering Cycle, in which Application engineers use the retrieved COTS or reusable Components. The application engineers put efforts on tailoring and gluing code. In addition, covering the cost of system volatility is also the issue that application engineers have to face. In this cycle, cost factor of benefits due to shortened TTM is calculated through TTM submodel. TTM submodel covers the costs and benefits of shortened TTM due to the use of core assets. They are increased sales revenues and stock value. By incorporating the benefits of shortened TTM into application engineering cycle, the corporate will make more reasonable decisions based on a long term point of view.

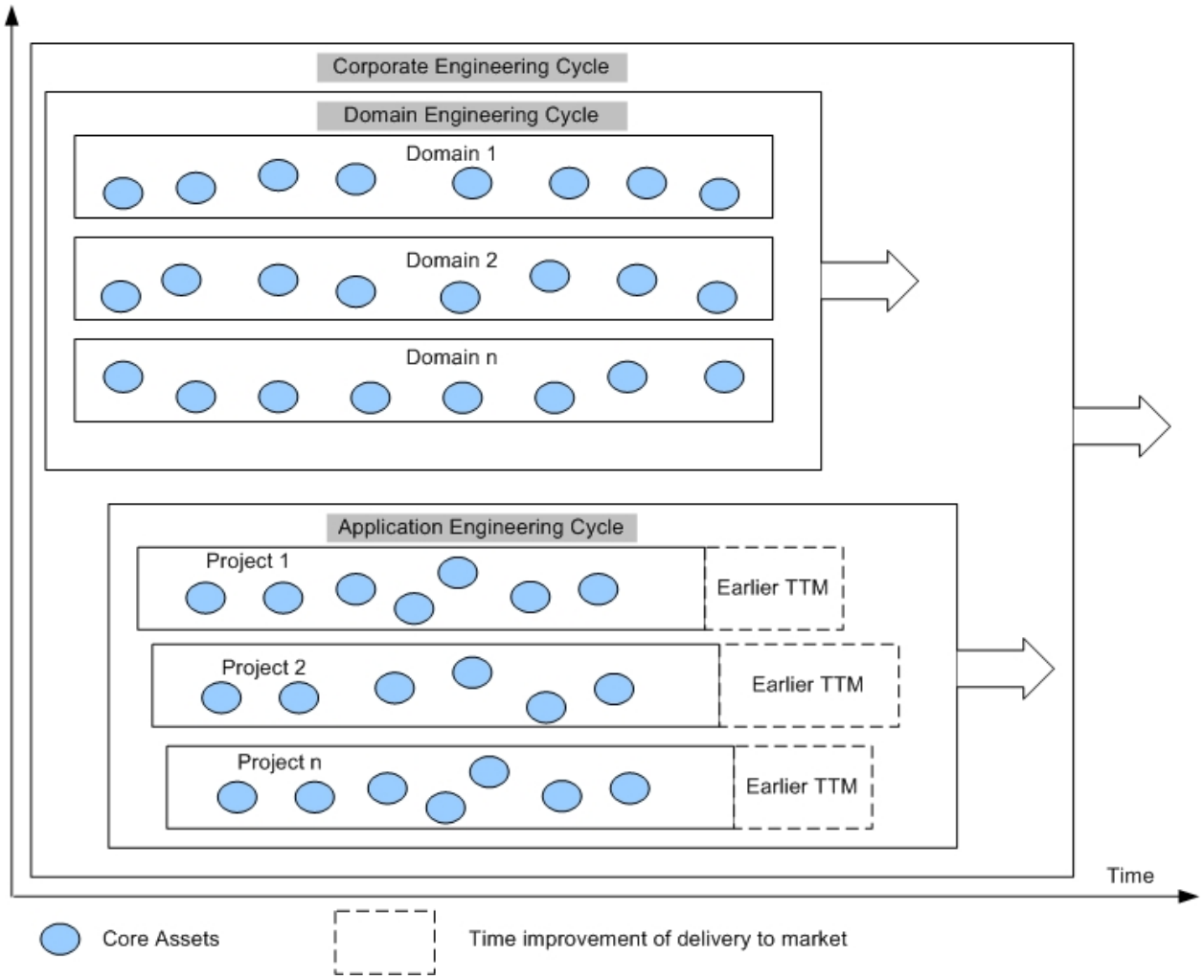

Figure 10: Integrated model extended to COTS, PLE and TTM

As Figure 10 indicates, life cycles of projects can be shortened by utilizing core assets from domains in domain engineering cycle. The benefits of earlier TTM are handled in TTM submodel section as showed in Figure 11.

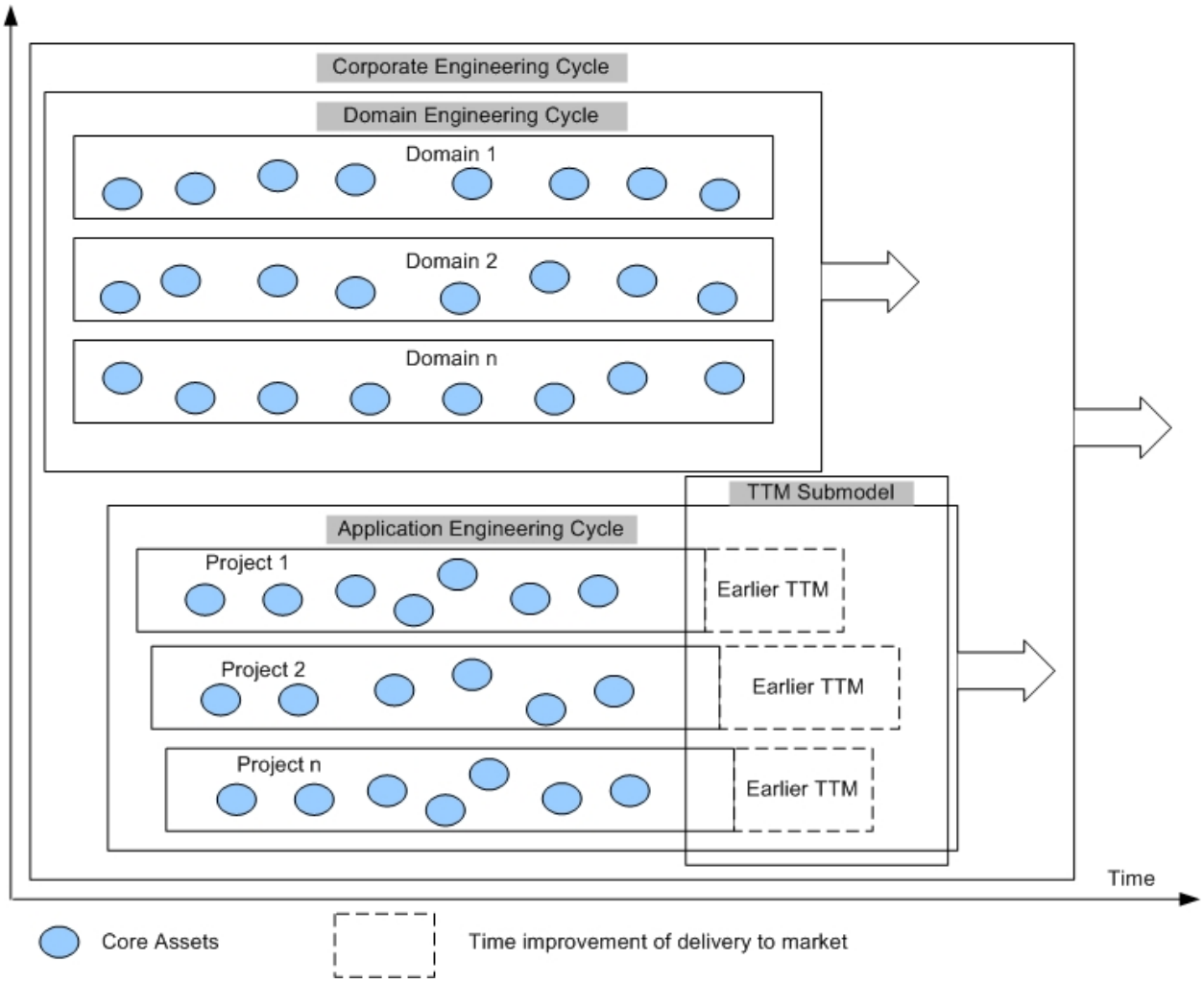

Figure 11: Integrated model extended to COTS, PLE and TTM

The integrated cost model can cover COTS, PLE and TTM with projects duration of several years long. It can treat long term project and incorporates the benefits of shortened TTM. Final NPV and ROI of corporate engineering cycle are calculated plus considering benefits due to shortened TTM derived from TTM submodel. The economic functions are used for corporation management to make final decision of initiating reuse activities. The quantifications are detailed in the following sections.

5.2 Component Engineering Cycle

The stakeholders in this cycle are component development engineers. There are three ways of core assets imported into domain and each core asset corresponds to a component engineering cycle.

- Develop from scratch
- Mine legacy software inside corporation
- Buy from market, COTS

And each component engineering cycle has a years-duration investment cycle. The ways of the core asset imported into domain carry different set of cost and benefit parameters.

5.2.1 Initial costs (y = starting year (SY))

The costs occur at the starting year of the investment cycle.

5.2.1.1 Mine legacy software (Mine inside corporation)

The costs are from:

- **Assessment Cost**: Analysis and selection efforts are spent to select and identify reusable components from existing resources inside corporation.
- **Library Insertion**: cost for a component to be inserted into reusable library.

Therefore, the total initial costs are:

$$IC[COEC] = AC + LI$$

Where

IC[COEC]: Initial Costs of Component Engineering Cycle
AC: Assessment Cost
LI: Library Insertion

5.2.1.2 Develop from scratch

The costs are from:

- **Development Cost**: cost spent to develop the component for reuse
- **Library Insertion**: cost for a component to be inserted into reusable library.

Therefore, the total initial costs are:

$$IC[COEC] = ER + LI$$

Where

IC[COEC]: Initial Costs of Component Engineering Cycle
ER: Development Cost for reuse
LI: Library Insertion

If ER is not available, the development cost can be derived by using the following equation derived from COCOMO [COCOMO 1994].

$$ER = RCWR(E)$$

$$E = a(S^b)$$

Where

ER: Development Cost for reuse
E: cost of development from scratch for single use
RCWR: Relative Cost of Writing for Reuse
a, b: COCOMO parameter, dependent on project characteristics
S: Size of component in thousand lines of code

5.2.1.3 COTS

For the components, which are purchased from market, the initial costs are composed of:

- **Assessment Cost**: Analysis and selection efforts are spent to select and identify reusable components from market.
- **Purchase Cost**: the cost of a COTS component. The cost occurs at the beginning of the project to get the reusable asset. The corporate pays to purchase COTS component.
- **Library Insertion**: cost for a component to be inserted into reusable library.

Therefore, the total initial costs are:

$$IC[COEC] = AC + Purchase + LI$$

Where

IC[COEC]: Initial Costs of Component Engineering Cycle
AC: Assessment Cost
Purchase: price paid to purchase COTS component from Vendor
LI: Library Insertion

5.2.2 Annual costs (SY<y≤SY+Y-1)

The costs occur at the years after the first year of the investment cycle, which include operating the reuse library and maintenance cost of the components.

Y: the number of years of the cycle.

5.2.2.1 Mine legacy software and Develop from scratch

- **Maintenance**: The maintenance costs can be derived from the COCOMO maintenance-effort equations. Annual Change Traffic (ACT) is the ratio of the yearly maintenance cost to the development cost and is typically around 0.15 [Boehm 1981] since annual maintenance is 15% of the cost of development from scratch for single use. Annual Change Traffic for reusable assets (ACT') is around 0.09, because the operating cost differential due to software reuse is 9% of the development effort [Chmiel 2000
- **Cost of operating the reuse library**: the cost of operating the reuse library prorated to a single component. The operating costs can be derived from dividing the labor costs of operating the library by the size of the library. For example, there are total *n* reusable components in library and the cost of operating one component is the total cost divided by *n*.

Therefore, the total annual costs are:

$$AC[COEC](y) = MN(y) + OC(y)$$

$$MN(y) = 0.09(E)$$

$$OC(y) = \frac{L}{n}$$

Where

AC[COEC](y): Annual Costs of Component Engineering Cycle at year y
MN(y): Maintenance cost at year y
OC(y): Operating Cost at year y
E: cost of development from scratch for single use
L: Number of librarians
n: Total number of reusable assets in library

5.2.2.2 COTS

By using COTS components, corporation can save efforts and costs on maintenance. Harris states that the software maintenance costs for internally developed software are about 55% of the total cost of the product life cycle. This includes costs associated with fixing defects as well as implementing enhancements [Harris 1992]. NASA Manager's Handbook for Software Development states that the maintenance can be 70% [Honeywell 1996].

For example, a project lasts four years, assuming that the maintenance cost is evenly distributed over the four-year life cycle. The project plans to purchase a GUI COTS component instead of developing from scratch. Suppose the actual costs to design, implement, test, and integrate the GUI module is $50,000. The estimated maintenance cost is

$$C_M = (0.55/0.45)(\$50{,}000) = \$61{,}111.11$$

The maintenance cost of each year would be ($61,111.11)/4 = $15,277.78

Therefore, the NPV of the saved maintenance cost at 10% discount rate is

$$Saved\ Cost = \sum_{n=0}^{3} \frac{\$15{,}277.78}{(1+.10)^n} = \$53{,}271.36$$

Therefore, the annual costs of a COTS component are composed of:

- **Annual license fee**: the cost to pay annual license for the COTS component if applied.
- **Cost of operating the reuse library**: the cost of operating the reuse library prorated to a single component. The operating costs can be derived from dividing the labor costs of operating the library by the size of the library. For example, there are total *n* reusable components in library and the cost of operating one component is the total cost divided by *n*.

Therefore, the total annual costs are:

$$AC[COEC](y) = AL + OC(y)$$

$$OC(y) = \frac{L}{n}$$

Where

AC[COEC](y): Annual Costs of Component Engineering Cycle at year y
AL: Annual license fee
OC(y): Operating Cost at year y
L: Number of librarians
n: Total number of reusable assets in library

5.2.3 Initial Benefits (y=SY)

The benefits occur at the starting year of the investment cycle.

5.2.3.1 Develop from scratch

There is no initial benefit for this kind of core asset.

$$IB[COEC\ ] = 0$$

Where

IB[COEC]: Initial benefits of Component Engineering Cycle

5.2.3.2 Mine legacy software and COTS

The initial benefits for these two are development costs are saved. This benefit results from the labor cost saved in Component Engineering Cycle by mining existing components or purchasing COTS components. Therefore, the initial benefits of Component Engineering Cycle are:

$$IB[COEC\ ] = ER$$

$$ER = RCWR\ (a)(S^{b})$$

Where

IB[COEC]: Initial benefits of Component Engineering Cycle
ER: Development Cost for reuse

5.2.4 Annual Benefits (SY<y≤SY+Y)

The benefits occur at the years after the first year of the investment cycle. Only if the core assets are utilized by application engineering cycle, the value of the core assets can be realized. Otherwise, costs to assess, develop, or purchase these assets won't get paid back.

Here are three main factors determining how much benefit these core assets can bring.

- Relative Black Box (RBP): price of black box use selling to application engineers. This study uses the default value Chmiel provides, 0.4, or it can be set in the supporting tool. COTS components are utilized this way.
- Relative White Box (RWP): price of white box use selling to application engineers. This study uses the default value Chmiel provides, 0.15, or it can be set in the supporting tool.

5.2.4.1 Develop from scratch and mine legacy software

$$AB[COEC](y) = \text{FreqB(y)(RBP)E} + \text{FreqW(y)(RWP)E}$$

$$E = a(S^b)$$

Where

AB[COEC]: Annual benefit of Component Engineering Cycle at year y
FreqB(y): Frequency of black box reuse of the component at year y
FreqW(y): Frequency of white box reuse of the component at year y
RBP: Relative Black Box
RWP: Relative White Box
S: Size of component in thousand lines of code
a, b: COCOMO parameters

5.2.4.2 COTS

$$AB[COEC](y) = \text{FreqB(y)(RBP)(E)}$$

$$E = a(S^b)$$

Where

AB[COEC]: Annual benefit of Component Engineering Cycle at year y
FreqB(y): Frequency of black box reuse of the component at year y
RBP: Relative Black Box
S: Size of COTS component in thousand lines of code

a, b: COCOMO parameters

5.3 Domain Engineering Cycle

In Domain Engineering Cycle, domain engineers define the domain characteristics, generic attributes and architecture, collect data, analyze domain artifact for its reusability, define reusability guidelines, and analyze reuse structure.

5.3.1 Costs (SY≤y≤SY+Y)

- **Domain analysis cost**: analysis of reuse structure and architecture
- **Costs cascaded** from costs in Component Engineering Cycle

The annual costs in Domain Engineering Cycle are cascaded from Component Engineering Cycles.

$$C[DEC](y) = \sum_{i=1}^{m} C[DOM]_i(y)$$

$$C[DOM]_i(y) = DA_i + \sum_{j=1}^{n} C[COEC]_j(y)$$

Where

AC[DEC](y): Annual Costs of Domain Engineering cycle at year y
$C[DOM]_i(SY)$: cost of domain *i* at year y
DA_i: Domain Analysis costs of domain *i*
$C[COEC]_j(SY)$: cost of component *j* at year y
m: number of domains in the domain engineering cycle
n: number of components in the domain

5.3.2 Benefits (SY≤y≤SY+Y)

The benefits in Domain Engineering Cycle are cascaded from Component Engineering Cycles.

$$B[DEC](y) = \sum_{i=1}^{m} B[DOM]_i(y)$$

$$B[DOM](y) = \sum_{j=1}^{n} B[COEC]_j(y)$$

Where:

B[DEC](y): Benefits of Domain Engineering Cycle at year y
$B[DOM]_i(y)$: Benefit of domain *i* at year y
$B[COEC]_j(y)$: Benefit of component *j* at year y
m: number of domains in the domain engineering cycle
n: number of components in the domain

5.4 Application Engineering Cycle

In Application Engineering Cycle, application engineers pay extra more efforts on reuse adoption, tailoring, glue code, and system volatility due to the use of core assets. The cost factor of benefit of shortened TTM is also incorporated in this cycle.

5.4.1 Initial costs (y=SY)

5.4.1.1 Develop from scratch and mine legacy software

- Cost of acquiring black box assets for the application
- Cost of acquiring white box assets for the application
- Cost of EDSI
- Cost of glue code

Cost of EDSI and glue code can be derived from COCOMO effort equation as depicted in Chapter 3.

$$IC[PRJ] = RBP \times \sum_{j=1}^{m} E_j + \mathrm{R}WP \times \sum_{k=1}^{n} E_k + a \times \left(\sum_{i=1}^{m+n} (EDSI_i + GC_i) \right)^b$$

Where

IC[PRJ]: Initial Costs of a project
m: number of mine-inside or develop from scratch components black-box-used in a project
n: number of mine-inside or develop from scratch components white-box-used in project
GC: Glue Code size

5.4.1.2 COTS

The initial costs include:

- Cost of acquiring COTS assets for the application
- Tailoring Effort
- Glue code Effort

Therefore, the initial costs are:

$$IC[PRJ] = \mathrm{R}BP \times \sum_{\mathrm{i}=1}^{\mathrm{n}} E_i + \frac{73.2}{45.5} \times \sum_{i=1}^{n} GE_i$$

Where

IC[PRJ]: Initial Costs of a project
n: number of COTS components in project
GC: Glue Code size
GE: glue code effort

$$GE = A \times [Size]^B \times \prod_{i=1}^{14} EM_i$$

$$A = 12.0$$

$$B = 1.00 + (0.04 \times SF)$$

$$Size = KSLOC \times \left(1 + \frac{Brak}{100}\right)$$

5.4.2 Annual costs (SY<y≤SY+Y)

For components of develop from scratch and mine legacy software, there are no periodic costs. For COTS, volatility cost occurs. Volatility in this context refers to the frequency with which new versions or updates of the COTS software being used in a larger system are released by the vendors over the period of the project development. The vendors usually release new versions every 10 months and this study assumes an annual COTS volatility in each calendar year. This study assumes that the volatility cost is evenly distributed over the Y-year life cycle.

Following is a table showing COTS efforts distribution [Madachy 2004].

COTS development phase	Efforts percentage (%)
Assessment	7.8
Tailoring	27.7
Glue code	45.5
Volatility	19.0

Table 7: COTS efforts distribution

Therefore, the volatility cost is calculated as followed:

$$VC = \frac{19}{45.5} \times GE$$

Where

VC: Volatility Cost
GE: Glue code cost

Therefore,

$$AC[PRJ](y) = \left[\sum_{i=1}^{n} VC_i \right] \times \frac{1}{Y}$$

$$Y = PD - Floor\,[STTM\,/12]$$

Where

AC[PRJ]: Annual Cost of a project
n: number of components
Y: Actual number of years of the project
PD: Planned duration of the project in year
STTM: earlier time to market in month, details covered in TTM submodel section.

5.4.3 Initial benefits (y=SY)

The initial benefits are due to development cost saved by utilizing core assets.

$$IB[PRJ] = a\left(\sum_{i=1}^{n} S_i \right)^b$$

Where

IB[AEC]: Initial benefits of Application Engineering Cycle
S: size of a component in K
n: number of components

5.4.4 Annual benefits (SY<y≤SY+Y)

The annual benefits are due to quality gains. This assumes that a component developed with reuse (especially if it is a black box component) is of better quality than a component developed from scratch for single use since it has been designed more carefully, tested more thoroughly, and documented more precisely [Chmiel 2000].

Since ACT is around 0.15 and Annual Change Traffic for reusable assets (ACT') around 0.09, because the operating cost differential due to software reuse is 9% of the development effort [Chmiel 2000]. Therefore, the quality gains are:

$$(ACT - ACT') \times E$$

For COTS components, there are no efforts on maintenance.

Therefore,

$$AB[PRJ] = (ACT - ACT') \times a\left(\sum_{i=1}^{n} S_i\right)^b \qquad (SY<y<SY+Y)$$

$$AB[PRJ] = (ACT - ACT') \times a\left(\sum_{i=1}^{n} S_i\right)^b + RI \quad (y=SY+Y)$$

Where

AB[PRJ]: Annual benefits of a project
n: number of components
RI: Revenue Improvement. More sales revenues are gained and RI is derived from TTM submodel.

This study assumes that TTM of the project can be shortened by utilizing core assets instead of developing from scratch for application engineers. Thus, as depicted in previous chapter, more sales revenues are gained in the earlier TTM period. The benefits of TTM are detailed in TTM submodel section.

5.4.5 Sum of costs and benefits in AEC (SY≤y≤SY+Y)

The costs and benefits in AEC are cascaded from all of the projects being carried on in the cycle.

$$C[AEC](y) = \sum_{i=1}^{n} C[PRJ]_i(y)$$

$$B[AEC](y) = \sum_{i=1}^{n} B[PRJ]_i(y)$$

Where:

C[AEC](y): Costs of Application Engineering Cycle at year y
B[AEC](y): Benefits of Application Engineering Cycle at year y
$C[PRJ]_i(y)$: Cost of project *i* at year y
$B[PRJ]_i(y)$: Benefit of project *i* at year y
n: number of projects in AEC

5.4.6 TTM submodel

As discussed in previous chapter, Application engineers will get rewards from more market sales after the project is done and the product goes to public. The task of this submodel is to quantify these rewards.

5.4.6.1 Benefits of shortened TTM

The benefits are gained outside the application cycle and gets reward after the project is done and product is delivered to market.

- **Shortened TTM (STTM)**: This benefit is the total time saved of the whole life cycle of the project. This factor will be used to calculate how much more sales revenues to achieve with shortened TTM for this product.
- **Revenue Improvement (RI)**: More market share and more sales revenues. If the product goes to public earlier, it will obtain more market share and more sales revenues; especially there are no competing products.
- **Stock Improvement (SI)**: Increase or decrease of stock value of the corporation. Stock dealers and analyzers give more credits based on the better performance and balance sheet of a corporation, since using COTS saves development cost with long term run and a better cash flow shows up on balance sheet. Customers like to see the new products going to public and it would be better to satisfy the stock holders with earlier TTM. And earlier TTM can result in more sales revenues and then makes the project a positive. If a firm takes on a project with positive NPV, the wealth of the stockholders is improved [Brigham 1999]. Therefore, the share value increases.

Following is a detailed description on how to quantify these elements that are composing the whole costs and benefits due to shortened TTM.

5.4.6.2 Quantify benefits of TTM

This submodel attempts to quantify the effects of shortened TTM due to use of core assets. As described in last section, the related cost and benefit factors are taken into account and estimated in a quantitative approach.

Shortened TTM (STTM)

By using existing core assets instead from developing from scratch, the application engineering cycle saves. This calculation is based on the equations from intermediate COCOMO, which calculate labor and time effort of development [COCOMO 1994].

$$TDEV = 2.5 \times PM^{c}$$

Where

TDEV: Effort and development time
PM: Person Month

This equation estimates how much time to spend to develop with a unit in month. Therefore, the shortened TTM can be derived as followed:

$$STTM = T_1 - T_2$$

$$T_1 = 2.5 \times \left[a \times \left[\sum_{i=1}^{n} S_i \right]^{b} \right]^{c}$$

$$T_2 = 2.5 \times \left[a \times \left[\sum_{i=1}^{n} EDSI_i + \sum_{i=1}^{n} GC_i \right]^{b} \right]^{c}$$

T_1: Time to develop from scratch
T_2: Time on efforts with reuse
S: size of a component in K
EDSI: Equivalent delivered source instructions in K
GC: glue code size in K
n: number of components reused

Revenue Improvement (RI)

This benefit results from more market share and sales revenues due to shortened TTM.

$$RI = ASR \times (STTM / 12)$$

Where

RI: Revenue Improvement
ASR: Annual Sales Revenues
STTM: Shortened TTM, total time saved in month due to use of reusable components

$$ASR = NC \times USP$$

Where

NC: estimated number of copies of the product sold annually on market
USP: Unit Sale Price

$$USP = BC \times \left(\frac{DC}{NC} \right)$$

Where

BC: benefit coefficient
DC: development cost
NC: estimated number of copies of the product sold annually in market

$$DC = \sum_{y=0}^{Y-1} \frac{C(y = (SY + y))}{(1+d)^y}$$

Where

SY: starting year
d: discount rate
Y: duration of project, number of years

Therefore,

$$RI = BC\,(DC)(STTM \ / 12)$$

Benefit factor is a coefficient that the corporation set up for product price to make profit on market. Before the corporation can decide upon a fair price for product, it needs to know how much it's costing

Once development costs are identified, the corporation can determine your break-even point. This is the point at which te neither make nor lose money in producing a product. For example, it would be at the break-even point if it cost $100 to produce a product that it sells for $100.

Stock Improvement (SI)

This benefit results from possible increase of stock value of the corporation as EVA theory indicates. The positive NPV pf a project increases the market value of a corporation.

$$SI = \frac{NPV\ (PRJ\ )}{TS}$$

Where

SI: stock improvement
NPV(PRJ): Net present value of project
TS: Total number of Shares

5.4.6.3 TTM Conclusion

Therefore, the total benefits due to shortened TTM are calculated below:

$$BTTM = \sum_{i=1}^{n} RI_i \cup \sum_{i=1}^{n} SI_i$$

Where

BTTM: Benefits of shortened TTM
RI_i: Revenue improvement of project *i*
SI_i: Stock improvement of project *i*
n: number of projects in AEC

5.5 Corporate Engineering Cycle

Corporate Engineering Cycle is the highest cycle in which a final decision is made based on all of the costs and benefits from other three engineering cycles and TTM submodel.

5.5.1 Initial Costs (y=SY)

The up-front costs in Corporate Engineering Cycle result from building reuse structure, training, other, and those cascaded from Domain and Application Engineering Cycles.

$$IC[CEC](SY) = IS + C[DEC](SY) + C[AEC](SY)$$

Where

IC[CEC]: Initial Costs of Corporate Engineering Cycle at year SY
IS: Infrastructure Costs
C[DEC](SY): Costs of DEC at year SY
C[AEC](SY): Costs of AEC at year SY
DEC: Domain Engineering Cycle
AEC: Application Engineering Cycle

5.5.2 Annual Costs (SY<y≤SY+Y)

$$AC[CEC](y) = C[DEC](y) + C[AEC](y)$$

Where

AC[CEC](y): Annual Costs of CEC at year y
C[DEC](y): Costs of DEC at year y
C[AEC](y): Costs of AEC at year y

5.5.3 Benefits (SY≤y≤SY+Y)

The benefits are cascaded from Domain and Application Engineering Cycle.

$$B[CEC](y) = B[DEC](y) + B[AEC](y)$$

Where

B[CEC](y): Benefits of CEC at year y
B[DEC](y): Benefits of DEC at year y

B[AEC](y): Benefits of AEC at year y

5.6 NPV and ROI

Here are the equations to calculate NPV and ROI for each cycle.

$$NPV(Cycle) = \sum_{y=0}^{Y-1} \frac{B[cycle](y) - C[Cycle](y)}{(1+d)^y}$$

$$ROI = \frac{NPV(Cycle)}{IC[Cycle]}$$

Where

Cycle: COEC, DEC, AEC, and CEC
B[Cycle](y): Benefits of cycle at year y
C[Cycle](y): Costs of cycle at year y
IC[Cycle](y): initial costs of cycle
Y: number of years of investment cycle
d: discount rate

Chapter 6 An Example of Application of Model

LinkySky Science and Technology, INC. is a high-tech company set up by returned overseas students. The company is located in Beijing, China. The company engages in network systems integration, systems design and implementation, as well as E-business.

In 2000, a domain is developed for reusable assets. From 2001 to 2003, the company developed two projects, which are based reusable components and share the same architecture. The projects characteristics are semi-detached. The average FTSP of the company is equivalent to 9K/year in US dollar. There is one software engineer responsible for reuse library. The discount rate is 10%. Since the company is not a public one, the data of number of shares of the company is not available.

6.1 Component Engineering Cycle

There are four reusable components in reuse library. One of them is mine legacy software, component 1. Two are develop from scratch, component 2 and 3. The left one is buy from market, component 4.

Component	Type	Size(in K)
Component 1	Mine	6.5
Component 2	Develop	12.5
Component 3	Develop	8
Component 4	Buy	22

6.1.1 Mine inside (component 1)

Component 1 has a size of 6.5K.

The initial cost includes
Assessment and identification cost (AC) is 5PM.
Library insertion cost (LI) is 2PM

$$E = a(S^b) = 3(6.5^{1.12}) = 24.41PM$$

Therefore, the initial cost for this component in 2001 is

$$C(y = 2001) = AC + LI = 5 + 2 = 7PM = 7(\frac{\$9k}{12}) = \$5{,}250$$

The annual costs include maintenance and operating reuse library after 2000.

$$C(y) = MN(y) + OC(y)$$

$$MN(y) = \frac{0.09 \times E \times FTSP}{12}$$

$$OC(y) = \frac{L \times FTSP}{n \times 12}$$

Where

Maintenance cost is

$$MN(y > 2001) = 0.09(E) = 0.09(24.41) = 3.66\ PM = 3.66(\frac{\$9k}{12}) = \$2{,}745$$

Operating reuse library cost is

$$OC(y > 2001) = \frac{1}{4} = 0.25\,PM = 0.25(\frac{\$9k}{12}) = \$375$$

Therefore, the annual cost is

$$C(y > 2001) = MN(y > 2000) + OC(y > 2000) = \$2{,}745 + \$375 = \$3{,}120$$

The initial benefit due to development cots saved is

$$B(y = 2000) = RCWR\ (E) = 1.5(E) = 1.5(24.41) = 36.62\ PM = 36.62(\frac{\$9K}{12}) = \$27{,}465$$

Componen 1		
Year	**FreqB**	**FreqW**
2000	0	0
2001	2	0
2002	0	0
2003	0	0

The table above shows the frequency of component 1, which is used as black box. The annual benefit is

$$B(y) = \text{FreqB(y)} \times \text{RBP} \times \text{E} + \text{FreqW(y)} \times \text{RWP} \times \text{E}$$

The benefit in year 2001 is

$$B(y = 2001) = 2(0.4)(24.41) = 19.53PM = 19.53(\frac{\$9K}{12}) = \$14{,}646$$

The benefit in year 2002 is

$$B(y = 2002) = 0$$

The benefit in year 2003 is

$$B(y = 2003) = 0$$

Component 1		
Year	**Cost**	**Benefit**
2000	$5,250	$27,465
2001	$3,120	$14,646
2002	$3,120	$0
2003	$3,120	$0

Following is the equation to calculate NPV of this cycle.

$$NPV = \sum_{y=0}^{Y=1} \frac{B(SY + y) - C(SY + y)}{(1 + d)^y}$$

NPV: net present value
SY: starting year
Y: duration of investment cycle in years
d: discount rate

Following is the equation to calculate ROI of this cycle.

$$ROI = \frac{NPV}{IC}$$

ROI: return on investment
IC: initial cost

Following is to calculate NPV of Component 1.

$$NPV = \frac{B(2000)-C(2000)}{(1+d)^0} + \frac{B(2001)-C(2001)}{(1+d)^1} + \frac{B(2002)-C(2002)}{(1+d)^2} + \frac{B(2003)-C(2003)}{(1+d)^3}$$
$$= \frac{\$27{,}465-\$5{,}250}{(1+0.1)^0} + \frac{\$14{,}646-\$3{,}120}{(1+0.1)^1} + \frac{0-\$3{,}120}{(1+0.1)^2} + \frac{0-\$3{,}120}{(1+0.1)^3}$$
$$= \$27{,}771$$

Following is to calculate ROI of Component 1.

$$ROI = \frac{NPV}{IC} = \frac{\$27{,}771}{\$5{,}250} = 5.29$$

6.1.2 Develop from scratch (component 2 and component 3)

Component 2 has a size of 12.5K and Component 3 has a size of 8K.

6.1.2.1 Component 2

The initial cost includes development cost and library insertion. The library insertion cost is 4PM.

The development cost is

$$ER = RCWR(a)(S)^b = 1.5(3)(12.5)^{1.12} = 1.5(50.78) = 76.17\,PM = 76.17(\frac{\$9K}{12}) = \$57{,}128$$

The library insertion cost is

$$LI = 4\,PM = 4(\frac{9K}{12}) = \$3{,}000$$

Therefore, the initial cost is

$$C(y = 2000) = ER + LI = \$57{,}128 + \$3{,}000 = \$60{,}128$$

The annual costs include maintenance and operating reuse library after 2000.

Maintenance cost is

$$MN(y > 2000) = 0.09(E) = 0.09(50.78) = 7.62PM = 7.62(\frac{\$9k}{12}) = \$5{,}715$$

Operating reuse library cost is

$$OC(y > 2000) = \frac{1}{4} = 0.25\,PM = 0.25(\frac{\$9k}{12}) = \$375$$

Therefore, the annual cost is

$$C(y > 2000) = MN(y > 2000) + OC(y > 2000) = \$5{,}715 + \$375 = \$6{,}090$$

Componen 2		
Year	**FreqB**	**FreqW**
2000	0	0
2001	2	1
2002	0	0
2003	0	0

The table above shows the frequency of component 2, which is used as black box and white box. The annual benefit is

$$B(y) = \text{FreqB(y)} \times \text{RBP} \times \text{E} + \text{FreqW(y)} \times \text{RWP} \times \text{E}$$

The benefit in year 2000 is

$$B(y = 2000) = 0$$

The benefit in year 2001 is

$$B(y = 2001) = 2(0.4)(50.78) + 1(0.15)(50.78) = 48.24PM = 48.24(\frac{\$9K}{12}) = \$36{,}180$$

The benefit in year 2002 is

$$B(y = 2002) = 0$$

The benefit in year 2002 is

$$B(y = 2003) = 0$$

Component 2		
Year	**Cost**	**Benefit**
2000	$60,128	0
2001	$6,090	$36,180
2002	$6,090	0
2003	$6,090	0

Following is to calculate NPV of Component 2.

$$NPV = \frac{B(2000) - C(2000)}{(1+d)^0} + \frac{B(2001) - C(2001)}{(1+d)^1} + \frac{B(2002) - C(2002)}{(1+d)^2} + \frac{B(2003) - C(2003)}{(1+d)^3}$$
$$= \frac{\$0 - \$60,218}{(1+0.1)^0} + \frac{\$36,180 - \$6,090}{(1+0.1)^1} + \frac{0 - \$6,090}{(1+0.1)^2} + \frac{0 - \$6,090}{(1+0.1)^3}$$
$$= -\$42,382$$

Following is to calculate ROI of Component 2.

$$ROI = \frac{NPV}{IC} = \frac{-\$42,382}{\$60,128} = -0.70$$

6.1.2.2 Component 3

The initial cost includes development cost and library insertion. The library insertion cost is 3PM.

Development cost is

$$ER = RCWR(a)(S)^b = 1.5(3)(8)^{1.12} = 1.5(30.8) = 46.2PM = 46.2(\frac{\$9K}{12}) = \$34,650$$

Library insertion cost is

$$LI = 3PM = 3(\frac{\$9K}{12}) = \$2,250$$

Therefore, the initial cost is

$$C(y = 2000) = ER + LI = \$34,650 + \$2,250 = \$36,900$$

The annual costs include maintenance and operating reuse library after 2001.

Maintenance cost is

$$MN(y > 2000) = 0.09(E) = 0.09(30.8) = 4.62PM = 4.62(\frac{\$9k}{12}) = \$3{,}465$$

Operating reuse library cost is

$$OC(y > 2000) = \frac{1}{4} = 0.25\,PM = 0.25(\frac{\$9k}{12}) = \$375$$

Therefore, the annual cost is

$$C(y > 2000) = MN(y > 2000) + OC(y > 2000) = \$3{,}465 + \$375 = \$3{,}840$$

Componen 3		
Year	**FreqB**	**FreqW**
2000	0	0
2001	2	1
2002	0	0
2003	0	0

The table above shows the frequency of component 3, which is used as black box and white box.

The annual benefit is

$$B(y) = \text{FreqB(y)} \times \text{RBP} \times \text{E} + \text{FreqW(y)} \times \text{RWP} \times \text{E}$$

The benefit in 2000 is

$$B(y = 2000) = 0$$

The benefit in 2001 is

$$B(y = 2001) = 2(0.4)(30.8) + 1(0.15)(30.8) = 29.26\,PM = 29.26(\frac{\$9K}{12}) = \$21{,}945$$

The benefit in 2002 is

$$B(y = 2002) = 0$$

The benefit in 2003 is

$$B(y = 2003) = 0$$

Component 3		
Year	**Cost**	**Benefit**
2000	$36,900	0
2001	$3,840	$21,945
2002	$3,840	0
2003	$3,840	0

Following is to calculate NPV of Component 3.

$$NPV = \frac{B(2000) - C(2000)}{(1+d)^0} + \frac{B(2001) - C(2001)}{(1+d)^1} + \frac{B(2002) - C(2002)}{(1+d)^2} + \frac{B(2003) - C(2003)}{(1+d)^3}$$
$$= \frac{\$0 - \$36,900}{(1+0.1)^0} + \frac{\$21,945 - \$3,840}{(1+0.1)^1} + \frac{0 - \$3,840}{(1+0.1)^2} + \frac{0 - \$3,840}{(1+0.1)^3}$$
$$= -\$26,500$$

Following is to calculate ROI of Component 3.

$$ROI = \frac{NPV}{IC} = \frac{-\$26,500}{\$36,900} = -0.72$$

6.1.3 Buy from market (component 4)

Component 4 has a size of 22K. The purchase fee is $2,230 and the annual license fee is zero.

The initial cost includes assessment, purchase and library insertion. The assessment is 8PM. The library insertion cost is 5PM.

$$\mathrm{E} = a(S)^b = 3(22)^{1.12} = 95.64\,PM$$

The initial cost is

$$C(y = 2000) = 8(\frac{\$9K}{12}) + \$2{,}230 + 5(\frac{\$9K}{12}) = \$6{,}000 + \$2{,}230 + \$3{,}750 = \$11{,}980$$

The annual cost includes annual license fee and operating reuse library.

$$C(y > 2000) = 0 + 0.25(\frac{\$9K}{12}) = \$188$$

The initial benefit due to development cost saved is

$$B(y = 2000) = RCWR(E) = 1.5(95.64) = 143.46\,PM = 143.36(\frac{\$9K}{12}) = \$107{,}520$$

Componen 4		
Year	**FreqB**	**FreqW**
2000	0	0
2001	2	0
2002	0	0
2003	0	0

The table above shows the frequency of component 4, which is used as black box and white box. The annual benefit is

$$B(y) = \text{FreqB(y)} \times \text{RBP} \times \text{E}$$

The benefit in 2001 is

$$B(y = 2001) = 2(0.4)(95.64) = 76.51PM = 76.51(\frac{\$9K}{12}) = \$57{,}383$$

The benefit in 2002 is

$$B(y = 2002) = 0$$

The benefit in 2003 is

$$B(y = 2003) = 0$$

Component 4		
Year	**Cost**	**Benefit**
2000	$11,980	$107,520
2001	$375	$57,383
2002	$188	$0
2003	$188	$0

Following is to calculate NPV of Component 4.

$$NPV = \frac{B(2000) - C(2000)}{(1+d)^0} + \frac{B(2001) - C(2001)}{(1+d)^1} + \frac{B(2002) - C(2002)}{(1+d)^2} + \frac{B(2003) - C(2003)}{(1+d)^3}$$
$$= \frac{\$107,520 - \$11,980}{(1+0.1)^0} + \frac{\$57,383 - \$375}{(1+0.1)^1} + \frac{0 - \$188}{(1+0.1)^2} + \frac{0 - \$188}{(1+0.1)^3}$$
$$= \$147,069$$

Following is to calculate ROI of Component 4.

$$ROI = \frac{NPV}{IC} = \frac{\$147,069}{\$11980} = 12.25$$

6.2 Domain Engineering Cycle

The costs of domain cycle include domain analysis and costs cascaded from component engineering cycles. The domain is built in 2000 and contains four reusable assets. The table below shows the cost and benefits of each component engineering cycle.

	Component 1		Component 2		Component 3		Component 4	
Year	**Cost**	**Benefit**	**Cost**	**Benefit**	**Cost**	**Benefit**	**Cost**	**Benefit**
2000	$5,250	$27,465	$60,128	0	$36,900	0	$11,980	$107,520
2001	$3,120	$14,646	$6,090	$36,180	$3,840	$21,945	$375	$57,383
2002	$3,120	$0	$6,090	0	$3,840	0	$188	$0
2003	$3,120	$0	$6,090	0	$3,840	0	$188	$0

The domain analysis cost is 30PM.

The cost in 2000 is

$$C(y = 2000) = 30(\frac{\$9K}{12}) + \$5{,}250 + \$60{,}128 + \$36{,}900 + \$11{,}980 = \$22{,}500 + \$114{,}258 = \$136{,}758$$

The cost in 2001 is

$$C(y = 2001) = \$3{,}120 + \$6{,}090 + \$3{,}840 + \$188 = \$13{,}238$$

The cost in 2002 is

$$C(y = 2002) = \$3{,}120 + \$6{,}090 + \$3{,}840 + \$188 = \$13{,}238$$

The cost in 2003 is

$$C(y = 2003) = \$3{,}120 + \$6{,}090 + \$3{,}840 + \$188 = \$13{,}238$$

The benefits of domain cycle are cascaded from component cycles.

The benefit in 2000 is

$$B(y = 2000) = \$27{,}465 + \$107{,}520 = \$134{,}985$$

The benefit in 2001 is

$$B(y = 2001) = \$14{,}646 + \$36{,}180 + \$21{,}945 + \$57{,}383 = \$130{,}154$$

The benefit in 2002 is

$$B(y = 2002) = 0$$

The benefit in 2003 is

$$B(y = 2003) = 0$$

Domain 1		
Year	**Cost**	**Benefit**
2000	$136,758	$134,985
2001	$13,238	$130,154
2002	$13,238	$0
2003	$13,238	$0

Following is to calculate NPV of Domain 1.

$$NPV = \frac{B(2000) - C(2000)}{(1+d)^0} + \frac{B(2001) - C(2001)}{(1+d)^1} + \frac{B(2002) - C(2002)}{(1+d)^2} + \frac{B(2003) - C(2003)}{(1+d)^3}$$
$$= \frac{\$134{,}985 - \$136{,}758}{(1+0.1)^0} + \frac{\$130{,}154 - \$13{,}238}{(1+0.1)^1} + \frac{0 - \$13{,}238}{(1+0.1)^2} + \frac{0 - \$13{,}238}{(1+0.1)^3}$$
$$= \$83{,}628$$

Following is to calculate ROI of Domain 1.

$$ROI = \frac{NPV}{IC} = \frac{\$83{,}628}{\$136{,}758} = 0.61$$

6.3 Application Engineering Cycle

The table below shows the basic info of each reusable component.

Component	Type	Size(in K)	E (PM)
Component 1	Mine	6.5	24.41
Component 2	Develop	12.5	50.78
Component 3	Develop	8	30.80
Component 4	Buy	22	95.64

There are two applications starting from 2001, application 1 and 2. The table below shows the frequency of each component used in each year and application.

	Component 1		Component 2		Component 3		Component 4	
Year	Black	White	Black	White	Black	White	Black	White
2000	0	0	0	0	0	0	0	0
2001	App1&2	0	App1&2	App 1	App1&2	App 2	App1&2	0
2002	0	0	0	0	0	0	0	0
2003	0	0	0	0	0	0	0	0

6.3.1 Application 1

The table below shows the components used.

Application 1							
2000		2001		2002		2003	
Black	White	Black	White	Black	White	Black	White
		Com 1	Com 2				
		Com 2					
		Com 3					
		Com 4					

The table below shows the glue code for each component.

Application 1			
Component	Type	Size(in K)	Glue Code
Component 1	Mine	6.5	4
Component 2	Develop	12.5	5
Component 3	Develop	8	3
Component 4	Buy	22	8

Cost to acquire component 1, 2 and 3 is

$$\frac{RBP \times FTSP}{12} \times \sum_{i=1}^{n} E_i$$

$$0.4(24.41 + 50.78 + 30.8) + 0.15(50.78) = 42.4 + 7.62 = 50.02(\frac{\$9K}{12}) = \$37{,}515$$

For the white box use of component 2,

DM	CM	IM
0	15	5

$$AAF = 0.4(DM) + 0.3(CM) + 0.3(IM)$$

$$EDSI = (ADSI)\frac{AAF}{100}$$

Therefore, the adaptation adjustment factor (AAF) is

$$\text{AAF} = 0.4(0) + 0.3(15) + 0.3(5) = 6$$

$$\text{EDSI} = 12.5(\frac{6}{100}) = 0.75K$$

Cost of EDSI and glue code in 2001 for component 1, 2 and 3 is

$$3(4+5+5+3+0.75)^{1.12} = 3(17.75)^{1.12} = 75.2PM = 75.2(\frac{\$9K}{12}) = \$56{,}401$$

Component 4 is COTS and glue code is 8K. The costs in 2001 include acquire reusable asset, tailoring and glue code. The following steps show the statistics to calculate.

The cost to acquire component from domain cycle is

$$0.4(3)(22)^{1.12} = 38.26PM = 38.26(\frac{\$9K}{12}) = \$28{,}692$$

Linear scaling factor A is a constant. The value is 12.0.
Non-linear scaling factor AAREN SF is 2.0.
Breakage is 0%.

The values of effort multipliers are

1	2	3	4	5	6	7	8	9	10	11	12	13	14
ACIEP	ACIPC	AXICP	APCON	ACPMT	ACSEW	APCPX	ACPPS	ACPTD	APVOL	ACREL	AACPX	ACPER	ASPRT
1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

$$Size = KSLOC \times \left(1 + \frac{Brak}{100}\right) = 8(1 + \frac{0}{100}) = 8K$$

$$B = 1.00 + (0.04 \times SF) = 1 + 0.04(2) = 1.08$$

Therefore, the glue code cost is

$$GE = A \times [Size]^B \times \prod_{i=1}^{14} EM_i = 12(8)^{1.08}(1) = 113.38PM = 113.38(\frac{\$9K}{12}) = \$85{,}031$$

The tailoring cost is

$$113.38(\frac{27.7}{45.5}) = 69.02PM = 69.02(\frac{\$9K}{12}) = \$51{,}769$$

Therefore, costs of component 4 in 2001 is

$$\$28{,}692 + \$85{,}031 + \$51{,}769 = \$165{,}492$$

Therefore, the cost of application 1 in 2001 is

$$C(y = 2001) = \$37{,}515 + \$56{,}401 + \$165{,}492 = \$253{,}850$$

Volatility cost in year after 2001 is

$$C(y = 2002) = (\frac{19}{45.5})(\frac{113.38}{2}) = 23.67PM = 23.67(\frac{\$9K}{12}) = \$17{,}755$$

$$C(y = 2003) = \$17{,}755$$

The benefits in 2001 are due to development cost saved by utilizing core assets.

$$B(y = 2001) = 3(6.5 + 12.5 + 12.5 + 8 + 22)^{1.12} = 3(61.5)^{1.12}$$

$$= 302.46PM = 302.46(\frac{\$9K}{12}) = \$226{,}845$$

Annual benefit after 2001 is

$$302.46(ACT - ACT') = 302.46(0.15 - 0.09) = 18.15PM = 18.15(\frac{\$9K}{12}) = \$13{,}613$$

Therefore,

$$B(y = 2002) = \$13{,}613$$

$$B(y = 2003) = \$13{,}613 + RI = \$13{,}613 + \$229{,}888 = \$243{,}501$$

Application 1		
Year	Cost	Benefit
2001	$253,850	$226,845
2002	$17,755	$13,613
2003	$17,755	$243,501

Incorporation of TTM into Application 1

Discount rate d is 10%.

The shortened TTM in calendar month is

$$SSTM = 2.5\left[3(6.5+12.5+12.5+8+22)^{1.12}\right]^{0.35} - 2.5\left[3(4+5+5+3+8+0.75)^{1.12}\right]^{0.35}$$
$$= 2.5(302.46)^{0.35} - 2.5(114.08)^{0.35} = 2.5(7.38-5.25) = 5.33\ Month$$

The cost of Application 1 is

$$DC = \frac{\$253,850}{(1+0.1)} + \frac{\$17,755}{(1+0.1)^2} + \frac{\$17,755}{(1+0.1)^3} = \$258,786$$

ASR is 5,000 copies. BC is 2.0. Therefore, revenue improvement is

$$RI = 2(\$258,786)(\frac{5.33}{12}) = \$229,888$$

Following is to calculate NPV of Application 1.

$$NPV = \frac{B(2001)-C(2001)}{(1+d)^1} + \frac{B(2002)-C(2002)}{(1+d)^2} + \frac{B(2003)-C(2003)}{(1+d)^3}$$
$$= \frac{\$226,845-\$253,850}{(1+0.1)^1} + \frac{\$13,613-\$17,755}{(1+0.1)^2} + \frac{\$243,501-\$17,755}{(1+0.1)^3}$$
$$= \$141,633$$

Following is to calculate ROI of Application 1.

$$ROI = \frac{NPV}{IC} = \frac{\$141{,}633}{\$253{,}850} = 0.56$$

6.3.2 Application 2

The table below shows the components used.

Application 2							
2000		2001		2002		2003	
Black	White	Black	White	Black	White	Black	White
		Com 1	Com 3				
		Com 2					
		Com 3					
		Com 4					

The table below shows the glue code for each component.

Application 2			
Component	Type	Size(in K)	Glue Code
Component 1	Mine	6.5	3
Component 2	Develop	12.5	4
Component 3	Develop	8 92)	3 (3)
Component 4	Buy	22	8

Cost to acquire component 1, 2 and 3 is

$$0.4(24.41 + 50.78 + 30.8) + 0.15(30.8) = 42.4 + 4.62 = 47.02(\frac{\$9K}{12}) = \$35{,}265$$

For the white box use of component 3

DM	CM	IM
0	20	10

Therefore, the adaptation adjustment factor (AAF) is

$$\text{AAF} = 0.4(0) + 0.3(20) + 0.3(10) = 16$$

$$\text{EDSI} = 12.5(\frac{16}{100}) = 2K$$

Cost of EDSI and glue code in 2001 for component 1, 2 and 3 is

$$3(3+4+3+3+2)^{1.12} = 3(15)^{1.12} = 75.2PM = 62.28(\frac{\$9K}{12}) = \$46{,}709$$

Component 4 is COTS and glue code is 8K. The costs in 2001 include acquire reusable asset, tailoring and glue code. The following steps show the statistics to calculate.

The cost to acquire component from domain cycle is

$$\text{RBP(a)(S)}^{\text{b}} = 0.4(3)(22)^{1.12} = 38.26PM = 38.26(\frac{\$9K}{12}) = \$28{,}692$$

Linear scaling factor A is a constant. The value is 12.0.
Non-linear scaling factor AAREN SF is 2.0.
Breakage is 0%.

The values of effort multipliers are

1	2	3	4	5	6	7	8	9	10	11	12	13	14
ACIEP	ACIPC	AXICP	APCON	ACPMT	ACSEW	APCPX	ACPPS	ACPTD	APVOL	ACREL	AACPX	ACPER	ASPRT
1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

$$Size = KSLOC \times \left(1 + \frac{Brak}{100}\right) = 8(1 + \frac{0}{100}) = 8K$$

$$B = 1.00 + (0.04 \times SF) = 1 + 0.04(2) = 1.08$$

Therefore, the glue code cost is

$$GE = A \times [Size]^{B} \times \prod_{i=1}^{14} EM_i = 12(8)^{1.08}(1) = 113.38PM = 113.38(\frac{\$9K}{12}) = \$85{,}031$$

The tailoring cost is

$$113.38(\frac{27.7}{45.5}) = 69.02PM = 69.02(\frac{\$9K}{12}) = \$51{,}769$$

Therefore, costs of component 4 in 2001 is

$$\$28{,}692 + \$85{,}031 + \$51{,}769 = \$165{,}492$$

For App 2, the cost in 2001 is

$$C(y = 2001) = \$35{,}265 + \$46{,}709 + \$165{,}492 = \$247{,}466$$

Volatility cost in year after 2001 is

$$C(y = 2002) = (\frac{19}{45.5})(\frac{113.38}{2}) = 23.67PM = 23.67(\frac{\$9K}{12}) = \$17{,}755$$

$$C(y = 2003) = \$17{,}755$$

The benefits in 2001 are due to development cost saved by utilizing core assets.

$$B(y = 2001) = 3(6.5 + 12.5 + 8 + 8 + 22)^{1.12} = 3(57)^{1.12}$$

$$= 277.78PM = 277.78(\frac{\$9K}{12}) = \$208{,}335$$

Annual benefit after 2001 is

$$277.78(ACT - ACT') = 277.78(0.15 - 0.09) = 16.67PM = 16.67(\frac{\$9K}{12}) = \$12{,}500$$

Therefore, the benefit in 2002 is

$B(y = 2002) = \$12{,}500$

The benefit in 2003 is

$B(y = 2003) = \$12{,}500 + RI = \$12{,}500 + \$226{,}841 = \$239{,}341$

Application 2		
Year	**Cost**	**Benefit**
2001	$247,466	$208,335
2002	$17,755	$12,500
2003	$17,755	$239,341

Incorporation of TTM into Application 2

Discount rate d is 10%.

Shortened TTM in calendar month is

$$SSTM = 2.5\left[3(6.5+12.5+8+8+22)^{1.12}\right]^{0.35} - 2.5\left[3(3+4+3+3+8+2)^{1.12}\right]^{0.35}$$
$$= 2.5(277.78)^{0.35} - 2.5(100.52)^{0.35} = 2.5(7.17-5.02) = 5.38\ Month$$

The cost of Application 2 is

$$DC = \frac{\$247{,}466}{(1+0.1)} + \frac{\$17{,}755}{(1+0.1)^2} + \frac{\$17{,}755}{(1+0.1)^3} = \$252{,}982$$

BC is 2.0. Therefore, the revenue improvement is

$$RI = 2(\$252{,}982)(\frac{5.38}{12}) = \$226{,}841$$

Following is to calculate NPV of Application 2.

$$NPV = \frac{B(2001) - C(2001)}{(1+d)^1} + \frac{B(2002) - C(2002)}{(1+d)^2} + \frac{B(2003) - C(2003)}{(1+d)^3}$$
$$= \frac{\$208{,}335 - \$247{,}466}{(1+0.1)^1} + \frac{\$12{,}500 - \$17{,}755}{(1+0.1)^2} + \frac{\$239{,}341 - \$17{,}755}{(1+0.1)^3}$$
$$= \$126{,}564$$

Following is to calculate ROI of Application 2.

$$ROI = \frac{NPV}{IC} = \frac{\$126{,}564}{\$247{,}466} = 0.51$$

6.4 Corporate engineering cycle

Infrastructure cost: $20,000
Training cost: $10,000
Operational impact: $5,000
Management restructuring: $3,000
Others: $2,000

Total is $40,000.

The balance of domain engineering cycle is

Domain 1		
Year	Cost	Benefit
2000	$136,758	$134,985
2001	$13,238	$130,154
2002	$13,238	$0
2003	$13,238	$0

The balance of Application 1 is

Application 1		
Year	Cost	Benefit
2001	$253,850	$226,845
2002	$17,755	$13,613
2003	$17,755	$243,501

The balance of Application 2 is

Application 2		
Year	Cost	Benefit
2001	$247,466	$208,335
2002	$17,755	$12,500
2003	$17,755	$239,341

The balance of corporate is

Corporate Cycle		
Year	**Cost**	**Benefit**
2000	$176,758	$134,985
2001	$514,554	$565,334
2002	$48,748	$26,113
2003	$48,748	$482,842

Following is to calculate NPV of Corporate 1.

$$NPV = \frac{B(2000) - C(2000)}{(1+d)^0} + \frac{B(2001) - C(2001)}{(1+d)^1} + \frac{B(2002) - C(2002)}{(1+d)^2} + \frac{B(2003) - C(2003)}{(1+d)^3}$$
$$= \frac{\$134{,}985 - \$176{,}758}{(1+0.1)^0} + \frac{\$565{,}334 - \$514{,}554}{(1+0.1)^1} + \frac{\$26{,}113 - \$48{,}748}{(1+0.1)^2} + \frac{\$482{,}842 - \$48{,}748}{(1+0.1)^3}$$
$$= \$311{,}825$$

Following is to calculate ROI of Corporate 1.

$$ROI = \frac{NPV}{IC} = \frac{\$311{,}825}{\$176{,}758} = 1.76$$

7 Optimal Corporate ROI

There are four stakeholders: Corporate, Domain, Application and Component and each has a ROI. In order to optimize the corporate ROI, all of the four ROI's should be positive. In this model, all of costs and benefits are finally cascaded into corporate engineering cycle as followed:

	Investment Costs	**Episodic Costs**	**Episodic Benefits**
Corporate Engineering Cycle (CEC)	Infrastructure costs;	From DEC;	From AEC;
Application Engineering Cycle (AEC)	Training; Process impact;	Reuse overhead; Reuse risks;	Productivity gains; Quality gains;
Domain Engineering Cycle (DEC)	Domain analysis;	From COEC	Cost transfers from project;
Component Engineering Cycle (COEC)	Development for reuse;	Library overhead;	Sale of asset;

Table 8: Costs and Benefits cascade pattern in the model

We want to optimize (maximize) the corporate ROI under the condition that all four ROI's are positive. That means in order to maximize corporate ROI, at the same time ROI's of component cycle, domain cycle, and application cycle are positive. There is a pair of RBP and RWP values which can meet these criterions. By designing an algorithm and applying it to the data, we can discover this pair of values to make the maximized value of corporate ROI.

7.1 Algorithm design

The idea of this algorithm is that by looping through RBP and RWP intervals and checking if all of the ROI's of the four cycles are positive.

1. If the pair of values makes a component engineering cycle a negative ROI, then the loop walks out to next pair of RBP and RWP values;
2. Otherwise, the pair of values continues to check ROI of domain engineering cycle. If the pair of values makes a domain engineering cycle a negative ROI, then the loop walks out to next pair of RBP and RWP values.
3. Otherwise, the pair of values continues to check ROI of application engineering cycle. If the pair of values makes an application engineering cycle a negative ROI, then the loop walks out to next pair of RBP and RWP values.
4. Otherwise, the pair of values continues to check ROI of corporate engineering cycle. If the pair of values makes a corporate engineering cycle a negative ROI, then the loop walks out to next pair of RBP and RWP values.
5. And there is a data structure to hold the pairs of values of RBP and RWP, which makes all of the four cycles positive ROI's, and the values of ROI's.
6. In the end of the algorithm, loop through the data structure and find the maximized corporate ROI.

Following is the pseud code of the algorithm.

```
/data structure which holds values of RBP, RWP and ROI
Structure corROI
        RBP as double
        RWP as Double
        ROI as Double
End structure

/function which returns value of ROI of component engineering cycle
Function Com_ROI (String Component_Name, Double RBP, Double RWP)
return Double Component_ROI;

/function which returns value of ROI of domain engineering cycle
Function Dom_ROI (String Domain_Name)
return Double Domain_ROI;

/function which returns value of ROI of application engineering cycle
Function App_ROI (String Application_Name, Double RBP, Double RWP)
return Double Application_ROI;

/function which returns value of ROI of corporate engineering cycle
Function Cor_ROI (String Corporation_Name)
return Double Corporation_ROI;

/array of corROI structure which holds ROI values
corROIArray(200) As corROI

flag = "Positive"
```

```
While i <= 0.8
        While j<=0.33
                While m < comNameArray.Length
                        If Com_ROI(comNameArray(m), i, j) < 0 Then
                                flag = "Negative"
                         End If
                         m = m + 1
                End while

         If flag = "Positive" Then
               m = 0
               While m < domainNameArray.Length

                  If Dom_ROI(domainNameArray(m)) < 0 Then
                     flag = "Negative"
                  End If
                  m = m + 1
               End While
        End If

        If flag = "Positive" Then
               m = 0
               While m < appNameArray.Length
                  If App_ROI(appNameArray(m), i, j) < 0 Then
                     flag = "Negative"
                  End If
                  m = m + 1
               End While
        End If

        If flag = "Positive" Then
               If Cor_ROI(corporation name) < 0 Then
                  flag = "Negative"
               End If
        End If

        If flag = "Positive" Then
               corROIArray(k).RBP = i
               corROIArray(k).RWP = j
               corROIArray(k).ROI = Cor_ROI(corporation name)
        End If

        j = j + 0.05
        k = k + 1
    End While
```

```
    i = i + 0.1
End While

 While i < corROIArray.Length

        If corROIArray(i).ROI <= corROIArray(i + 1).ROI Then
           max = corROIArray(i + 1).ROI
           j = j + 1
        End If
        i = i + 1
End While

Max ROI = corROIArray(j).ROI
RBP = corROIArray(j).RBP
RWP = corROIArray(j).RWP
```

7.2 Algorithm results

Following is the results by running this algorithm in the supporting tool.

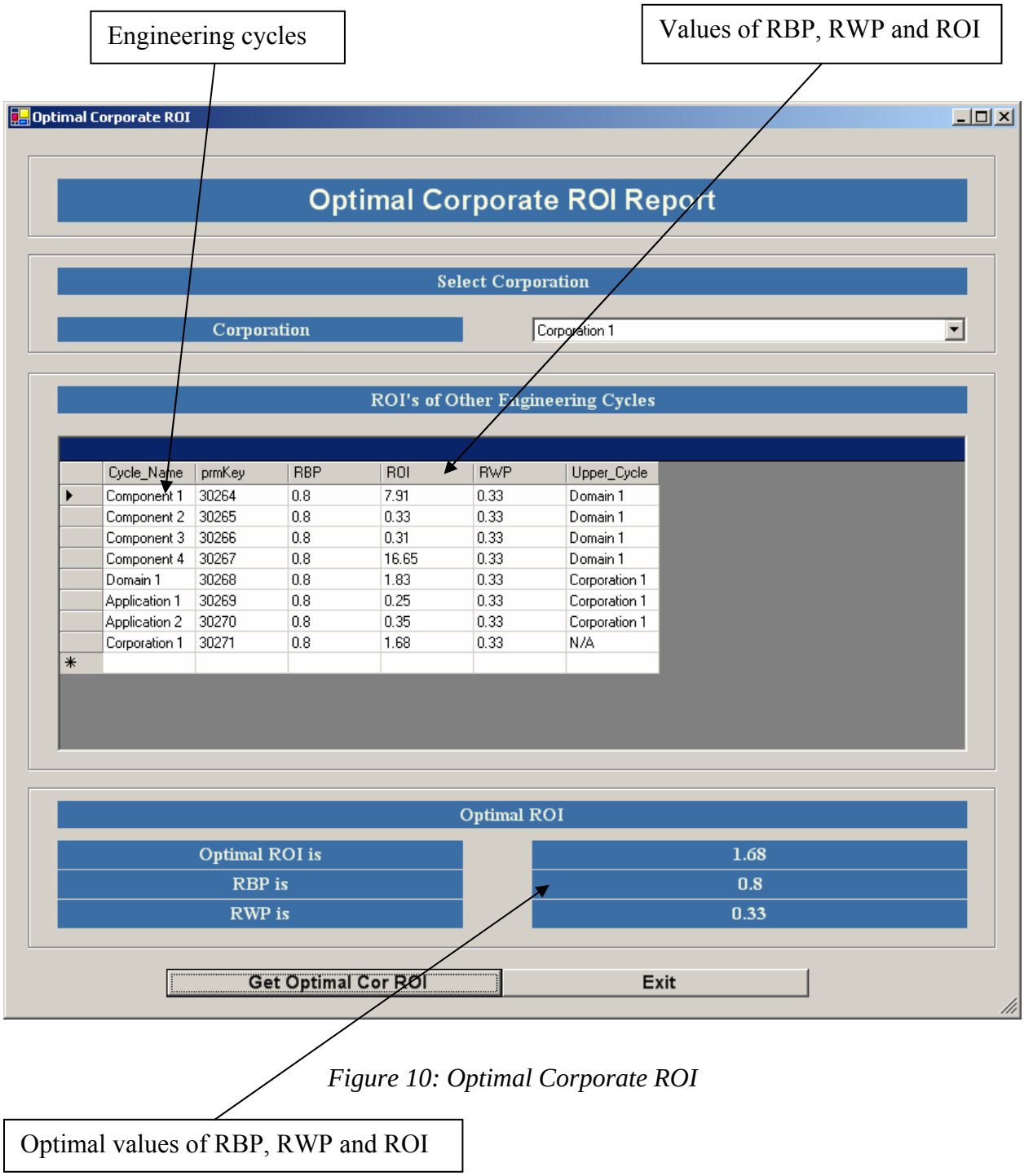

Figure 10: Optimal Corporate ROI

7.3 Optimal corporate ROI analysis

ROI's of Component 2 and Component 3 are negative as depicted in previous sections with default RBP 0.40 and RWP 0.15. By increasing the value of RBP (Relative Black box Price) from 0.4 to 0.8 and RWP (Relative White box Price) from 0.13 to 0.3 in favor of Component Cycle and Domain Cycle, the benefits of these two cycles increase. Therefore, the NPV and ROI of these two cycles also increase

Since the domain engineer must fix the price of the asset so as to maximize his benefit per sale and maximize its ROI on domain engineering cycle, while making sure the acquisition of the asset is still attractive to the application engineer (otherwise the application engineering will develop his own). The changing gives engineers in domain and component engineers confidence to work on this reuse project. By increasing values of RBP and RWP, the result shows that all of the four ROI's are positive.

The cascading of costs and benefits in thsi model is the base of the rationale. By maximizing the ROI at the component, domain, and application engineering cycle levels, the maximum benefit can be achieved at the corporate level. To maximize the ROI at the corporate level, one correspondingly must maximize the ROI of Domain, Component, and Application Engineering Cycle. If the result of ROI at the component level is negative, the ROI's at the Domain and Application level need to be non-negative for the investment to be worthwhile at the corporate level.

Chapter 8 Reuse Expert: the Supporting Tool

8.1 Introduction

The customers to this supporting tool are corporate management and development engineers. The proposed tool is composed of a series of interfaces in which customers may input parameters related to COTS components and PLE, corporation reuse structure, and other related information. Based on the information collected from the customers, estimations and calculations are carried out. In the end, the corporate management will get final reports produced by this supporting tool for each engineering cycle. Reuse Expert is the name of the tool.

8.2 Development Environment

8.2.1 Hardware

- CPU: Intel Pentium Mobile CPU 1.7GHZ
- Memory: 1024MB of Hynix DDR SDRAM
- Hard drive: 60G Hitachi 9.5 MM notebook hard drive

8.2.2 Software

- Operating System: Windows 2003 Server Standard Edition
- Development tool: Visual Studio .Net 2003 Professional
- Development database: Microsoft Access 2003

8.3 Basic processing structure of Reuse Expert

The basic structure of the tool is as followed:

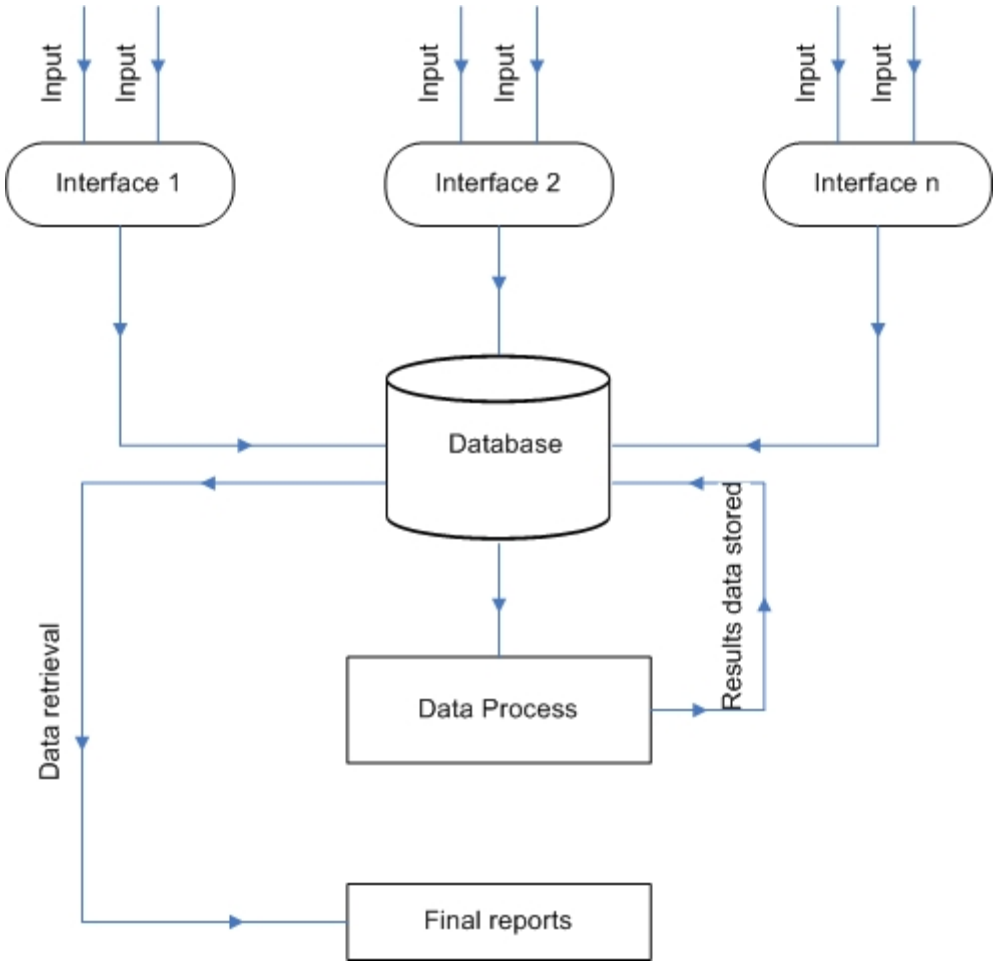

Figure 12: processing structure of Reuse Expert

The supporting tool has user-friendly interface and allows managers and development engineers to input values of related parameters. In the end, a detailed report is produced, by which the management can make final decision if it's worthy to purchase COTS components instead of developing from scratch and related results of PLE. The report is composed of details about costs and benefits, benefits from shortened TTM, final project NPV (Net Present Value) and ROI (Return On Investment).

8.4 Reuse Expert screens

8.4.1 Starting Page

The first screen is composed of three parts: input, output and viewer.

Input part includes:

- Corporate Cycle: input parameters of corporate cycle
- Domain Cycle: input parameters of domain cycle
- Component Cycle: input parameters of component cycle
- Application Cycle: input parameters of application cycle
- Default Arguments: input parameters of default arguments

Output part includes:

- Corporate Cycle: report of corporate cycle
- Domain Cycle: report of domain cycle
- Application Cycle: report of application cycle
- Component Cycle: report of component cycle

Viewer part includes:

- Viewer: show overall reuse information

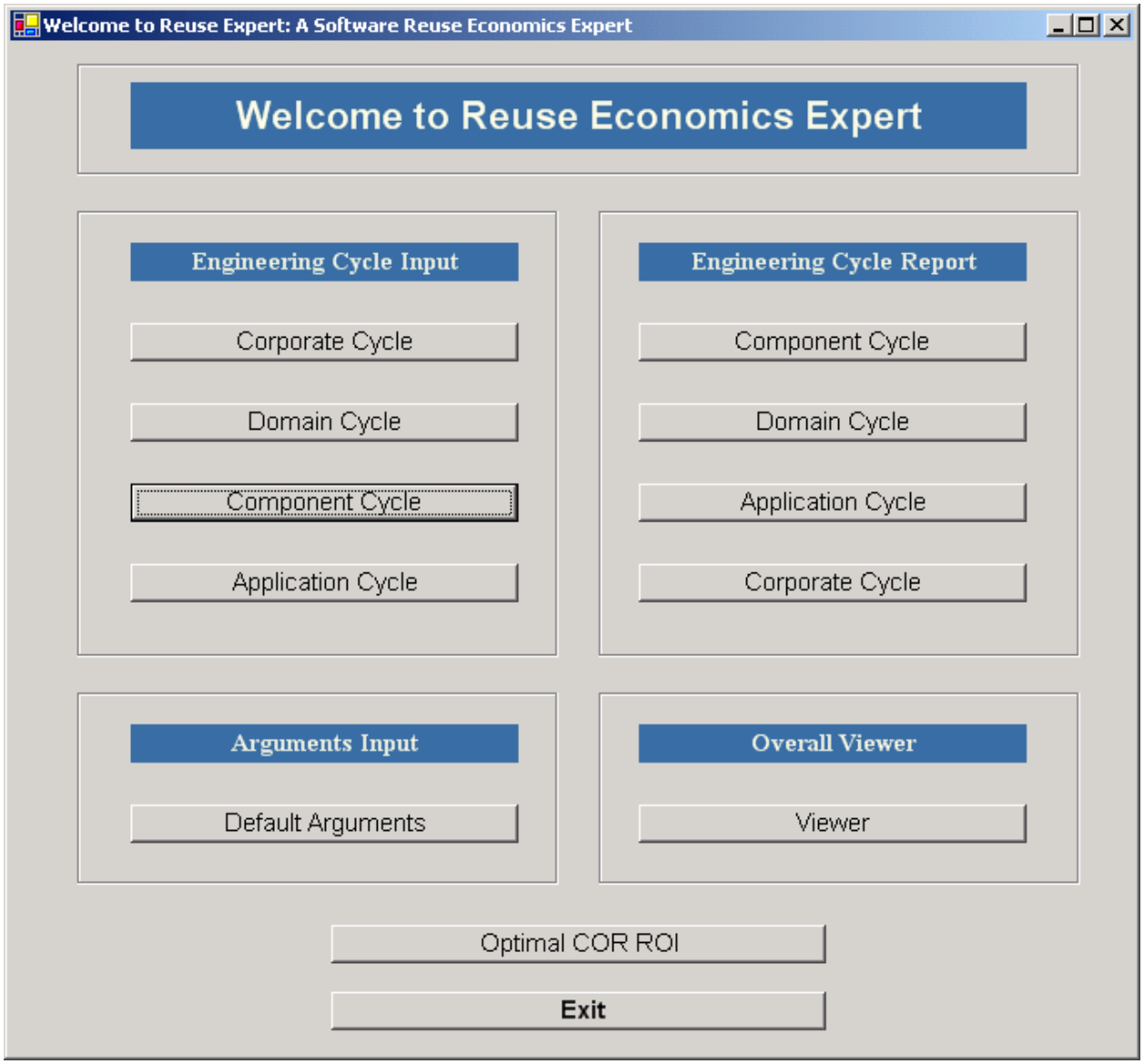

Figure 13: Starting Page

8.4.2 Corporation Cycle Input

This form includes the following input parameters.

- Corporation: corporation name
- Activity Date: initial year of corporate cycle
- Investment Duration: number of years of corporate cycle

Costs input parameters include:

- Infrastructure (in dollar)
- Training (in dollar)
- Operational Impact (in dollar)
- Management Restructuring (in dollar)
- Others (in dollar)

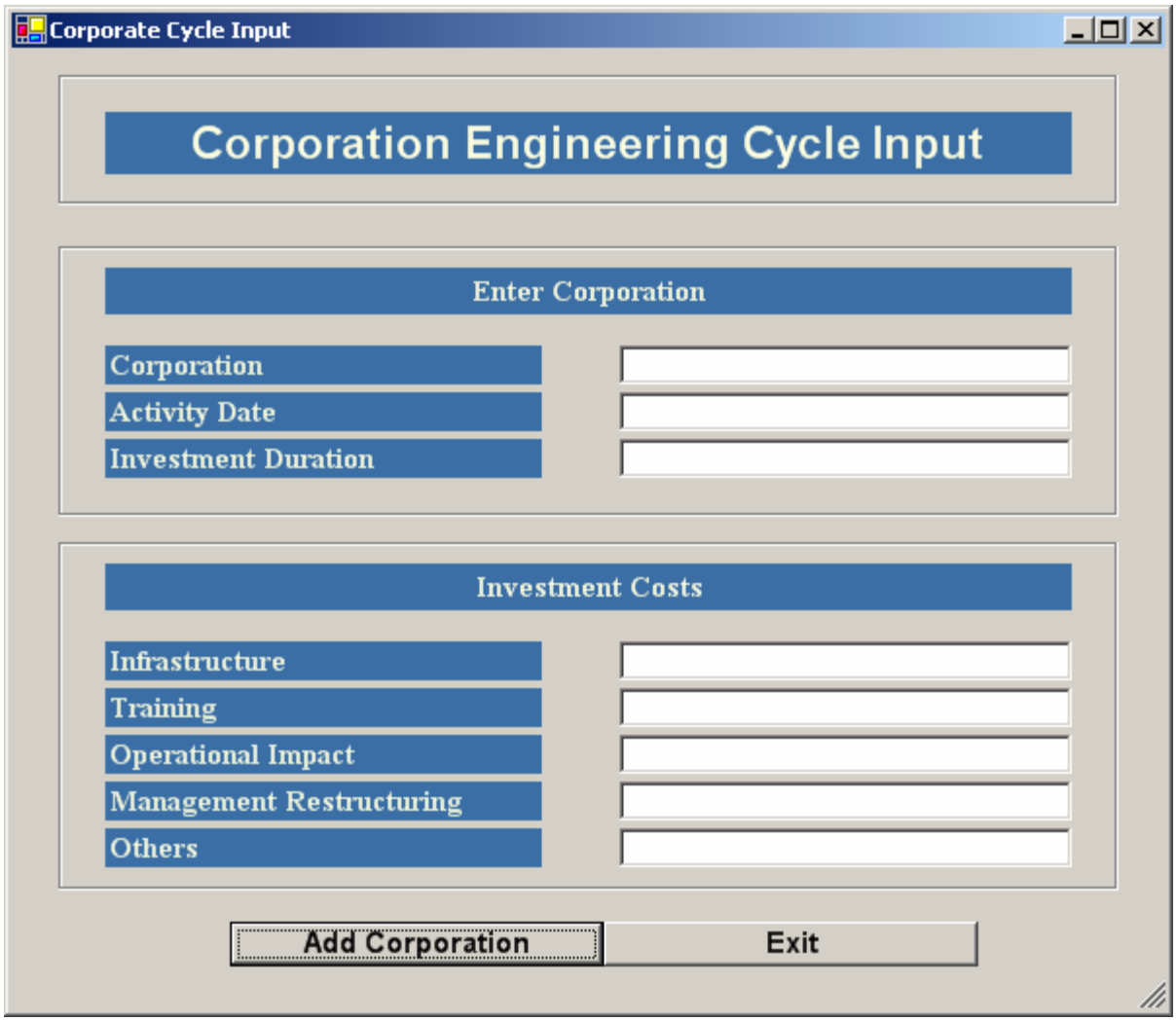

Figure 14: CEC Input

8.4.3 Domain Cycle Input

This form includes the following input parameters.

- Corporation: corporation name
- Domain: domain name
- Activity Date: initial year of domain cycle
- Investment Duration: number of years of domain cycle

Costs input parameters include:

- Domain Analysis Cost (in PM)

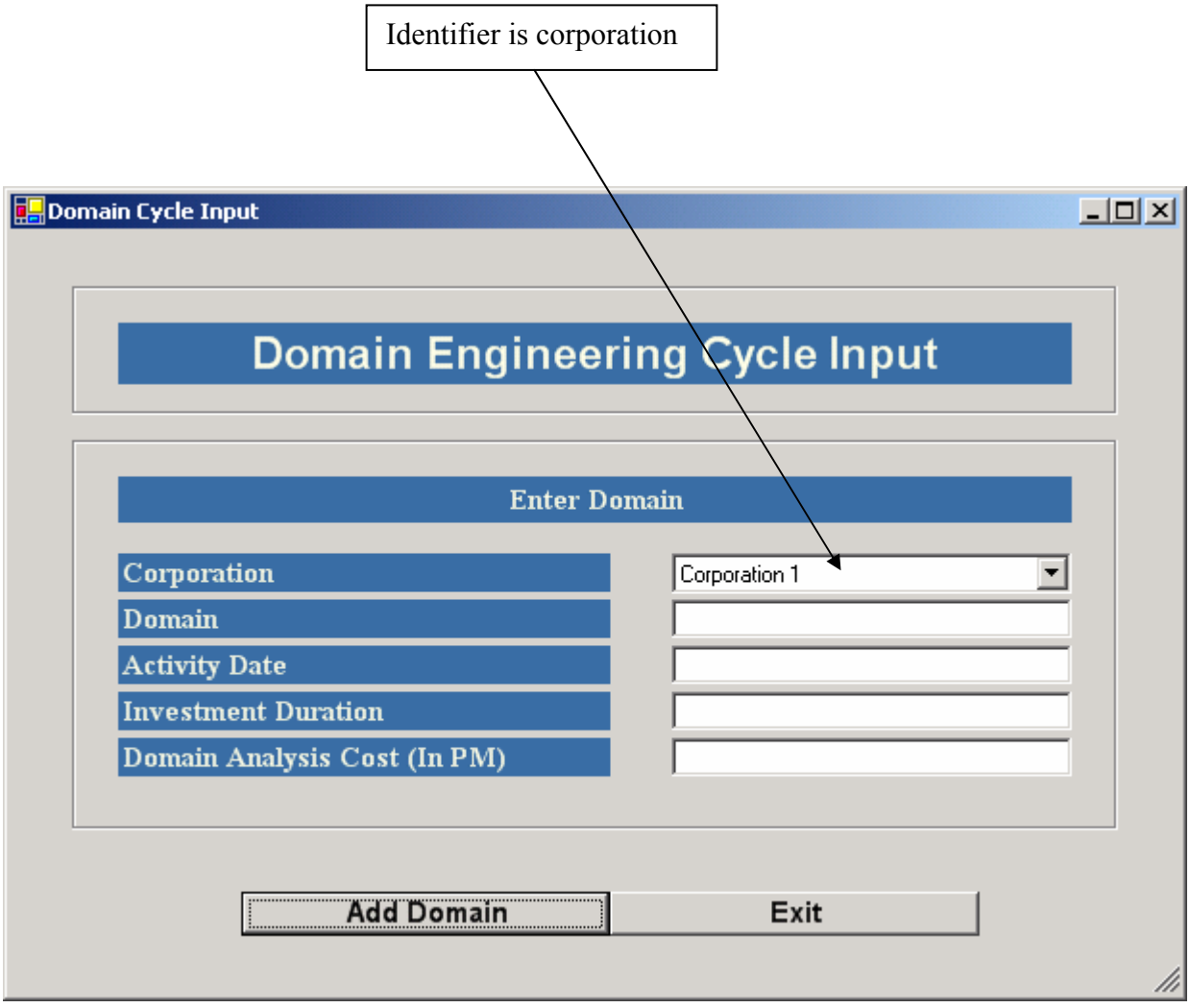

Figure 15: DEC Input

8.4.4 Component Cycle Input

This form includes the following input parameters.

- Corporation: corporation name
- Domain: domain name
- Component: component name
- Activity Date: initial year of component cycle
- Investment Duration: number of years of component cycle
- Component Type: develop from scratch, mine inside corporation, buy (COTS)
- Component Size: in k lines of source code
- Component Description: brief description of component

Three ways of core assets imported: develop from scratch, mine inside corporation, buy (COTS) and each has different set of input parameters.

Mine inside corporation

- Assessment and Identification cost (in PM)
- Library Insertion cost (in PM)

Develop from scratch

- Development Cost (in PM)
- Library Insertion cost (in PM)

Buy from market (COTS)

- Assessment and Identification cost (in PM)
- Purchase cost (in dollar)
- Annual License cost (in dollar)
- Library Insertion cost (in PM)

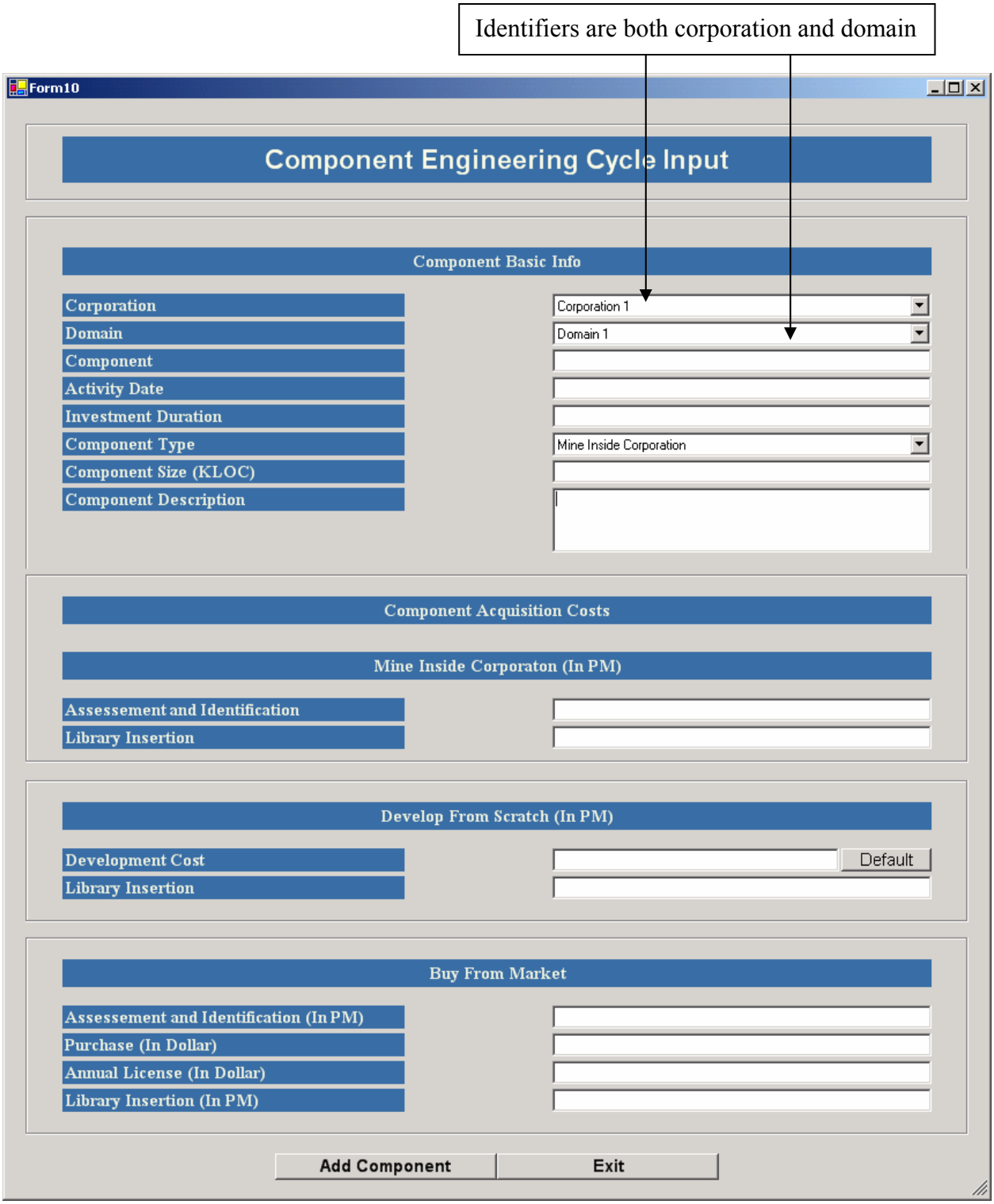

Figure 16: COEC Input

8.4.5 Application Cycle Input I

This form includes the following input parameters.

- Corporation: corporation name
- Project: application name
- Project Characteristics: semi-detached, organic, and embedded.
- Activity Date: initial year of application cycle
- Investment Duration: number of years of application cycle

This form can either add a new project or click Add Component button to add components into a selected project.

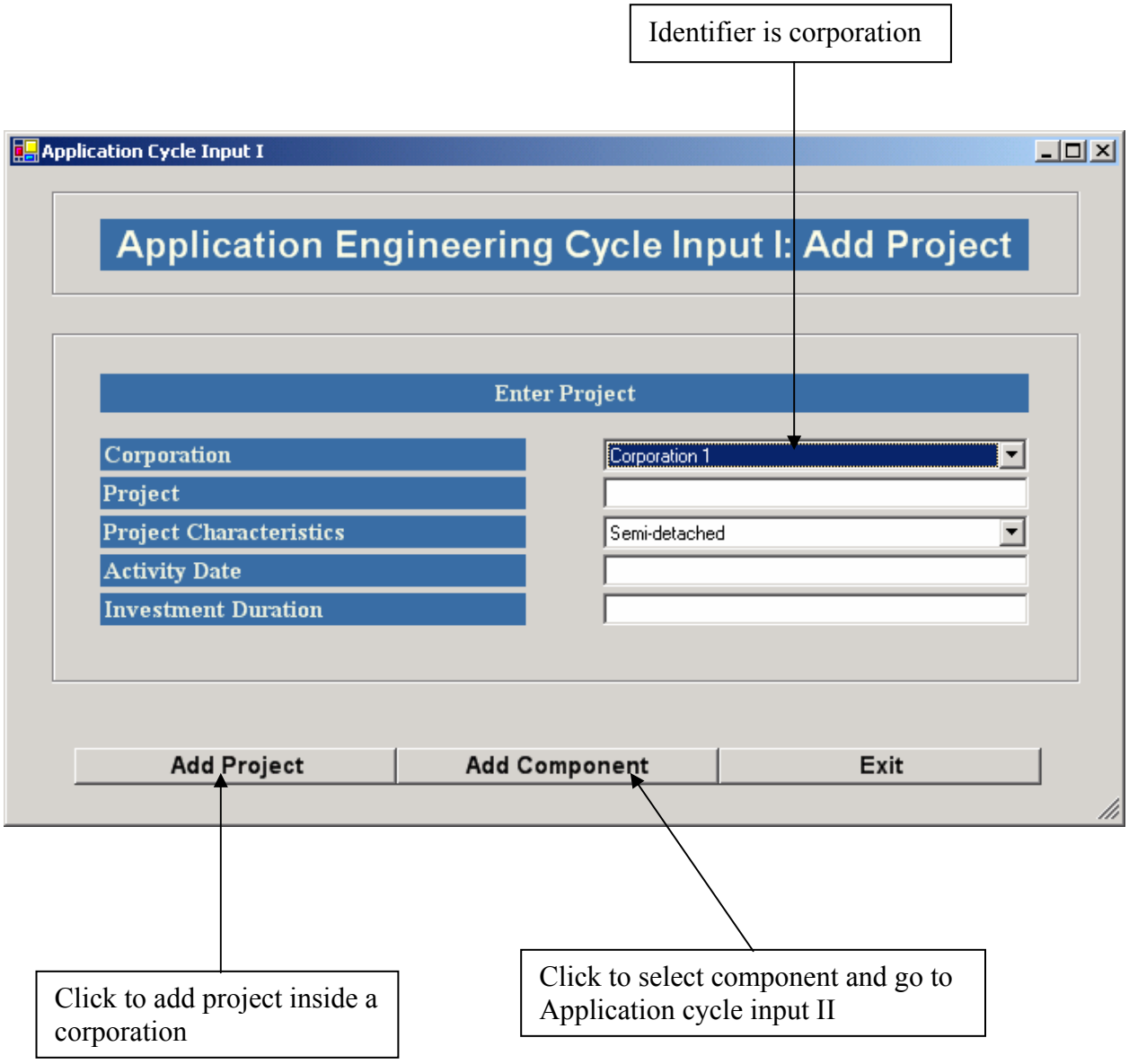

Figure 17: AEC Input I

8.4.6 Application Cycle Input II

This form includes the following input parameters.

- Corporation: corporation name
- Application: application name
- Domain: domain name
- Component: component name
- Reuse Type: black box and white box
- Glue Code Size: glue code to integrate the component

Each project can select components from different domains, black box use, white box use, etc. The identifier of project is corporation. The identifiers of component are both corporation and domain.

Reuse of different kind of components carries different input parameters.

Mine inside corporation or develop from scratch

- Percent of Design Modification (DM). The percentage of the adapted software's design that received modification to fulfill the objectives and environment of the new product.
- Percent of Code Modification (CM). The percentage of the adapted software's code that receives modification to fulfill the objectives and environment of the new product.
- Percent of Integration Required for Modified Software (IM). The percentage of effort needed for integrating and testing of the adapted software in order to combine it into the new product.

Buy from market

- ACIEP: COTS Integrator Experience with Product
- ACIPC: COTS Integrator Personnel Capability
- AXCIP: Integrator Experience with COTS Integration Processes
- APCON: Integrator Personnel Continuity
- ACPMT: COTS Product Maturity
- ACSEW: COTS Supplier Product Extension Willingness
- APCPX: COTS Product Interface Complexity
- ACPPS: COTS Supplier Product Support
- ACPTD: COTS Supplier Provided Training and Documentation
- APVOL: COTS Product Volatility
- ACREL: Constraints on System/Subsystem Reliability
- AACPX: Application Interface Complexity
- ACPER: Constraints on System/subsystem Technical Performance
- ASPRT: System Portability

- AAREN: Application Architectural Engineering
- Brak: Percentage of COTS glue code thrown away due to requirements volatility

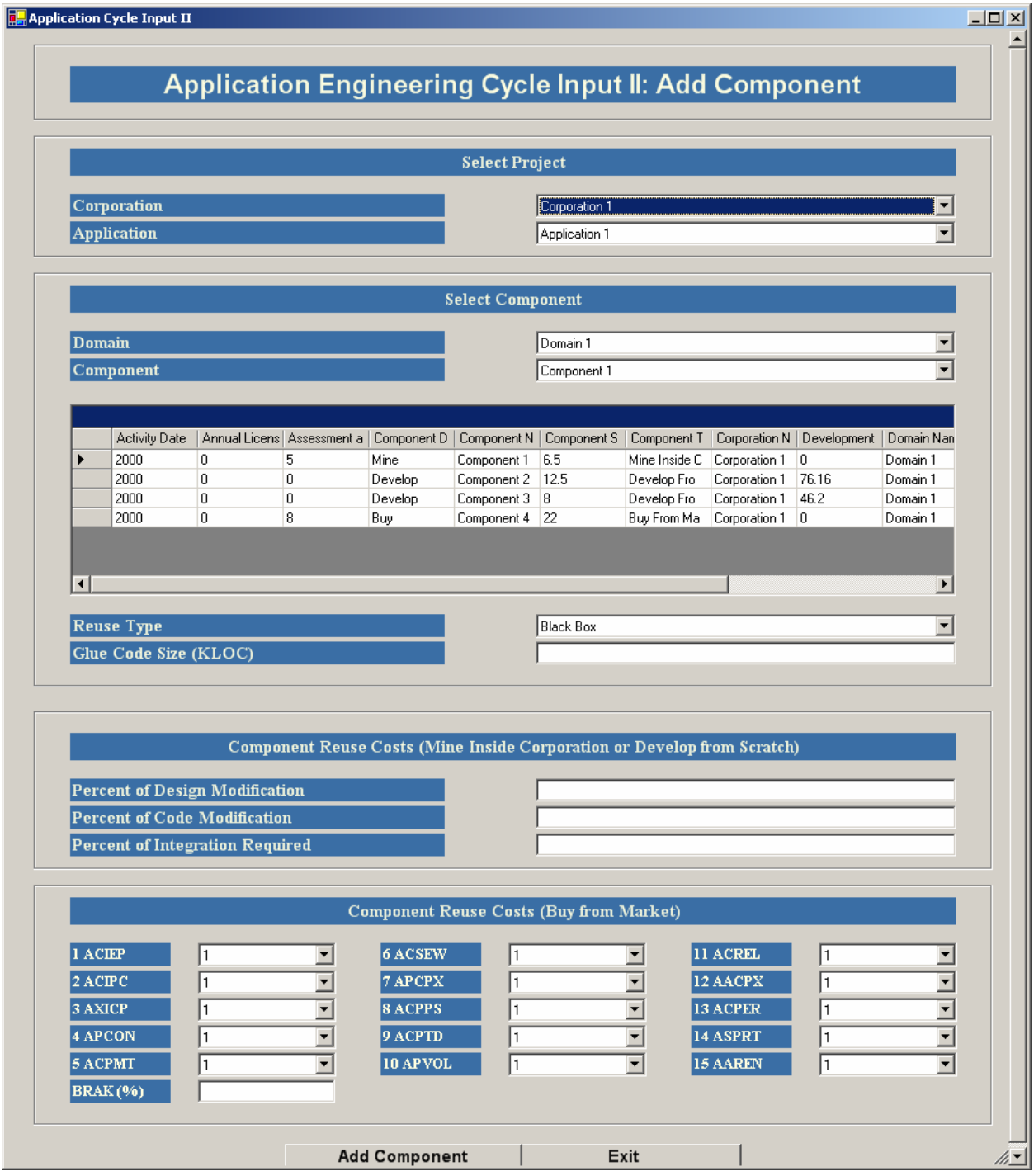

Figure 18: AEC Input II

8.4.7 Default Arguments Input

Each corporation has a set of default arguments. This form includes the following input parameters.

- Corporation: corporation name
- RCWR: Relative Cost of Writing for Reuse
- ACT: Annual Change Traffic
- ACT': Annual Change Traffic for reusable assets
- RBP: Relative Black box Price
- RWP: Relative White box Price
- Discount Rate: annual discount rate
- Benefit Coefficient: benefit coefficient for a product
- FTSP: annual cost of a Full-time Software Person (in dollar)
- Number of Librarians
- Number of Shares: number of shares of a corporation

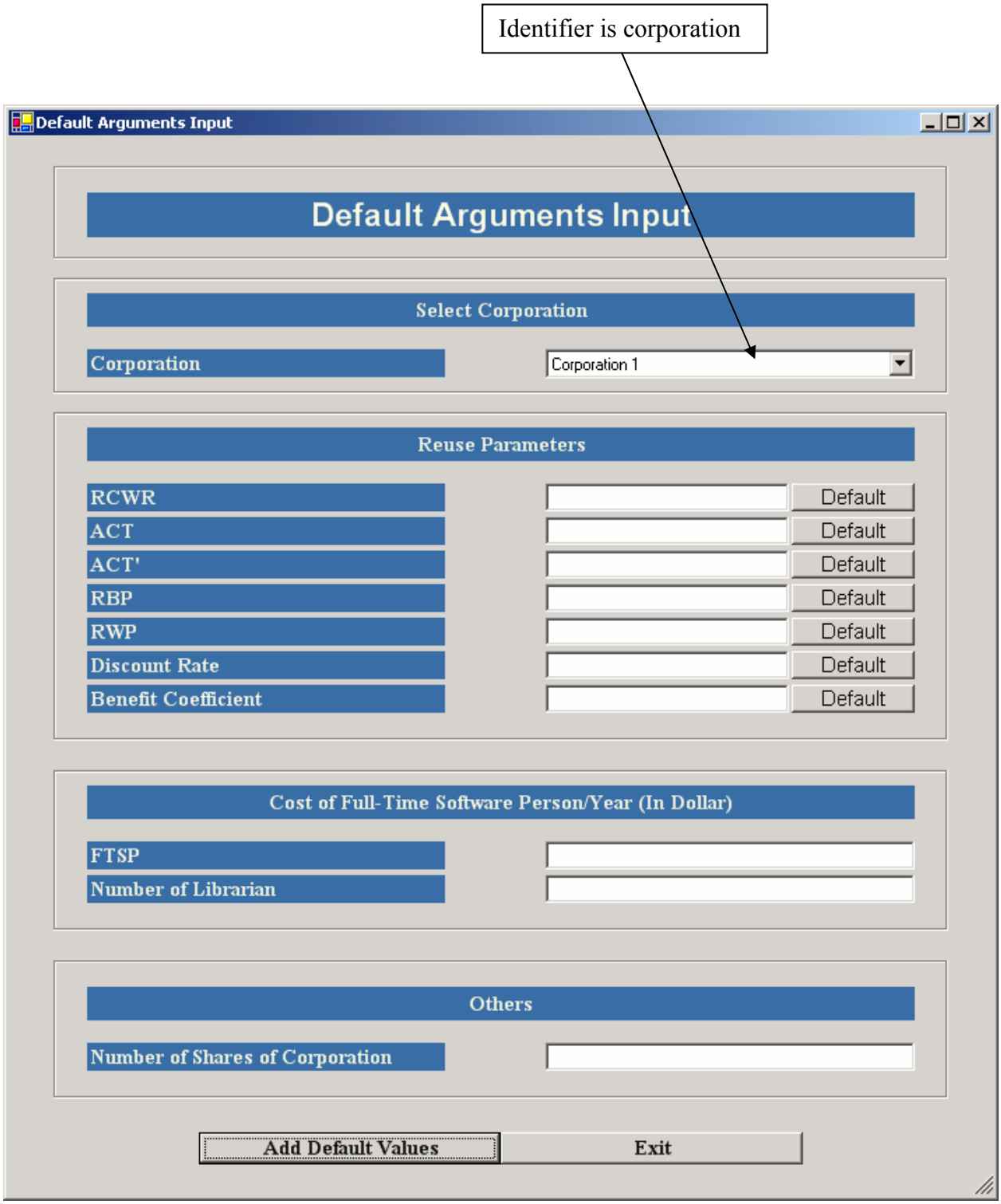

Figure 19: Default Arguments Input

8.4.8 Overall Reuse Info Viewer

As the figure shows, the model is based on a Top-down structure.
Each corporation has domains.
Each corporation has projects.
Each corporation has default arguments.
Each domain has core assets.
Each project utilizes core assets.

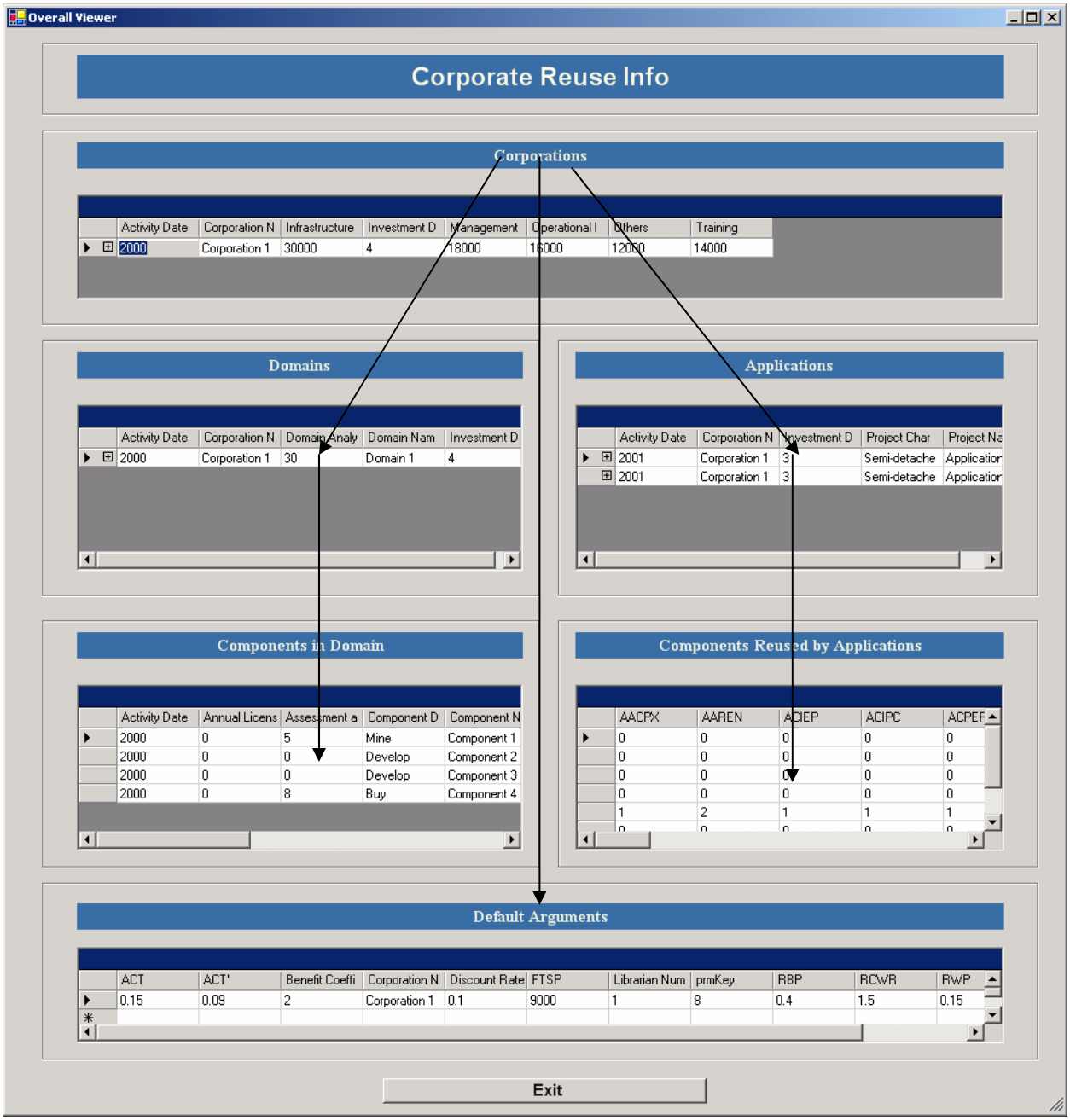

Figure 20: Overall Viewer

8.4.9 Component Cycle Report I

This form produces results of a component cycle, including costs and benefits, NPV, and ROI.

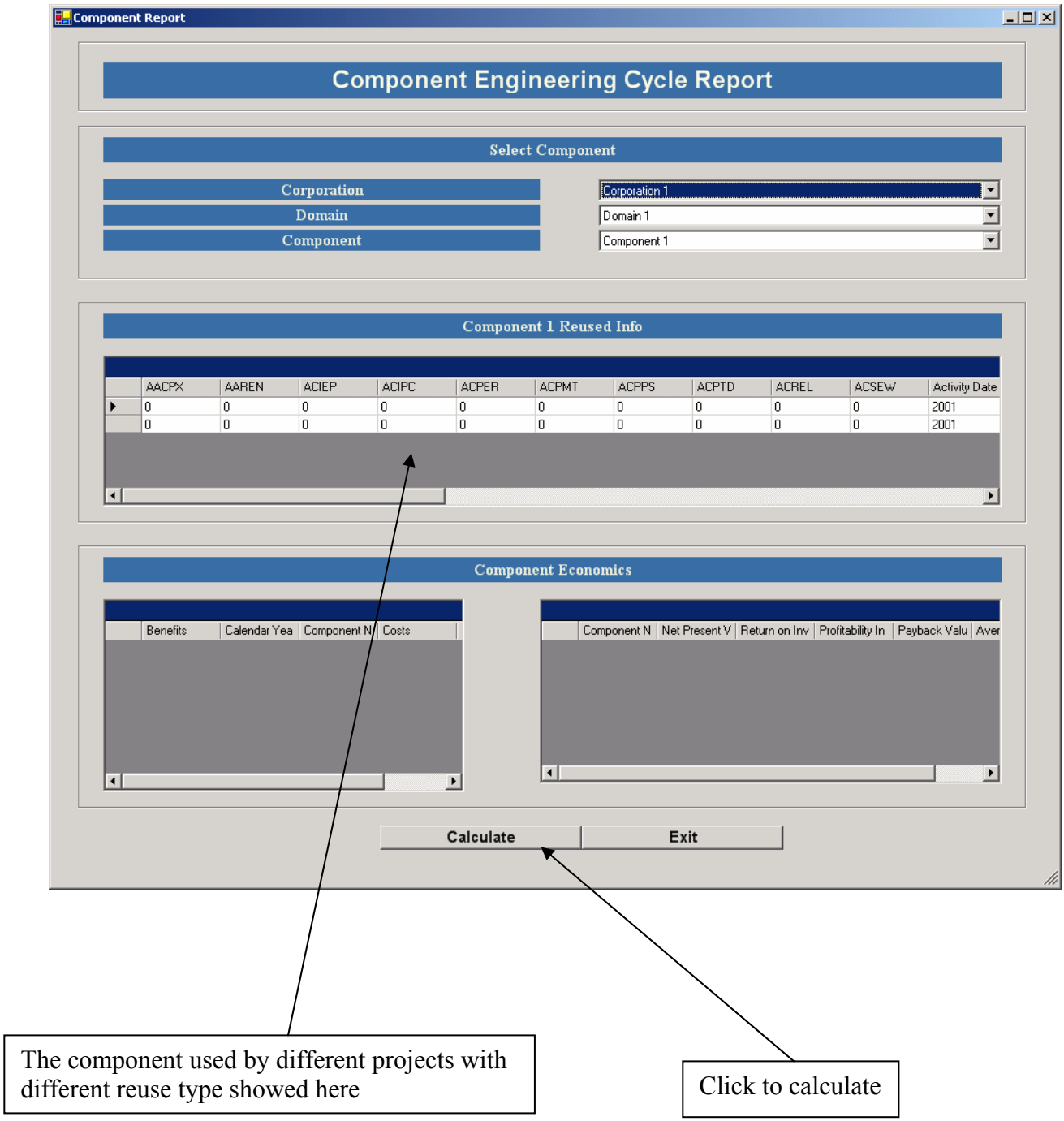

Figure 21: COEC Report

8.4.10 Component Cycle Report II

This form produces results of a component cycle, including costs and benefits, NPV, and ROI.

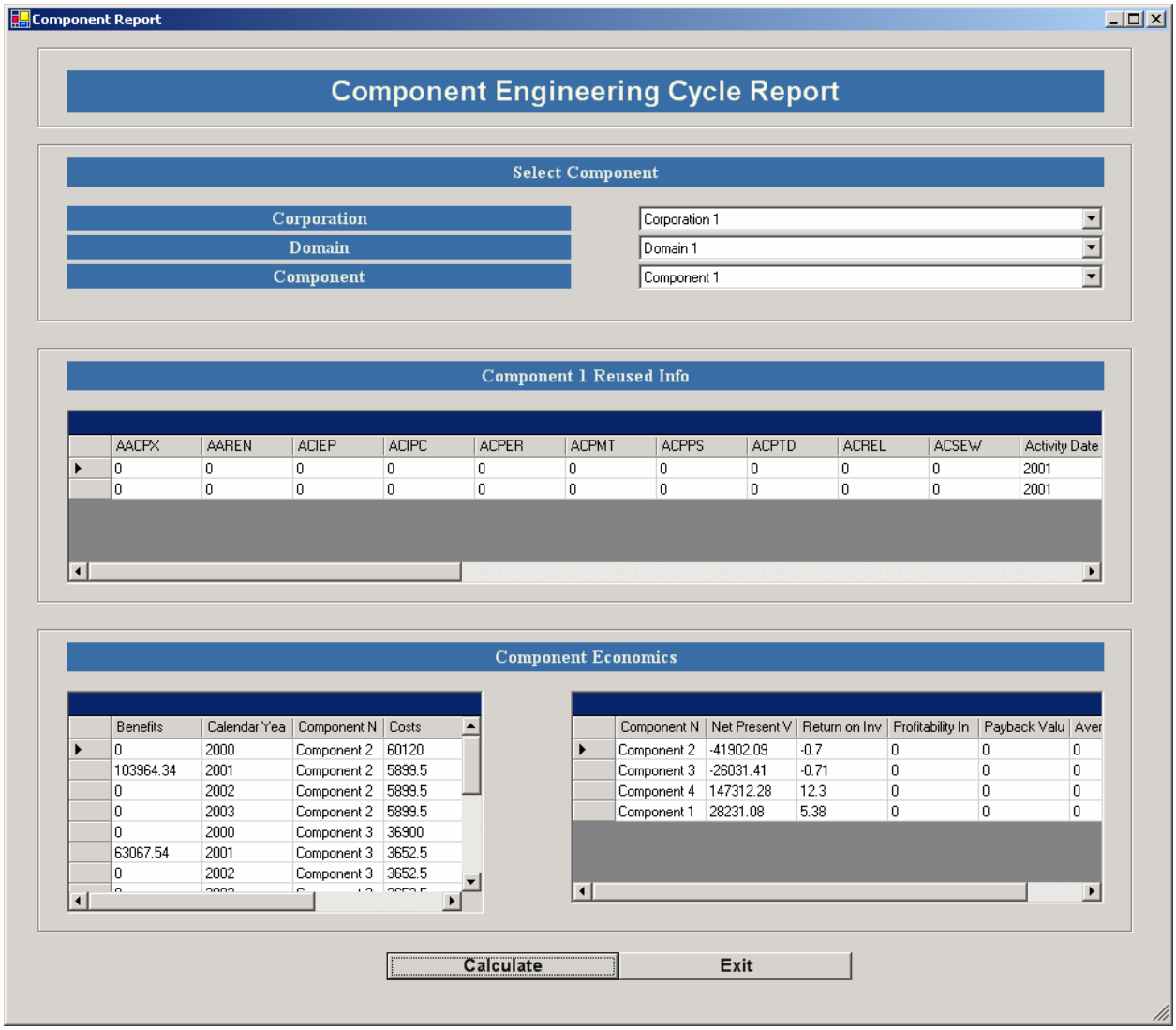

Figure 22: COEC Report

8.4.11 Domain Cycle Report

This form produces results of a domain cycle, including costs and benefits, NPV, and ROI.

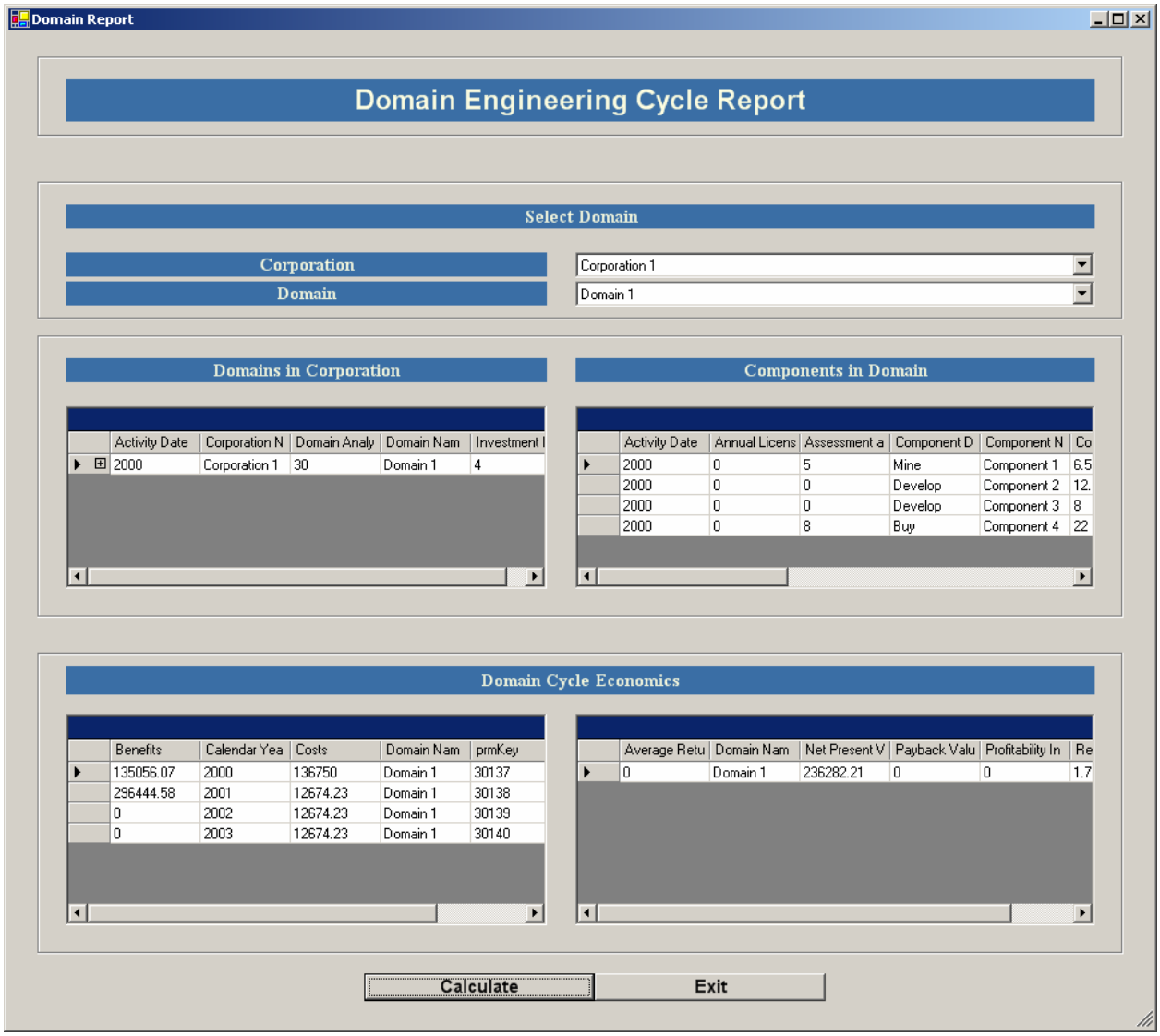

Figure 23: DEC Report

8.4.12 Application Cycle Report

This form produces results of an application cycle, including costs and benefits, NPV, and ROI.

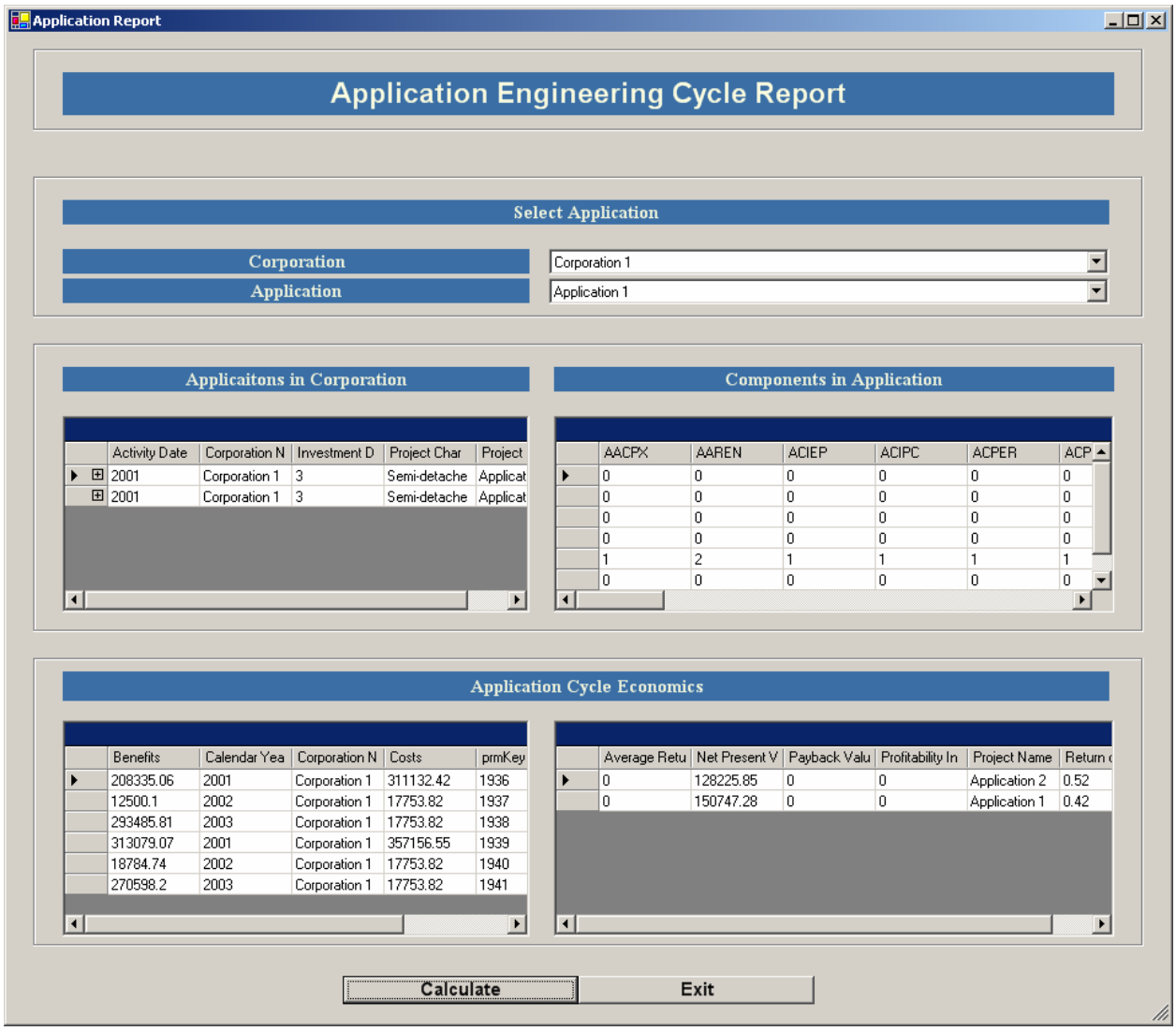

Figure 24: AEC Report

8.4.13 Corporate Cycle Report

This form produces results of a corporation cycle, including costs and benefits, NPV, and ROI.

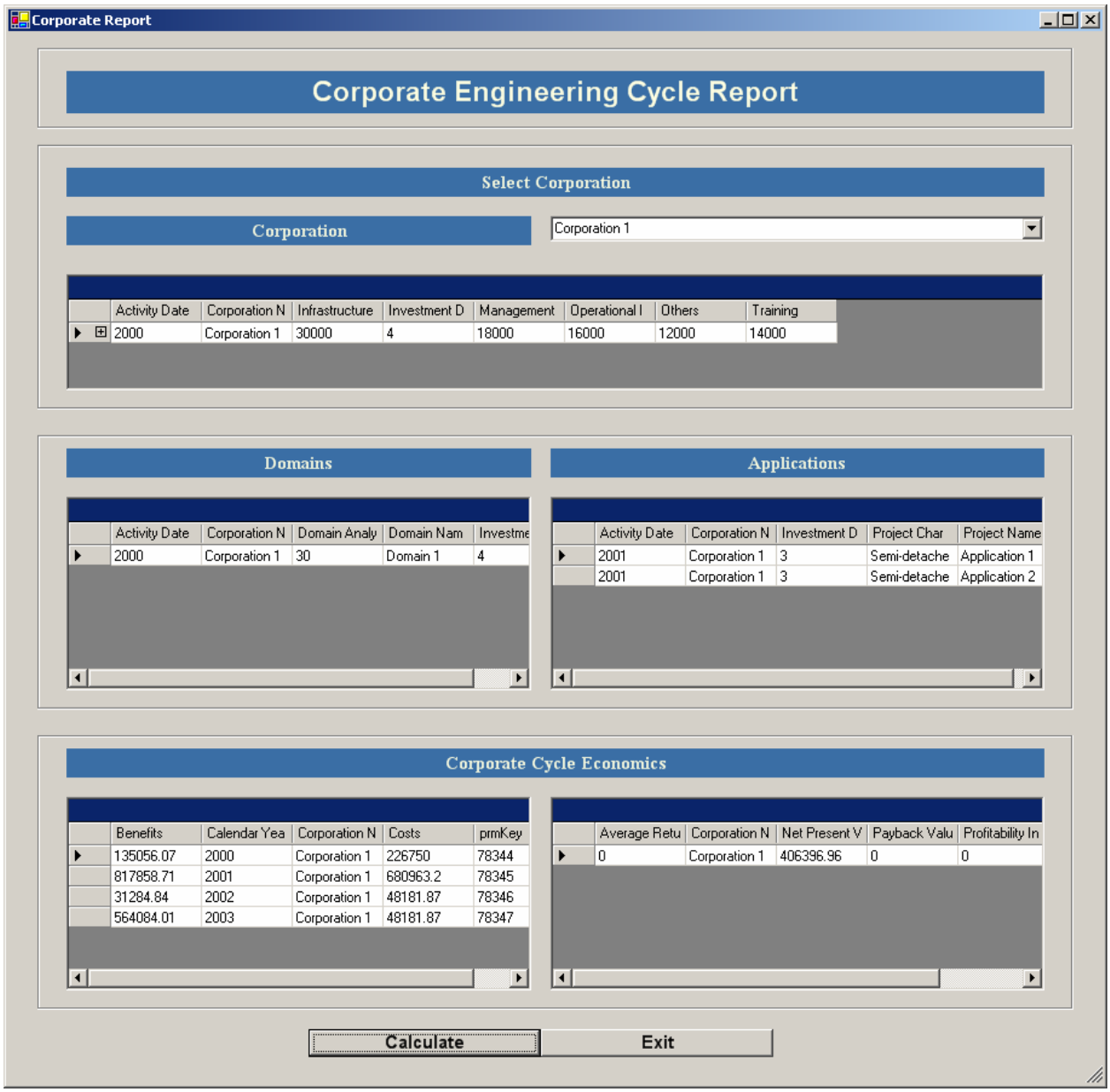

Figure 25: CEC Report

8.4.14 Optimal Corporate ROI Report

This form produces optimal corporate ROI and ROI's of other cycles. The pair of RBP and RWP values is also calculated.

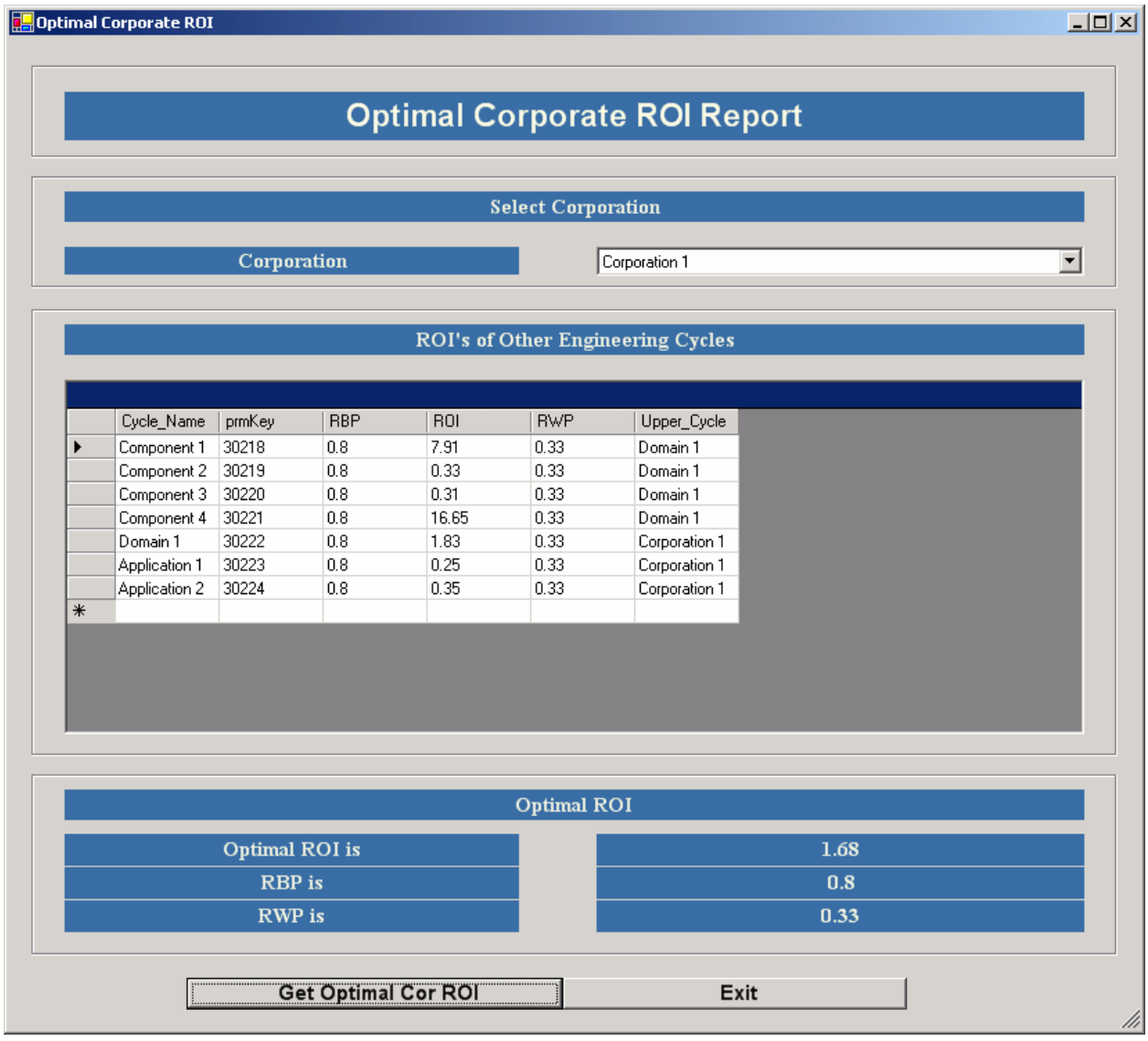

Figure 26: Optimal ROI

Chapter 9 Conclusion

9.1 Motivation

Chmiel'model doesn't cover Commercial-Off-The-Shelf (COTS), Product Line Engineering (PLE) and benefits due to shortened Time-To-Market (TTM) and can only deal with internal traffic since the model assumes all of the reusable components are built from scratch in house. For example, Extra efforts caused by the use of COTS, assessment, tailoring, glue code and COTS volatility, are not covered in this model. And this model doesn't treat the benefits of TTM, for example, business performance.

By extending Chmiel's model, the new model is applied to CBSE (Component-Based Software Engineering), COTS reuse systems and PLE. In addition, attempts of quantifying benefits of shortened TTM are made within this study and a TTM submodel is developed to cover this issue.

9.2 Summary

The original work and contribution of the study is that by extending Chmiel's model, the new model is applied to CBSE (Component-Based Software Engineering), COTS reuse systems and PLE. In addition, attempts of quantifying benefits of shortened TTM are made within this study and a TTM submodel is developed to cover this issue.

Another main contribution of the study is that it also addresses the issue to analyze and optimize corporate Return On Investment (ROI). The rational is that optimizing (maximizing) the corporate ROI under the condition that all other ROI's are positive. By designing an algorithm and applying it to the data, this study discovers the method how to make the maximized value of corporate ROI.
The extended model can cover:

- CBSE
- COTS
- PLE
- TTM

This study also addresses the issue to analyze and optimize corporate Return On Investment (ROI). The rational is that optimizing (maximizing) the corporate ROI under the condition that all other ROI's are positive. By designing an algorithm and applying it to the data, this study discovers the method how to make the maximized value of corporate ROI.

9.3 Future Work

Future work could:

- Analyze further quantification on reuse library.
- Estimate the TTM results on corporation's initiative on reuse.
- Enhance the supporting tool to make it more adjustable.

References

[Abts 1997] Christopher M. Abts, Barry W. Boehm, COTS/NDI Software Integration Cost Estimation & USC-CSE COTS Integration Calculator V2.0 User Guide, University of Southern California, September 1997.

[Abts 1999] Chris Abts, Barry W. Boehm, Elizabeth Bailey Clark, COCOTS: A COTS Software Integration Lifecycle Cost Model-Model Overview and Preliminary Data Collection Findings, University of Southern California, 1999**.**

[Agrizzi 2003] Dila Agrizzi, Advanced Management Accounting, Capital Investment Decisions I, 2003.

[Barnes 1991] Barnes, B. and T. Bollinger. Making Reuse Cost Effective. IEEE Software, Jan. 1991, 8(l): 13-24.

[Boehm 1981] Barry W. Boehm, Software Engineering Economics, Prentice Hall, NJ, 1981.

[Boehm 1998] Barry W. Boehm, COCOTS, Software Integration Cost Model: An Overview, University of South California, July 1998: 8-12.

[Brigham 1999] Eugene F. Brigham, Louis Gapenski, Philip Daves, Intermediate Financial Management, the Dryden Press, 1999.

[Carney 1997] Carney, D. Assembling Large Systems from COTS Components: Opportunities, Cautions, and Complexities. SEI Monographs on Use of Commercial Software in Government Systems, 1997.

[Careertools 2000] Determing Return On Investment in Training/Education, Careertools.org, 2000.

[Chmiel 2000] Senta Fowler Chmiel, An Integrated Cost Model for Software Reuse, West Virginia University, 2000.

[Clark 2003] Besty Clark, Managing COTS Integration for High Integrity Systems: Observations from the COCOTS Database, Software Metrics Inc., March 2003: 8-10.

[COCOMO 1994] USC COCOMO reference Manual, University of Southern California, 1994.

[Damordoran 2002] Damordoran, A Lecture of Economic Value Added (EVA), New York University, 2002.

[DOD 1997] Department of Defense. Software Reuse Executive Primer, Falls Church, VA, 1997.

[Erdogmus 2000] Hakan Erdogmus and Chris Abts, Breakout Session on Economic and Financial Issues, ICSE '2000 COTS Workshop, 2000.

[Griss 1993] Martin L. Griss, Software Reuse: From Library to Factory, Hewlett Packard, Software Technology Laboratory, HPL-93-67, July 1993.

[Harris 1992] K. Harris, Using an Economic Model to Tune Reuse Strategies, proceedings of the fifth annual software reuse workshop, 1992.

[Higaki 1995] Wesley H. Higaki, Applying an improved Economic Model to Software Build-versus-Buy Decisions, Hewlett-Packard Journal, August 1995.

[Honeywell 1996] Life Cycle Effort Breakdown, Honeywell Labs, 1997.

[Hudson 2001] Michael J. Hudson, What effect will code reusability have on strategic business applications, Blueprint Technologies Inc, Oct. 2001.

[Madachy 2004] Ray, Madachy, COTS Integration and COCOTS, Center for Software Engineering, USC, Feb. 2004.

[Makelainen 1998] Esa Makelainen, Introduction to Economic Value Added, EVA, Stern Stewart & Co., Oct. 1998.

[Malan 1993] Malan, R. Software Reuse: A Business Perspective. Technical Report, Hewlett Packard Laboratories, February 1993.

[Market Forecasters] Current and Emerging Embedded Markets-Electronic Market Forecasters, Redcellx.com, 1999.

[McKinsey] Technical report, McKinsey & Co.

[Meyers 2001] Meyers & Oberndorf, Managing Software Acquisition – Open Systems and COTS Products, Addison-Wesley, 2001.

[Nakano 2000] Real World ROI's, Russell Nakano, Principal Consultant, Interwoven Inc, 2000.

[Neilsen 2000] Mitchell L. Neilsen, Design with Reuse, 2000.

[Patterson 1993] M. Patterson, Accelerating Innovation, Van Nostrand Reinhold, 1993.

[Poulin 2002] Jeffrey S. Poulin, An Agenda for Software, Reuse Economics, International Conference on Software Reuse, 15 April 2002.

[Ray 2004] Ray, Madachy, COTS Integration and COCOTS, Center for Software Engineering, USC, 2004.

[Ross 1996] Ross, S.A. et al., Fundamentals of Corporate Finance, Irwin, 1997.

[SEI 2003] Software Product Line Acquisition: A Companion to A Framework for Software Product Line Practice, Software Engineering Institute, 2003.

[SEI 2004] Software reuse, http://www.sei.cmu.edu/cmm/cmm-v2/reuse-kpa.html, 2004.

[Smith and Reinertsen 1991] P.G. Smith, and D.G. Reinertsen, Developing Products in half the Time, Van Norstrand Reinhold, 1991.

[Vetzal 2003] Ken Vetzal, Introduction to Stock Valuation, University of Waterloo, 2003.

[Vigder 1998] Mark Vigder, An Architecture for COTS Based Software Systems, National Research Council Canada, Nov. 1998.

[Wallnau 1998] Kurt Wallnau, Commercial-Off-The-Shelf (COTS) Software: Five Key Implications for the System Architect, Software Engineering Institute,1998.

[Whitemarsh 1997] Buy or Generate Whitemarsh Information Systems Corporation, Section of Make, 1997.

Appendix A LinkySky Data

Appendix A.1 Data Collection Questionnaire

I am a Ph.D candidate student at Lane Department of Computer Science and Electrical Engineering Department. My research topic is extension of software reuse economics model. In this research, a fundamental requirement for such research is real-world software development project data. This data will be used to test hypotheses and simulate the model. The contribution of your data will ensure the final model is developed and simulated based on reasonable data.

The data that is contributed is important to the research. I will safeguard your contribution so as not to compromise company proprietary information. And the information is just for this academic project and won't be used as commercial purpose.

This questionnaire attempts to address four different levels of data granularity:

- Corporate level
- Domain level
- Application level
- Component level

Contact Information

Lin Yang (Research Assistant)
Room 403
Concurrent Engineering Research Center
West Virginia University
886 Chestnut Ridge Road
Morgantown, WV 26506-6506
USA
Voice: 001-(304)293-7226, Ext 4410
Fax: 001-(304)293-7541
Email: yang@csee.wvu.edu

Appendix A.2 LinkySky Data

LinkySky Science and Technology, INC. is a high-tech company set up by returned overseas students. The company is located in Beijing, China. The company engages in network systems integration, systems design and implementation, as well as E-business.

In 2000, a domain is developed for reusable assets. From 2001 to 2003, the company developed two projects, which are based reusable components and share the same architecture. The projects characteristics are semi-detached. The average salary of software engineer of the company is equivalent to 9K/year in US dollar. There is one software engineer responsible for reuse library. The discount rate is 10%. Since the company is not a public one, the data of number of shares of the company is not available.

Appendix A.2.1 Component Engineering Cycle

There are four reusable components in reuse library and they are incorporated in year 2000. One of them is mine legacy software, component 1. Two are develop from scratch, component 2 and 3. The left one is buy from market, component 4.

Component	Type	Size(in K)	As.&ID. (in PM)	Li. Ins. (in PM)	Development (in PM)	Pur. (in $)	Ann. License (in $)
Component 1	Mine	6.5	5.0	2.0	N/A	N/A	N/A
Component 2	Develop	12.5	N/A	4.0	50.78	N/A	N/A
Component 3	Develop	8.0	N/A	3.0	30.80	N/A	N/A
Component 4	Buy	22.0	8.0	5.0	N/A	2230.0	0.0

As. ID.: Assessment and Identification
Li. Ins.: Library Insertion
Pur.: Purchase
Ann. License: Annual License

Appendix A.2.2 Domain Engineering Cycle

Domain 1 is staring in year 2000. The domain analysis cost is 30PM.

Appendix A.2.3 Application Engineering Cycle

There are two applications.

Application 1

Application 1 is starting in year 2001. The table below shows the components used in Application 1.

Application 1							
2000		2001		2002		2003	
Black	White	Black	White	Black	White	Black	White
		Com 1	Com 2				
		Com 2					
		Com 3					
		Com 4					

The table below shows the glue code for each component.

Application 1			
	Type	Size(in K)	Glue Code
Component 1	Mine	6.5	4
Component 2	Develop	12.5	5
Component 3	Develop	8	3
Component 4	Buy	22	8

For the white box use of component 2,

DM	CM	IM
0	15	5

Component 4 is COTS and glue code is 8K. The costs in 2001 include acquire reusable asset, tailoring and glue code. The following steps show the statistics to calculate. The values of effort multipliers are

1	2	3	4	5	6	7	8	9	10	11	12	13	14
ACIEP	ACIPC	AXICP	APCON	ACPMT	ACSEW	APCPX	ACPPS	ACPTD	APVOL	ACREL	AACPX	ACPER	ASPRT
1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Application 2

Application 2 is starting in year 2001The table below shows the components used in Application 2.

Application 2							
2000		2001		2002		2003	
Black	White	Black	White	Black	White	Black	White
		Com 1	Com 3				
		Com 2					
		Com 3					
		Com 4					

The table below shows the glue code for each component.

Application 2			
	Type	Size(in K)	Glue Code
Component 1	Mine	6.5	3
Component 2	Develop	12.5	4
Component 3	Develop	8	3
Component 4	Buy	22	8

For the white box use of component 3

DM	CM	IM
0	20	10

Component 4 is COTS and glue code is 8K. The costs in 2001 include acquire reusable asset, tailoring and glue code. The following steps show the statistics to calculate. The values of effort multipliers are

1	2	3	4	5	6	7	8	9	10	11	12	13	14
ACIEP	ACIPC	AXICP	APCON	ACPMT	ACSEW	APCPX	ACPPS	ACPTD	APVOL	ACREL	AACPX	ACPER	ASPRT
1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Appendix A.2.4 Corporate Engineering Cycle

- Infrastructure cost: $20,000
- Training cost: $10,000
- Operational impact: $5,000
- Management restructuring: $3,000
- Others: $2,000

Appendix B Reuse Expert Database Design and Schema

Appendix B.1. Database Relationships

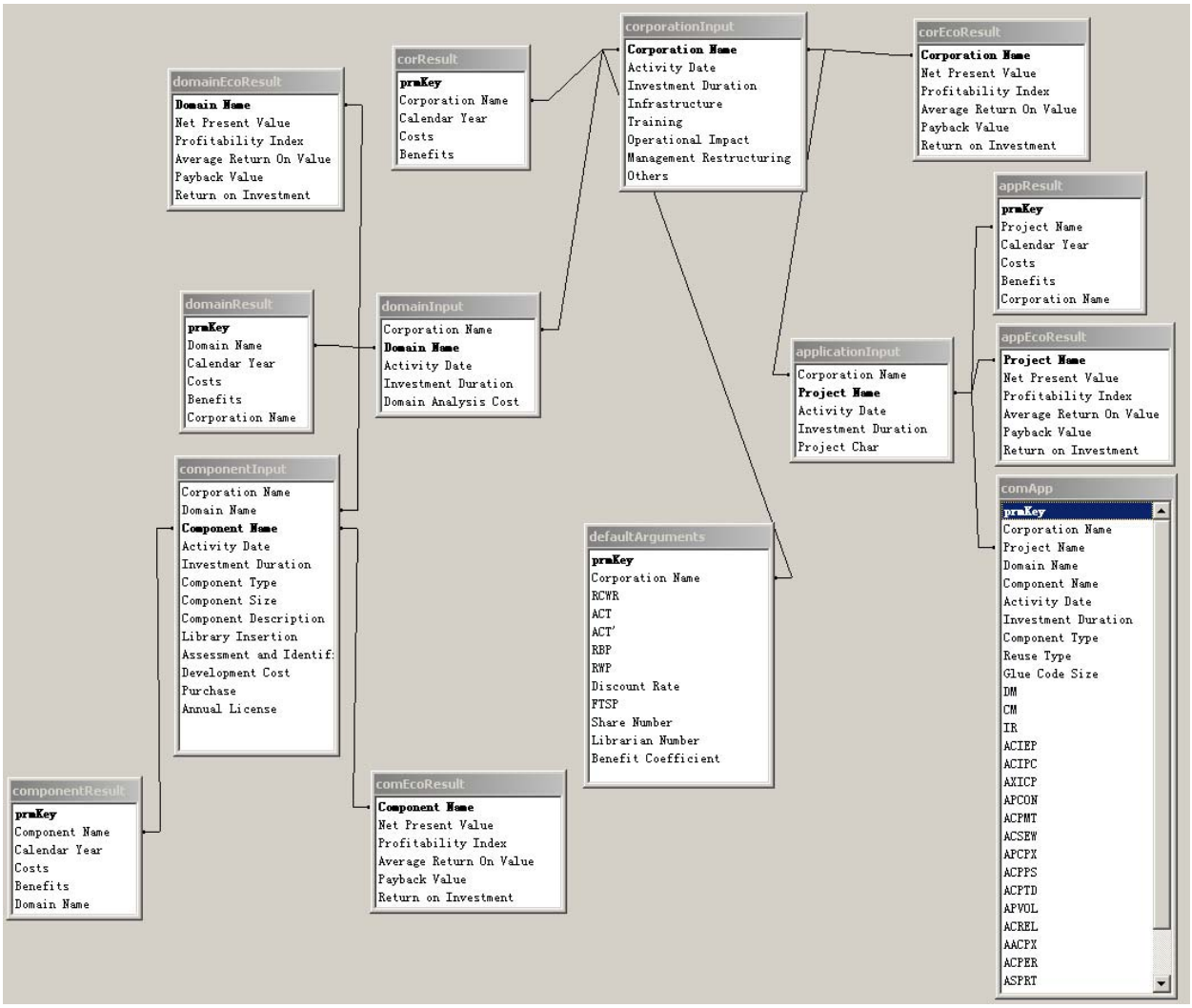

Figure 27: Database Relationships

Appendix B.2. Tables (alphabetical order)

Table Name	Field Name	Data Type
appEcoResult	Average Return On Value	Double
	Net Present Value	Double
	Payback Value	Double
	Profitability Index	Double
	Project Name	Text
	Return on Investment	Double

Table Name	Field Name	Data Type
applicationInput	Activity Date	Integer
	Corporation Name	Text
	Investment Duration	Integer
	Project Char	Text
	Project Name	Text

Table Name	Field Name	Data Type
appResult	Benefits	Double
	Calendar Year	Integer
	Corporation Name	Text
	Costs	Double
	prmKey	Auto Number
	Project Name	Text

Table Name	Field Name	Data Type
comApp	AACPX	Double
	AAREN	Double
	ACIEP	Double
	ACIPC	Double
	ACPER	Double
	ACPMT	Double
	ACPPS	Double
	ACPTD	Double
	ACREL	Double
	ACSEW	Double
	Activity Date	Integer
	APCON	Double
	APCPX	Double
	APVOL	Double
	ASPRT	Double

	AXICP	Double
	BRAK	Double
	CM	Double
	Component Name	Text
	Component Size	Double
	Component Type	Text
	Corporation Name	Text
	DM	Double
	Domain Name	Text
	Glue Code Size	Double
	Investment Duration	Integer
	IR	Double
	prmKey	Auto Number
	Project Name	Text
	Reuse Type	Text

Table Name	**Field Name**	**Data Type**
comEcoResult	Average Return On Value	Double
	Component Name	Text
	Net Present Value	Double
	Payback Value	Double
	Profitability Index	Double
	Return on Investment	Double

Table Name	**Field Name**	**Data Type**
componentInput	Activity Date	Integer
	Annual License	Double
	Assessment and Identification	Double
	Component Description	Text
	Component Name	Text
	Component Size	Double
	Component Type	Text
	Corporation Name	Text
	Development Cost	Double
	Domain Name	Text
	Investment Duration	Integer
	Library Insertion	Double
	Purchase	Double

Table Name	Field Name	Data Type
componentResult	Benefits	Double
	Calendar Year	Integer
	Component Name	Text
	Costs	Double
	Domain Name	Text
	prmKey	Auto Number

Table Name	Field Name	Data Type
corEcoResult	Average Return On Value	Double
	Corporation Name	Text
	Net Present Value	Double
	Payback Value	Double
	Profitability Index	Double
	Return on Investment	Double

Table Name	Field Name	Data Type
corporationInput	Activity Date	Integer
	Corporation Name	Text
	Infrastructure	Double
	Investment Duration	Integer
	Management Restructuring	Double
	Operational Impact	Double
	Others	Double
	Training	Double

Table Name	Field Name	Data Type
corResult	Benefits	Double
	Calendar Year	Integer
	Corporation Name	Text
	Costs	Double
	prmKey	Auto Number

Table Name	Field Name	Data Type
defaultArguments	ACT	Double
	ACT'	Double
	Benefit Coefficient	Double
	Corporation Name	Text
	Discount Rate	Double
	FTSP	Double
	Librarian Number	Integer
	prmKey	Auto Number
	RBP	Double
	RCWR	Double
	RWP	Double
	Share Number	Integer

Table Name	Field Name	Data Type
domainEcoResult	Average Return On Value	Double
	Domain Name	Text
	Net Present Value	Double
	Payback Value	Double
	Profitability Index	Double
	Return on Investment	Double

Table Name	Field Name	Data Type
domainInput	Activity Date	Integer
	Corporation Name	Text
	Domain Analysis Cost	Double
	Domain Name	Text
	Investment Duration	Integer

Table Name	Field Name	Data Type
domainResult	Benefits	Double
	Calendar Year	Integer
	Corporation Name	Text
	Costs	Double
	Domain Name	Text
	prmKey	Auto Number

Table Name	Field Name	Data Type
ROI	Cycle_Name	Text
	prmKey	Auto Number
	RBP	Double
	ROI	Double
	RWP	Double
	Upper_Cycle	Text

Appendix C List of Acronyms

AAF	Adaptation Adjustment Factor
AB	Annual Benefit
AC	Annual Cost
AC	Assessment Cost
ACT	Annual Change Traffic
ACT'	Annual Change Traffic for reusable assets
ADSI	Adapted Delivered Source Instructions
AEC	Application Engineering Cycle
ARBV	Average Retuen on Book Value
ASR	Annual Sales Revenues
BC	Benefit Coefficient
CBS	COTS-Based System
CBSE	Component-Based Software Engineering
CEC	Corporate Engineering Cycle
CF	Cash Flow
CM	Code Modification
COCOMO	Constructive COst MOdel
COCOTS	COnstructive COTS
COEC	COmponent Engineering Cycle
COTS	Commercial Off The Shelf
DC	Development Cost
DEC	Domain Engineering Cycle
DM	Design Modification
DOD	Department Of Defense
EDSI	Equivalent Delivered Source Instructions
ER	Effort for Reuse
EVA	Economic Valued Added
FTSP	Full Time Software Person
GE	Glue code Cost
IB	Initial Benefit
IC	Initial Cost
IM	Integration requireed for Modified software
IRR	Interbal Rate of Return
KDSI	Kilo-Delivered Source Instructions
KSLOC	Thousands of Source Lines Of Code
LI	Library Insertion

MM	Man Month
MN	Maintenance
NC	Number of Copies
NPV	Net Presnet Value
OC	Operating Cost
PB	Payback Value
PI	Profitibility Index
PLE	Product Line Engineering
PM	Person Month
RBP	Relative Black-box Price
RCWR	Relative Cost for Reuse
RI	Revenue Improvement
ROI	Return On Investment
RWP	Relative White-box Price
SEI	Software Engineering Institute
SI	Stock Improvement
SY	Starting Year
TDEV	Time of DEVelopment
TS	Total number of Shares
TTM	Time To Market
VC	Volatility Cost
WACC	Weighted Average Cost of Capital

Table 8: List of Acronyms

Vita

Lin Yang
Apt 6 3036 Van Sansul Ave, San Jose, CA 95128▪304-685-6235 (C)▪Email: lyang_us@yahoo.com

Outstanding Software Research and Development Scientist

Eager to contribute highly applicable skills and abilities in commercial software development. Also strong interest in network security, web service and application, and software reuse. I can start working immediately.

Summary of Qualification

- IT professional with 1+ years of Information Technology experience who provides top-notch service, sets high standards, and exceeds expectations.
- Highly motivated, dependable troubleshooter and problem-solver, excellent communicator and good listener.
- Detail oriented with excellent analytical, problem-solving, organization, and research skills.
- Thinks creatively within practical constraints.
- Customer-focused performer who is committed to quality in every task – from personal interaction with coworkers and users to high level of service provided to company/customer.
- Valued contributor who performs confidently and effectively under pressure and thrives on challenge.
- Enthusiastic learner who quickly grasps concepts and technical skills.

Professional Experience

Software Engineer, Software Reuse Project, *Venus Information Technology*, Beijing, China, May 2002 to July 2002

Enhance software analysis ability and Visual C skills through these activities.

- Played a key role in analyzing software reuse project feasibility for a customized firewall and IDS control system and took initiative of the project and wrote requirements and specifications.
- Independently designed and identified reusable components and built reuse library structure through which reusable assets are inserted and retrieved.
- Directed implementation team in all implementation aspects, including project management, process analysis, workflow design, configuration data set-up, systems interface
- Applied troubleshooting techniques to verify solutions and integrated reusable components into UI control system.
- Deployed methodology to test reliability and safety after integration.

Software Engineer and Web Designer, Metal Search Project, *Metalsite, L.P.,* Pittsburgh, PA, June 1999 to Dec 1999

Enhance dynamic web page design, Cold Fusion, Java Script, MS Access, and Oracle skills through these activities.

- Independently designed and implemented a search project through which customer can search metal products online from database.
- Contributed ideas to improve customer service and suggested ways for web design to interact with customers.

- Designed database for this project and wrote SQL functions to access in MS Access.
- Collaborated on solution to database design and e-business system architecture.
- Evaluated test results of the database after DBA implemented it in Oracle.

Academic Projects Experience

Software Developer, Software Reuse Economic Model Supporting Tool Project, Mar 2004 to present

Enhance Visual Basic and database skills/knowledge through these activities.

- Performed cost-benefit analysis of software reuse estimation process my dissertation which contains a software reuse economics model.
- Streamlined estimation process, enabling idea of implementing a supporting tool approach to complex software reuse estimation.
- Independently design system architecture of a supporting tool.
- Design and create UI for this tool which allows user to input software reuse economics data.
- Design data structure and schema for this data-driven tool based on the mathematic model.
- Implement the tool using Visual Basic and MS Access.
- Develop a questionnaire to collect real-world industrial data.
- Evaluate testing results of the tool by collecting and input data into the program.
- Plan to deliver and distribute the tool to software development companies and collect feedback.

Software Developer, USB EFS Lock Project, Jan 2005 to present

Enhance computer forensics and Visual Basic skills/knowledge through these activities.

- Review Encrypted File System (EFS) and certificate structure.
- Design a computer protected software to protect data from being stolen.
- Implement the software using Visual Basic.

Network Security Analyzer, Network Security and Attack Simulation Project, Jan 2004 to Sep 2004

Enhance network security skills/knowledge and be familiar kinds of hacker tools through these activities.

- Designed network topology.
- Tested and analyzed Cisco network security products, router, firewall, IDS, and VPN.
- Simulated network attack to these appliances by using hacker tools, DDOS, DOS, and
- Analyzed the network vulnerabilities and how to minimize the attack effects.

Web Designer and Developer, Course Web Homepage Project, Jan 2002 to June 2002

Enhance multimedia web design, HTML and Frontpage skills/knowledge through these activities.

- Designed a multimedia web page which contains video, audio, pictures, and PowerPoint slides.
- Built a forum through which people can post articles and discuss.

Areas and Expertise of Programming and Software Applications

- **Language:** Visual C/C++, Visual Basic, VC .NET, VB .NET, Ada, C/C++, HTML, Java, JavaScript, J2EE, and PL/SQL
- **Web Application**: Dynamic web page design, Cold Fusion, HTML, FrontPage, XML

- **Database:** MS Access, Oracle, MySQL
- **Network and Security:** Design topology of Cisco Firewall, LAN/WAN, wireless topology, IDS and router
- **Others:** Latex, Matlab, and Statistical analysis

Education

- Ph.D. in Computer Science, West Virginia University, WV (Expected graduation date: Aug 2005).
- Master in Computer Science, West Virginia University, WV.
- Bachelor in Computer Science, West Virginia University, WV.

Publications

- Robert David Cowan, Ali Mili, Hany Ammar, Alan McKendall Jr., **Lin Yang**, Dapeng Chen, and Terry Spencer, "Software Engineering Technology Watch", IEEE SOFTWARE July / August 2002.
- **Lin Yang**, "Generalization of an Integrated Cost Model and Extensions to COTS, PLE and TTM", Journal of Zhengzhou University (Engineering Edition), China, June 2005.
- **Lin Yang**, "Generalization of an Integrated Cost Model and Extensions to COTS, PLE and TTM", Journal of Henan Science and Technology, China, June 2005.