Enterprise Information Technology Project Portfolio Selection Through System Dynamics Simulations

by

Bin Zhou

B.A., Information Management, Nanjing University, 1991
M.S., Management Information Systems, Carnegie Mellon University, 1997

Submitted to the System Design and Management Program in
Partial Fulfillment of the Requirements for the Degree of

Master of Science in Engineering and Management

at the

Massachusetts Institute of Technology

June 2007

Signature of Author ______________________________
Bin Zhou
System Design and Management Program
May 2007

Certified By ______________________________
Dr. George Westerman
Thesis Supervisor
Sloan Center for Information Systems Research

Certified By [illegible]
Patrick Hale
Director
System Design and Management Program

THIS PAGE INTENTIONALLY LEFT BLANK

Enterprise Information Technology Project Portfolio Selection Through System Dynamics Simulations

by

Bin Zhou

Submitted to the System Design and Management Program
in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Engineering and Management

Abstract

As companies are increasingly relying on information technologies (IT) to help maintain their existing and develop new competitive advantages, investing effectively in IT is becoming more and more important. One of the biggest challenges facing an enterprise IT organization is how to select a project portfolio that is best aligned with the business strategies and to deliver highest value using limited IT resources.

In this research paper, I examined in detail a recently proposed IT governance framework, designed a System Dynamics model based on this framework, and developed a simulation application to investigate constructs, relationships and scenarios suggested by the framework.

My research identified and examined several levers through which IT managers can achieve better alignment with business goals and more efficient use of IT resources. I examined alternative IT governance regimes (combinations of rules and policies for selecting among opportunities and retaining existing systems) in terms of their effects on efficiency, feature satisfaction, and cost of the resulting legacy asset base. By choosing the right combination of relatively straightforward selection and retention policies, IT managers can steer their legacy assets toward a desired efficiency or satisfaction goal in concert with company strategies.

Thesis Supervisor: Dr. George Westerman
Title: Research Scientist, MIT Sloan Center for Information Systems Research

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgements

First and foremost, I am tremendously in debt to my thesis supervisor Dr. George Westerman at the Sloan Center for Information Systems Research, who not only provided me with insights on my thesis topic, but also had many impacts on all aspects during my thesis work. His kindness, wisdom, patience and support have helped make this a wonderful experience of my academic life. In addition, his understanding of my work load and his timely feedback and guidance to my work, despite his extremely overloaded schedule, are appreciated. I feel truly blessed to have had the privilege to work with him.

I also would like to thank Professor James Lyneis, who taught the System Dynamics subject for our class during the summer of 2005. It was a short summer term at MIT; however, what I had learned probably helped forever change my views on many things in this world. I would see not only many independent or separate events as I had before, but also the "loops" and "links" among them.

My thanks go to my wife Xiaojun Yang and my daughter Isabella Zhou whose patience, understanding, sacrifice have been unconditional. They have always been one of the strongest "positive feedback loops" for me, whenever I face challenges in my life.

I want to express my appreciation to the administration of the Systems Design and Management (SDM) program at the Massachusetts Institute of Technology for allowing me the opportunity to be part of the great SDM community. I am especially thankful to SDM coordinator Ted Hoppe for his humor, to Pat Hale for his flexibility, understanding and kindness, to Bill Foley, Christine Bates, and Jeff Shao for their timely support over the course of my study at MIT.

Last but not least, my life at MIT would not have been as rich and meaningful if not being for the chances interacting with my intelligent and unique SDM 2005 fellow classmates. I am really privileged to be able to work with all of them.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 1: Introduction

Background and Motivations

In its survey "State of the CIO 2003", CIO magazine found that CIOs cited aligning IT with business goals—along with prioritizing the demands of various business units—as their greatest challenge for the coming years (Jahnke, 2004). The 2006 Society for Information Management (SIM) annual survey, including more than 300 executives, also rated IT and business alignment as the number one concern (SIM Survey, 2006).

Some argue that IT can effectively align with business by closely following well defined business strategy. However, in reality, we constantly see misalignment between the two. On the one hand, the growing complexity of products and increasing competitiveness of the environment demand that IT deliver new products and services better, faster, and at lower costs. On the other hand, an IT organization always faces the difficulty dealing with the infamous iron triage – scope (what must be built), schedule (when it must be built by), and resources (how much it should cost) – in delivering and maintaining its products and services (Aguanno, 2003).

An astonishing 84 percent of all software projects do not finish on time, on budget, and with all features installed, according to a survey by the Standish Group (1995), which studied about 8,000 software projects from 365 companies in the United States. Furthermore, more than 30 percent of all projects were cancelled before completion. The rest ran significantly over deadline and were 189 percent (on average) over budget (Hoch, Roeding and Lindner, 2000). Even for corporate or enterprise information technology projects, which were supposed to be better planned and executed, the striking picture is that an estimated 68% of them were neither on time nor on budget (Jeffery and Leliveld, 2004). More important, not all of finished enterprise projects fulfilled their business goals or achieved their targeted Return of Investment (ROI). Considering that a typical corporate IT budget may comprise hundreds of possible projects across functions, business units and geographies, it is important to select a project portfolio that is synchronized with the company's strategy, has high Return on Investment (ROI), and mitigates the potential risks (Jeffery and Leliveld, 2004).

Despite improved communication tools such as email, web applications, and team based group-ware applications, alignment between IT and business seems to have NOT been significantly improved. Advances in project analysis and management methodologies and techniques such as Agile project management, Critical Path Analysis/ PERT, Design Structure Matrix, and Real Options Analysis have NOT resolved the alignment issue either. In the mean time, if we look at the alignment from the other side, the business or the financial side, we still see a great divide. According to a survey by CFO magazine, 44% of CFOs surveyed in 2003 described the alignment between IT and business to be weak, and 4% said the alignment did not even exist at all. Overall, about half of them believed that there was much work yet to do on alignment (CFO magazine, 2003).

How does IT project portfolio selection effectively achieve the alignment goal? What are the best models that can help to do so? What are the most important elements or key steps that contribute most to selecting the best IT portfolio for an organization?

During my more than 10 years experiences designing, developing and managing IT applications with several companies in multiple industries, I noticed the dynamic natural of IT portfolio management, the difficulty in understanding extremely complex and multidimensional concepts, and a necessity for an effective framework and a tool to analyze and visualize findings in a meaningful and explainable way.

While studying at Massachusetts Institute of Technology (MIT), I had the chance to learn a subject called System Dynamics, whose concepts of using qualitative analysis (mental simulation, causal relationship, stock and flow modeling, and delay and oscillation analysis) and quantitative tools through computer simulation, turned out to be excellent weapons to look into such a complicated and intertwined issue as IT governance.

Applying the System Dynamics approach to analyzing complex systems is not a brand new subject. Many works have even been developed in the project/product development area. Sterman (2000) described some common issues in project management such as cost overrun, "90% syndrome", in which a project is thought to be 90% complete for more than half of its total time required, and some counter-intuitive situations, one of which is the famous Brook's law described as "adding more resources to a late project makes it even later". Sterman (1992) also introduced the use of System Dynamics modeling to help tackle problems facing project management. Later Ford and Sterman (1997) came up with a System Dynamics model for product development processes. Reicheflt and Lyneis (1999) had a new product development "Rework Cycle" System Dynamics model to explain some behaviors in product development and project management. However, I have yet to see thorough research in the IT governance or IT portfolio management areas using methodologies or approaches of System Dynamics.

When I was doing journal search on potential topics for my degree thesis in this field, I had the chance to read a paper by George Westerman and Raymond Henry (2006) in which they proposed that enterprise IT governance framework could best be modeled in terms of three major distinct yet interrelated processes: variation, selection and retention. Westerman and Henry (2006) claimed that IT managers could achieve and maintain alignment between IT and business strategies by adjusting the three governance processes.

A well defined IT governance framework will help the alignment between IT and business. An understanding of the IT governance regimes and their effects on different dimensions of the resulting applications will help organizations make the right decisions in the life cycle of system development, enabling IT strategies to be aligned more effectively with the business strategies.

In this thesis, I tried to understand the proposed Westerman and Henry framework (2006), its processes, interrelations among those processes, and different results adopting the model with different strategic intents. I mapped out the process model using System Dynamics tools. A simulation application was also designed and developed to help create and analyze different scenarios of this proposed model.

Objectives

The objective of this study is to gain an understanding of the proposed IT governance model at a highly abstract level by using System Dynamics analysis tools. Efforts were spent in analyzing and constructing a System Dynamics model based on the proposed IT governance framework, developing a simulation application based on the model, verifying some key propositions from the model, identifying new findings and making further propositions.

The primary research objectives are:

1. To understand the IT governance framework and its major propositions from the paper of Westerman and Henry (2006).
2. To identify and analyze some key proposed interrelationships and feedback loop structures that play significant roles governing the model from a System Dynamics perspective.
3. To design a flexible System Dynamics model and develop, based on such model, a simulation application to examine the effects of constructs, interrelationships and feedbacks of the IT ecosystem.
4. To test the model through simulations with different scenarios.
5. To find out how IT can coordinate its process regimes through simulation and analysis.
6. To summarize insights gained from this body of research and propose potential policy implications that could help better align IT strategies with intended business goals.

Thesis Outline

To achieve the intended research objectives, the thesis is structured as follows:

- Chapter 1 (this chapter) discuses background, objectives and structure of this thesis.
- Chapter 2 reviews some existing knowledge in the fields of System Dynamics and IT portfolio management, followed by an introduction to the IT governance framework from Westerman and Henry (2006), which is used as the theoretical basis for my model design and implementation.
- Chapter 3 further discusses the model design using System Dynamics tools.
- Chapter 4 formalizes and develops the System Dynamics model, based on the prior discussions of the model.
- Chapter 5 illustrates how applications of a simulation and analysis tool kit are developed and how to use them.
- Chapter 6 presents scenarios running the simulation and analysis tools using different strategies. Some intuitions gained from running those scenarios are also discussed.
- Chapter 7 discusses our findings.
- Chapter 8 points out some further work that can be performed to enhance and expand the model and the applications.
- Appendix contains references and more detailed information on design and development of the simulation and analysis application tool kit.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 2: Knowledge Reviews

This chapter describes the relevant background information and literatures to help understand the work that this thesis is based on. It begins with a brief overview of the field of System Dynamics, followed by an introduction to some key works of IT portfolio management and IT governance processes, including the Westerman and Henry IT governance framework (2006).

System Dynamics Review[1]

The basic System Dynamics theories were introduced by Jay W. Forrester at the MIT Sloan Management School, in his book "Industrial Dynamics" (Forrester, 1961) in the 1960's. It focuses on using control theory to analyze behaviors of non-technical dynamic social systems (world population, inventory management, and project management, etc.). Its premise is that the behavior (reference mode) of a dynamic system is the result of not only the causal effect actions of individual parts in the system, but also the structure (feedback relationships and time delays) and interrelationships of the system. System Dynamics is particularly well suited for the understanding and modeling of complex systems where multiple feedback effects, time delays, and unclear or multiple system interactions determine the overall system behaviors.

In the next few decades, several new concepts and tools had been developed and used in the field of System Dynamics. One of the central concept tools is "Systems Thinking", popularized by Peter Senge (1990) who defined systems thinking as the "fifth" and probably the most important of the five disciplines for understanding the behavior of a system driven by the relationships and interactions of all parts of it. The essence of the discipline of systems thinking lies in a shift of mind seeing interrelationships rather than linear cause-effect chains and seeing processes of change rather than snapshots.

Another concept central to System Dynamics is of the mental models upon which decisions are made. Modeling and simulation is a discipline to develop a level of understanding of the parts and their interactions in a system, and of the system as a whole. There are always some trade-offs on the level of details to be included in a model based on observations and assumptions. We must admit that a model may be built on wrong or partial assumptions that will actually confirm our biases and support incorrect intuitions. Therefore, models are used to only challenge existing formulation, rather than to validate or verify them (Oreskes, Schrader-Frechette, and Belitz, 1994). System Dynamics is thus a tool to help understand dynamic behaviors of a complex system, helping improve our ability using mental models to increase confidence that one is using the best model available for decision making.

The third important aspect of System Dynamics is causal relationship. System behaviors are determined by the structures of its components and interactions among them according to the System Dynamics theory. An increase in value of one component will affect that of another component, either in the same direction (a positive or reinforcing causal relationship) or in an opposite direction (a negative or balancing

[1] This part of review on System Dynamics is mostly drawn from the knowledge I gained from attending System Dynamics and Project Management subjects at the Sloan Management School from the MIT during the summer of 2005.

relationship), all other things hold constant. Those relationships among components are represented by the concept of positive or negative causal feedback loops using Causal Loop Diagram tool (CLD). In a CLD, arrows are marked with a "+" sign to indicate a positive causal relationship, or alternatively a "-" sign to indicate a negative causal relationship. Loops are marked with either "R" to reflect a reinforcing (positive) feedback loop or "B" to designate a balancing (negative) feedback loop. In a CLD, loops are named to help recognize the meaning and impacts of them on a system. Relationships with time delays are marked with double slash arrows in the middle of a link. Causal loops in System Dynamics help understand and predict the behaviors of a system. Some archetypes reflecting positive causal relationships (Success to the Successful, Shifting the Burden, Eroding Goals, and Escalation) and negative relationships (Limit to Growth, Fixes that Failed) are well known and documented thoroughly by academia (Sterman, 2000).

A simplified CLD diagram describes how price affects market supply and demand of a product to help achieve the market equilibrium Figure 1.

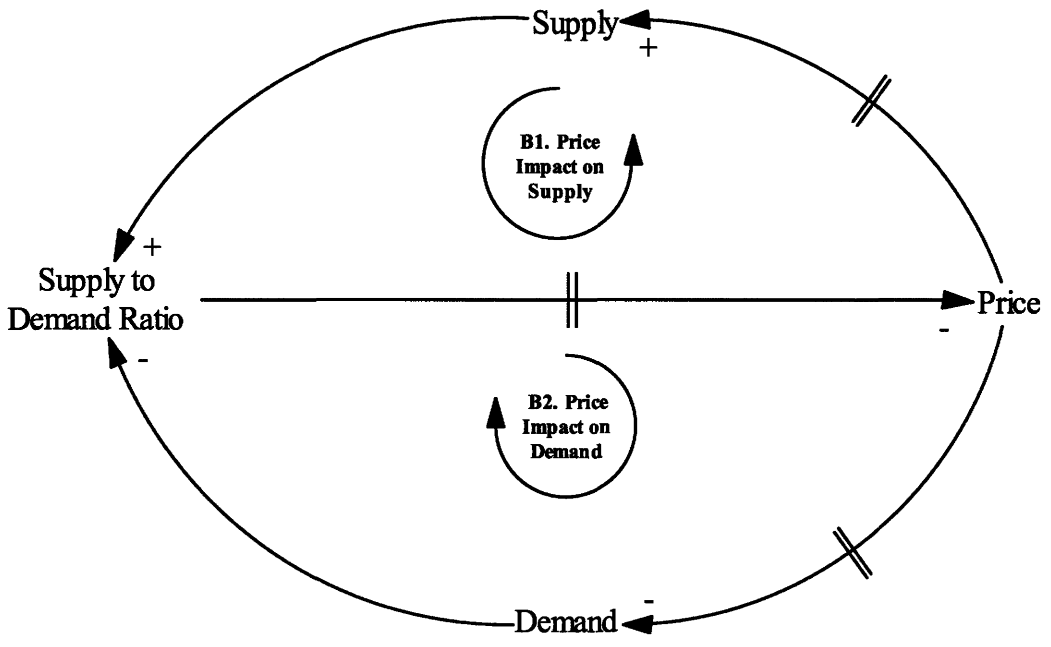


Figure 1: Causal loop diagram of price impacts on supply and demand for a product (adapted from System Dynamics class note, Professor James Lyneis, MIT subject ESD 74, 2005)

As we can see, there are two balancing causal loops in the model to help maintain the equilibrium price for a certain product. When price of the product goes up, supply of the product rises, leading to a higher value of the Supply to Demand Ratio, the increased value of which will help reduce the price. Thus loop B1 Price Impact on Supply is a balancing loop. Similarly, loop B2, Price Impact on Demand is the other balancing loop in the system.

Another useful tool in System Dynamics is the Stock-Flow Diagram (SFD). In the CLD, we described the components of a system and the relationships among them. However, CLD does not incorporate how quickly a value would change, which is illustrated by SFD. Stock models accumulation of something (either concrete concepts such as accumulated water in a bathtub, or an intangible concept such as accumulated loss of interest in watching football games). A Flow, measured as quantity changed per unit time (a day, a month or any other time unit), either accumulates (as inflow) or depletes (as outflow) a stock. Values of flow variables can be changed instantly by some processes while their magnitude of the changes may be affected by the levels of stocks and/or some other flows of the system, or even delay in the system. Shown in Figure 2 is an example of a Stock-Flow Diagram describing the impact of price changes on demand and supply for a product. An example of a flow variable in the model is the Change in Price, which refers to how much the market price of the product changes per time period (for example, in a week). It takes time to change the value of a stock variable, because such change happens through values of flows coming into or going out of the stock. Delays in changes of stock variables are the reason that some systems may oscillate. One example of stock variables in our market equilibrium model is Price, which reflects accumulated Changes in Price of the product. The third type of variables used in SFD is that of Auxiliary or Exogenous variables, either to hold intermediate or calculated values of a stock or flow value (auxiliary variables), or hold values that are outside of the model itself (exogenous variables). An example for auxiliary variable in the Price Impact on Supply and Demand of a Product model is Policy Governing Indicated Price, which acts as a policy regime, determined by external factors (such as the government regulation on price of the product) that are out of the model itself. Many of my analyses on the IT governance model will involve changing the values of regime parameters to shock the system. Introducing the stock-flow structure can help identify more loop structures inside a model (for example B3 Indicated Price Changes in this model) as well.

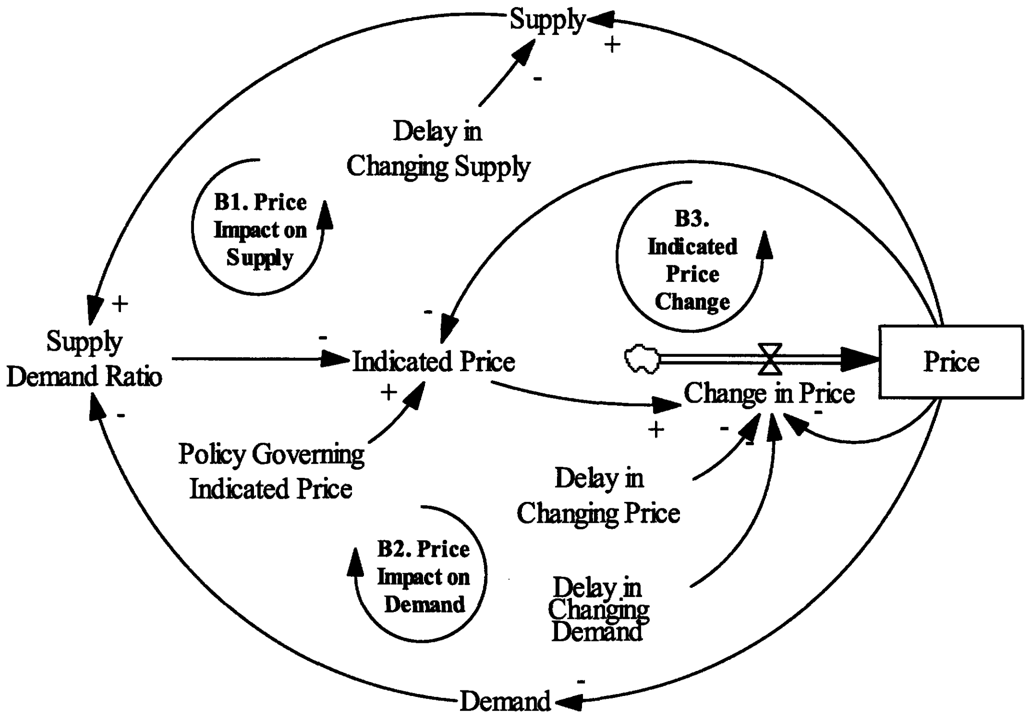


Figure 2: Stock flow diagram of price impact on supply and demand for a product (adapted from System Dynamics class note, Professor James Lyneis, MIT subject ESD 74, 2005)

Systems Thinking, CLD, and Stock-Flow Diagram provide us with mental tools to determine system behaviors and identify the nature of loops so that we can adopt policies to strengthen those loops that drive desired system behaviors while weakening those that drive undesirable ones. Those concepts and tools mentioned above are soft tools because they are trying to help understand problem using qualitative approaches or mental simulations.

While the concepts and soft tools introduced above are useful in helping us to understand the dynamic behaviors of a complex system, the more powerful tool is the application of computer simulations (referred to as hard tools in System Dynamics). Using computer simulations not only helps understand complex, and sometimes, even counter-intuitive system behaviors due to dynamics of multiple loop relationships and system delays, but also allows us to look into the system boundary and find out how aggressively we should implement or change policy regimes. Computer simulations also provide results that are visualized, easy to understand and to analyze.

Figure 3, System Dynamics Modeling Approaches, modified from a class note of my System Dynamics class at MIT (Lyneis, 2005), summarizes the main modeling tools used in System Dynamics. In this thesis, my discussion and modeling on the proposed IT governance framework will follow the sequence marked in the figure as small modeling approach using CLD, Stock-Flow Diagram, and Simulations. Some case studies based on the model would have been extremely useful to help verify the model if we could have had reliable data from typical enterprises such that we can perform a large model calibration. Nevertheless, following the small model route will still be valuable to help us with insights on the understanding. The intuitions behind applications of each of those modeling tools are well supported by prior academic researches and by my own personal experiences working in the IT industry.

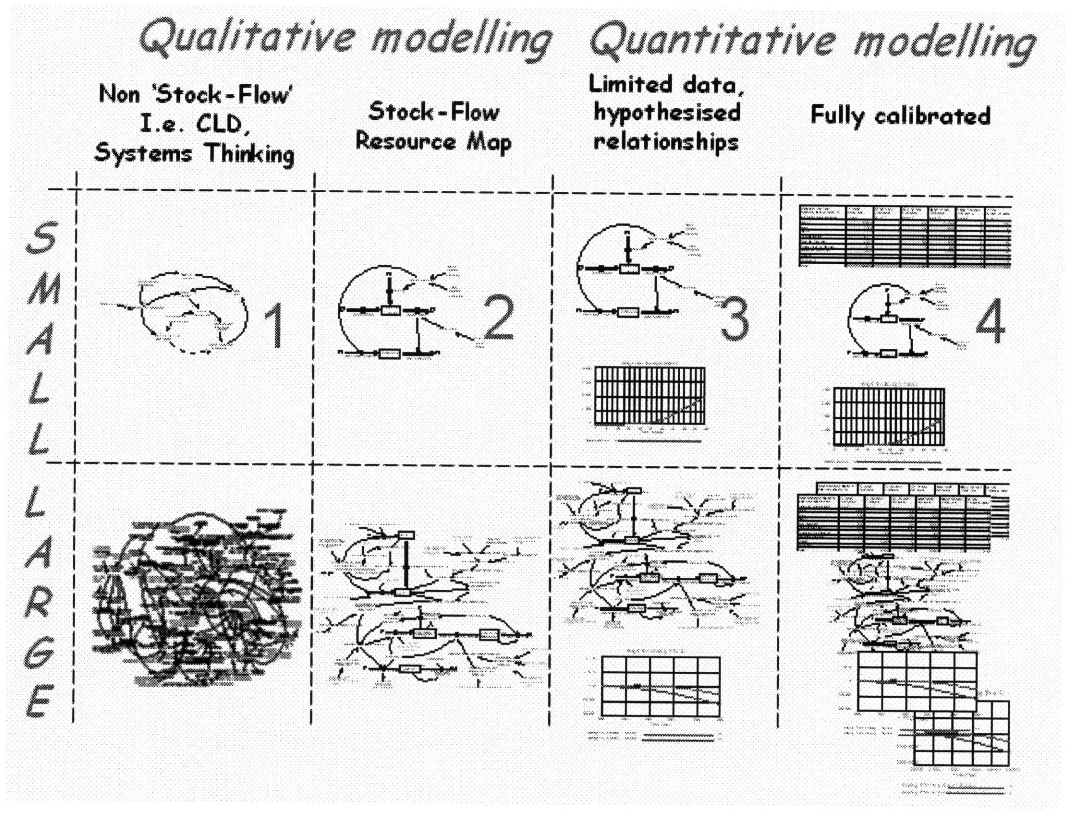


Figure 3: System Dynamics modeling approaches (adapted from System Dynamics class note, Professor James Lyneis, MIT subject ESD 74, 2005)

IT Project Portfolio Management

IT portfolio management was first proposed by McFarlan (1981), who adopted Modern Portfolio Theory (Markowitz, 1952) in IT management domain. McFarlan suggested that managers should allocate IT resources to appropriate IT project portfolio by employing a risk-based approach. At its most mature, IT Portfolio management is accomplished through the creation of two types of portfolios (Wikipedia, 2007). The first is application portfolio, which "contributes to the corporate profitability and some non-financial criteria such as stability, usability, and technical obsolescence". The second type is project portfolio, which focuses on the spending on new initiatives developing new or maintaining existing competitive advantages.

Firm performance is founded on the development of key inimitable IT capabilities and assets (Bharadwaj, 2000). The basic issue of IT portfolio management is to find out how a firm can allocate its IT resources to projects in a way that will maximize values of its IT investments while minimizing the risks. There is considerable research on the subject. The central theme is to measure benefits and costs of a project portfolio and its associated risks. Many methods have been proposed, including scoring techniques (Rengarajan and Jagannathan, 1997), utility theory (Kontio, 1999), the analytical hierarchy process (Levine and Wideman, 2005), the data envelopment analysis (Guan and Yam etc., 2004), and more recently developed decision tree theory (Heidenberger, 1996), game theory (Ali, Kalwani and Kovenock, 1993), simulation (Papageorgiou and Paskov, 1998), or real options (Schwartz and Zozaya-Gorostiza, 2003) approaches.

According to Datz (2003), IT portfolio management consists of processes of gathering information, evaluating portfolio, prioritizing and periodically reviewing on the portfolio. Gathering information process collects all information about projects and project candidates of a firm to build an inventory check list. Evaluating portfolio process identifies items from the inventory check list that match the business goals. Prioritizing process further finds the best projects that match business strategic intents and allocates IT resources upon selected items. Reviewing process monitors selected projects periodically and makes sure they continuously align with the business goals. In order to perform IT portfolio management effectively, IT and business need to work together. Without open communications and full cooperation between IT and business, those tasks can not be successfully fulfilled.

As we can see, IT portfolio management is basically a selection process that allocates resources to develop and sustain those projects that align best to a company's strategic goals. Many researchers have identified some key factors that influence performance, structure and costs associated with an enterprise information technology project portfolio.

Some researchers had investigated how IT budgets affected IT outputs. Gurbaxani, Melville, and Kraemer (2000) had a theoretical model in which a production function is used to transfer the IT inputs—hardware and personnel—into outputs, information services. Since a firm allocates IT budget to acquire hardware and personnel (software), according to their function, IT outputs are determined directly by the IT budget allocation. Although their research focused on finding the substitute relationship between the two categories of inputs mentioned above so that a firm could optimize the combination of them, achieving certain level of outputs at the lowest cost or, given a certain amount of resources to produce largest outputs, their empirical research indicated that IT outputs and budget were positively related. Kudyba and Diwan (2002) used a different production function to calculate relationships between IT inputs and outputs, and acknowledged the similar positive relationship between IT budget and capability of an IT organization to provide services.

Broadbent and Weill (1999) explored the links between firm-wide IT infrastructure and business process change. They argued that enforcement of common IT architecture and standards, development of a common systems development environment and establishment of firm-wide data management enabled cross-boundary applications to be implemented more quickly and easily. Projects conforming to the established or standard technologies and using those common infrastructure services could also be maintained at lower costs.

It is observed that system maintenance costs account for about 60%-80% of system life-cycle costs for complex systems (Banker and Slaughter 1997, Schneidewind 1987). Because of the predominant time and costs spent on maintenance effort, Dekleva (1992) argued that improvement in maintenance activities were even more important than enhancements of the development process. If an IT organization could save budget allocations on the maintenance effort significantly without compromising services to business, it would be more effective through the use of reclaimed maintenance resources to develop new projects.

Some scholars also discussed impacts of project abandonment. Kweku and Zblgnlew (1991) mentioned that there are two types of abandonment in IT, project abandonment, which "abandons projects under development" and system abandonment, which abandons existing systems currently in operation. Project management issues such as staffing, scheduling and communications were most often cited as the reasons for project failure (White, 1984, Pinto and Slevin, 1987). Lyytinen and Hirschheim (1987) proposed a concept of "expectation failure" which described project abandonment as the result of "perceived inability" to meet technical or managerial requirements or expectations. Kweku and Zblgnlew (1991) claimed that "IS project abandonment can occur when problems arise in perceiving, analyzing, designing, or configuring the system objectives; when the technological basis for the system and its behavioral, political, or organizational issues directly or indirectly affect ways to bring the project to a successful completion within the estimated budget and schedule constraints". Abandonment of projects itself may not be a bad practice as it may conserve scarce resources of a company and put them into better or more productive use, if the abandonment is performed on the right projects at the right time (Staw and Ross, 1987).

Another important factor that influences the performance of IT services is the information system or organization design. Many have had discussions on this topic. There are two basic types of IT organization design: centralized and decentralized. Centralized means IT responsibilities belong to a central information technology unit, while decentralized means IT responsibilities belong to business units. A general agreement, according to Brown and Magill (1994), about the primary organizational tradeoff is that "centralization affords greater efficiencies (economies of scale) and standardized controls as well as organizational integration, while decentralization provides local control and ownership of resources as well as greater responsiveness to business unit needs". Henderson and Venkatraman (1992) claimed alignment (with the firm's infrastructure and processes as well as with the IT strategy) pressure may contribute to the choice between centralized or decentralized IT organization. In practice, NO IT organizations are 100% centralized or 100% decentralized.

Although each factor discussed seems to be important in contributing to the performance of an enterprise IT ecosystem, normally the selection of an IT investment portfolio involves evaluating many of them (and many more that did not get covered in this discussion) at the same time. Parker and Benson (1998) considered six factors in evaluating information technology investment: return on investment, strategic match, competitive advantage, system support, competitive response, and strategic system architecture.

I will inspect some of the factors mentioned and their impacts to the IT ecosystem while designing and developing the IT governance model.

Dynamic Nature of IT Portfolio Management Processes: Westerman and Henry (2006) Framework

An IT organization works to bring values to a firm by delivering features for its internal and external business customers. IT governance is the key to the success for the firm to fully leverage the existing IT capability and develop new core competences by maximizing benefits from IT initiatives that represent new business opportunities and challenges. This goal would not be possible if the IT strategy is not aligned with the business strategies.

However, Westerman and Henry (2006) argued that perfect alignment is almost impossible given limited resources that an IT organization has and its role in most organizations. IT is rarely in the position to control or drive the business priorities of the organization. While striving to play a more strategic role in their firms, IT organizations should develop their governance processes to best evaluate and implement the diversity of requests so that the resulting IT assets best meet the strategies of the business,

Westerman and Henry (2006) proposed a conceptual model of structure and behavior for IT governance based on ecological principles. In their model, alignment between IT and business is more than a state; it is a governance process consisting of variation, selection and retention. Business units engage in the variation process that produces a Request Set consisting of potential initiatives with features provided through information technology to maintain or improve the organization's capability. Selection processes, conducted by IT and business units jointly, select projects from the Request Set generated in the variation processes using the selection regime that consists of decision rules (policies). The selection processes are central to performing the optimization, making sure the selected projects will increase the current and future value of the IT assets. Projects selected will be allocated resources necessary for development. Retention processes evaluate and maintain the Retained Set that contains applications to provide features to support the business strategies. Retention processes also have a retention regime to determine how and when existing features should be phased out.

The processes compete to acquire resources. IT governance will shape those processes to evolve toward or away from effective alignment through selection (Westerman and Henry, 2007). At the same time, governance can change the structure and effectiveness of the IT assets (including the retention) and even shape the nature of demand (variation). Figure 4 shows the basic concepts of the Westerman and Henry framework.

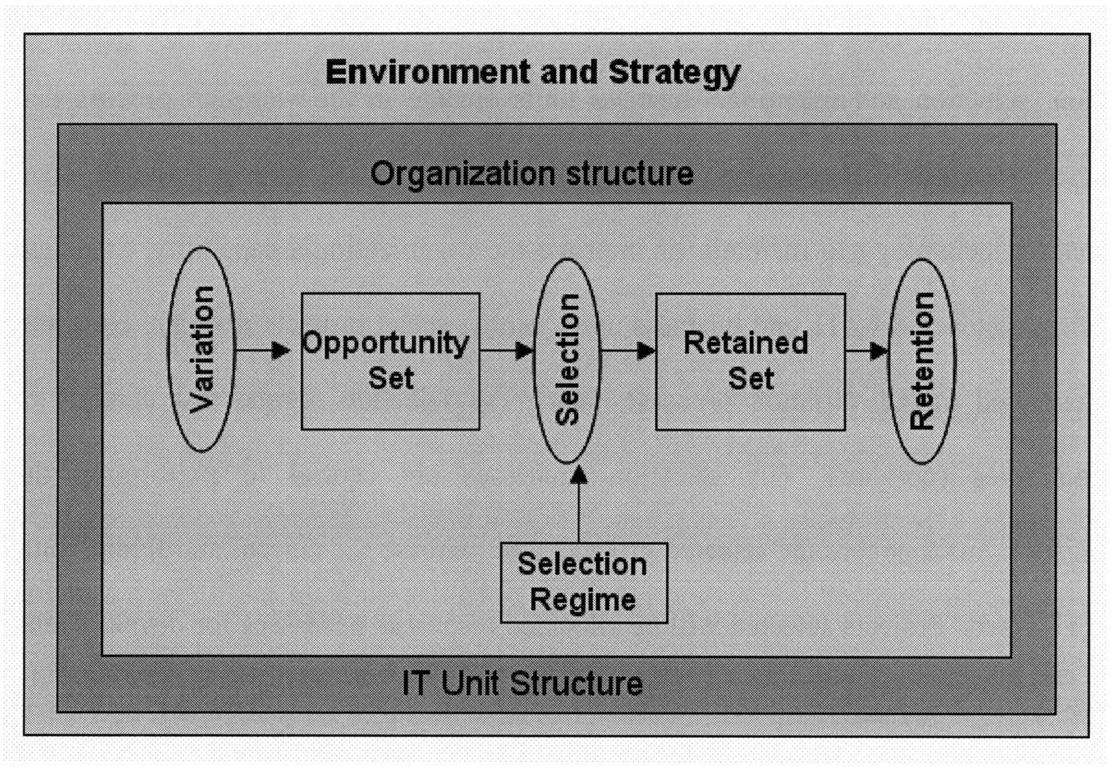


Figure 4: IT governance framework (Westerman and Henry, 2006)

Westerman and Henry finally concluded that, in most organizations, IT will be and should always be a bottleneck. However, that bottleneck can be a useful thing to allow the enterprise to enforce discipline in the process of requesting and implementing changes through IT. Potential variations are channeled into a set of retained applications through well defined selection process that efficiently utilizes resources (Westerman and Henry, 2006).

The three key processes of variation, selection, and retention operate interdependently to evolve and maintain the IT asset base. Based on their proposed model, Westerman and Henry derived several testable propositions, which will serve as the basis for my work. It seems that System Dynamics is a suitable approach to model the IT governance processes, due to the dynamic nature (multiple interrelated processes with delays and causal relationships) described in the framework.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 3: Information Technology Governance System Dynamics Model Design

This chapter discusses in detail how I used System Dynamics tools to analyze the IT governance framework proposed by Westerman and Henry (2006). I first identified and analyzed the concepts described in the paper. Then I presented those concepts using the causal loop diagram to find out the relationships among them. Finally a simplified stock flow diagram is developed to understand the nature and relationships of identified IT ecosystem constructs.

Identify Major Concepts

We have briefly discussed the proposed IT governance framework. To understand and formalize the theory, we first need to identify the constructs and relationships as the basic building blocks for the formalized model. I performed a textual analysis of the paper, followed by several interviews with Dr. Westerman. I identified those statements that seemed to refer to important constructs and relationships. I also tried to identify attributes and determined the types of them so that I could build a System Dynamics model based on the understanding. Table 1 shows an example how I performed the "translation" work and identified constructs, relationships and expected organizational behaviors from a statement of the paper.

Statement in the Paper	Construct	Variable Type	Attributes	Behavior	Relationship
Variation acts upon a Request Set consisting of potential initiatives that the organization sees useful for applying information technology... Each initiative is characterized by a set of features and cost to operate... Initiatives that build upon existing elements in the ecosystem (which we will call them as "conforming" initiatives) are generally lower in cost to implement and operate...	Initiative	Flow	Costs and Features	Variation regime determines the attributes of Initiatives	The higher the conforming of an initiative, the lower in cost to develop and maintain
	Request Set	Stock	Costs and Features		
	Conforming	Auxiliary			

Table 1: Identify constructs, behaviors and relationships for the IT governance model

It is necessary to point out that some constructs are NOT specified in detail in the paper (Westerman and Henry, 2006), but they may be important to understand in formulating the model. For example, the paper mentions, in many places, the concept of policy regimes such as selection regime and retention regime which are central to the proposed theory. However, it does NOT give formal definition or more details on what exactly the regimes are in terms of policies that can be used and how to measure them. In these cases, I had some additional discussions with Dr. Westerman to clarify the concepts.

Another notable fact is that in some cases during the modeling, a general concept may be substituted by a more specific one that is better measurable for practical modeling purposes. For example, I use the term budget rather than the more abstract term resources described in the paper as the parameter in the simulation model. By doing so, I do not have to track many types of resources. Rather, I assume that, how tightly the organization allocates its budget to the IT organization will determine the scarcity of resources that the IT organization can utilize.

Three types of concepts are important to the understanding of the proposed Westerman and Henry IT governance framework (Westerman and Henry, 2006).

The first are core concepts directly relate to the variation, selection and retention processes that are the basis for the proposed framework. Understanding each one of them and their relationships is critical.

Variation

Variation is the initial request for an IT system, generated by a business unit. The variation creation is the starting point of the process. As it moves through processes in the model, the variation changes its state. Initially, a variation is a Proposal when it is proposed by the business unit. It becomes an Initiative or Request when it is moved into the Initiative Set as a development project candidate. The variation becomes a Project once selected and stays in the development stage. After developed and deployed, The variation is an Application. A variation at different stages may have different characteristics too. For example, an application has deployed time, which tells when the variation finishes its status as a project and starts as an application.

Proposal Set

Proposal Set contains all accumulated proposals (both active and inactive) from the business units.

Request Set

Request Set contains all active proposals (All proposals minus those that have been retired) requested from business units. Variations in the set will be used to calculate Requested Features. It is the starting point of the IT governance work flow that I will design and develop.

Initiative Set

Initiative Set contains accumulated Variations that are under review before they either get moved into the Development Set as Projects or get thrown into the Discarded Set.

Discarded Set

Discarded Set contains accumulated discarded variations which had been reviewed by the Selection Regime, but were decided not to be developed. Variations in the Discarded Set have never been into actual development phase.

Development Set

Development Set contains accumulated variations that are currently in the development phase as active development projects. Projects in the set will incur development cost each period as long as they are active in the development stage and are allocated with development budget for the period.

Abandoned Set

Abandoned Set contains accumulated abandoned or failed projects. Projects in this set will not consume resources any more.

Retained Set

Retained Set contains developed and deployed active applications. Applications in this set provide features to support business needs, while incurring maintenance costs for each period.

Retired Set

Retired Set contains all inactive/retired applications that have been phased out from the Retained Set. Items in this set are inactive and will not consume resources.

Budget

The monetary resources allocated to an IT organization for a time period. In our simulation model, budget is the variable used to reflect the degree of munificence of the environment for the IT organization.

Number of Proposals

This variable measures how many variations are proposed to an IT organization as project candidates from business units for a period in the variation process.

Around the core concepts discussed above are some key policy or regime variables that help shape the IT governance processes. Many changes to the IT ecosystem are triggered through modifications of those policy or regime variables.

Conformance Threshold

Conformance, according to Westerman and Henry (2006), describes the degree in which a variation matches existing IT architecture standards, frameworks and development methodologies. All else equal, the higher the Conformance of a Variation, the lower the development and maintenance costs. Conformance Threshold is the minimum value on Conformance that a Variation needs to have in order to be considered as a Project candidate if the IT organization applies the rule to force conforming. It is an exogenous policy variable in our model.

Variation Regime

Variation Regime determines how variations are generated and proposed from business for each period.

Conforming Regime

Conforming Regime determines how the conforming factor, in the variation, selection and retention processes, alters and filters variations.

Budget Regime

Budget Regime determines how IT budget will be determined, grow and get allocated for an IT organization.

Selection Regime

Selection Regime determines whether a variation in the Initiative Set should go into the Development Set and how to discard variations into the Discarded Set. In my simplified model design and implementation, removing initiatives that have stayed longer in the Initiative Set than the value of the Delay in Discarding Initiative parameter is the rule I use to discard initiatives.

Retention Regime

Retention Regime determines how to retire applications from the Retained Set.

Abandonment Regime

Abandonment Regime determines how projects would fail and get abandoned into the Abandoned Set.

Delay in Reviewing Initiative Threshold

Delay in Reviewing Initiative threshold is the minimum time in periods for a Variation to stay in the Initiative Set before being considered for any possible development endeavor on it. It is an exogenous policy variable in our model.

Delay in Discarding Initiative Threshold

The minimum number of time in periods for a Variation to stay in the Initiative Set before it is considered to be discarded into the Discarded Set. It is an exogenous policy variable in our model.

Delay in Abandoning Project Threshold

The minimum number of time in periods for a Project to stay in the Development Set before it is reviewed on whether it should be abandoned. It is an exogenous policy variable in our model.

Delay in Retiring Application Threshold

The minimum number of times in periods that an Application should keep active after it was delivered into the Retained Set. It is an exogenous policy variable in our model.

Decay Rate on Requested Features

Requested features may lose their values gradually as time goes by. Decay Rate on Requested features measures how many percentage points of features in the Request Set will reduce its value after elapse of a period. If the decay rate is 5%, after each period, the number of features will decrease by 5%.

Decay Rate on Perceived Features

Similar to the Decay Rate on Perceived Features, this parameter measures corresponding concept on the perceived features.

Delay in Decay

Normally new requests or applications will not start decay on features immediately. Delay in Decay specifies when decays in number of features start. For example, if an application comes into being at period 10 and the Delay in Decay is 5, then the feature decay for application will start at period 15. If the initial number of features of the application is 100 and the Decay Rate on Perceived Features is 5%, then at period 16, the application will only provide 95 features.

In addition, we also have the third type of variables that consists of those that help us measure health of an IT ecosystem. They are derived from the core concepts and provide us a way to observe impacts on IT core variables through regime changes:

Average Application Conformance

A parameter reflects how well applications in the Retained Set utilize conforming technologies. It is calculated using weighted average of Conformance on maintained features on active applications in the Retained Set.

Development Spending

The spending on development effort for a period is calculated by summing up the Working Development Unit Cost of all projects in the Development Set.

Maintenance Spending

The spending on maintenance effort for a period is calculated by summing up the Working Maintenance Unit Cost of all applications in the Retained Set.

Cumulated Wasted Development Spending

Cumulated Wasted Development Spending is calculated by summing up development costs on projects that fail to make into the Retained Set.

Requested Features

Requested Features measures total number of features requested by business units that have NOT been fulfilled. It is calculated by summing up features of all variations in the Request Set.

Perceived Features

Perceived Features measures total number of features provided by active applications in the Retained Set. To calculate this value, we sum up Working Number of Features of all applications in the Retained Set.

Analyze Major Process

As argued in the working paper, IT governance is not a state, but rather iterations that evolve the following dynamic processes:

Variation Process

Each period, business units propose a set of Variations as Proposals, whose number is determined by the value of the random variable Number of Proposals for the period. The characteristics (Development Base Unit Cost, Maintenance Base Unit Cost, and Expected Duration for Development etc.) of each Proposal are influenced by variation regime parameters such as Conforming Impacts.

Selection Process

An Initiative can have three possible state transitions. First, it may move into the Discarded Set. Second, an Initiative may become a Project and move into the Development Set. Alternatively an Initiative can stay in the Initiative Set for some periods before it is reviewed.

An on-going Project can have three possible state transitions as well. The first is to become a failed project thrown into the Abandoned Set following the Abandonment Regime check. Alternatively, a project can be fully developed as an application and makes it into the Retained Set to provide with features to business units. Of course, every project will need some time to stay in the Development Set for development. While in the Development Set, active projects consume development resources each period.

Selection Regime determines the state transitions and status changes of a variation in the selection process in above mentioned processes.

Retention Process

After some time, an active application in the Retained Set may lose its value and move into the Retired Set, determined by the Retention Regime

Other Related Processes

While the three processes (variation, selection and retention) listed above are the main focuses of our discussion, some other processes that change model parameter values may affect the course or behavior of the ecosystem as well. For example, changes in values such as that for the Conformance Threshold may influence the result of the next periods of the ecosystem.

Figure 5 describes the IT ecosystem main constructs and processes from high level.

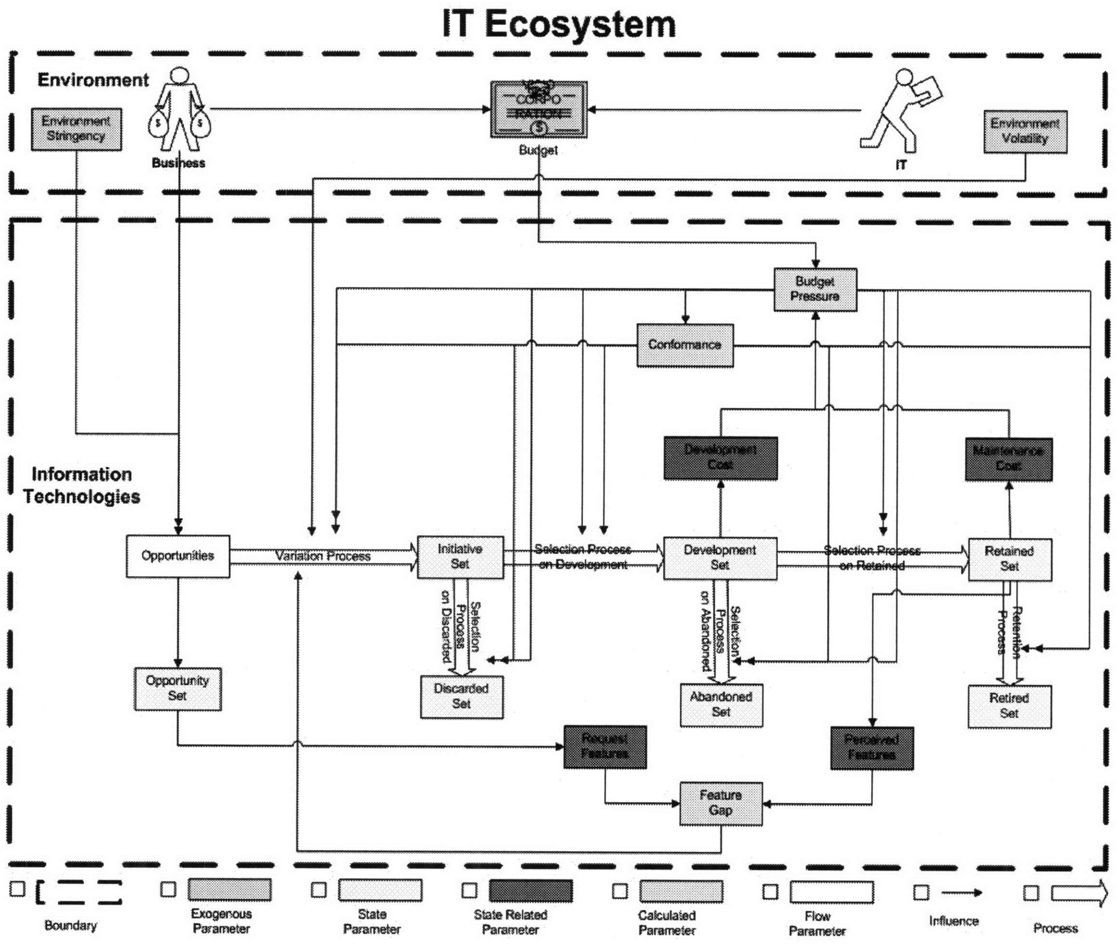


Figure 5: IT ecosystem main constructs and processes

The upper dotted rectangle contains stake holders and processes that determine values of exogenous and environmental parameters in green boxes such as budget and time span under consideration.

The Information Technologies boundary within the lower dotted rectangle is the dynamic part of the ecosystem upon which we are doing the field of research and modeling using the System Dynamics approach. Parameters in the yellow boxes such as the Initiative Set and the Retained Set are stock variables. Those in red boxes are auxiliary variables related to those stock variables as characteristics, for example, Development Cost is related to the Development Set as it is the sum of Working Unit

Development Cost for all projects in the set. Grey boxes are calculated variables that also have feedback impacts to the ecosystem. For example, value of budget pressure, calculated by comparing budget allocation and the total amount of the Development Spending and the Maintenance Spending, will influence the future state of the ecosystem through its impacts on several processes. Parameters in white boxes are the flow parameters. In this high level description, I only show one flow parameter as Proposals and hide the others for to avoid clutter in the chart. Lines with arrows are used to designate influences and the direction of the impacts, for example, Conformance affects the Selection Process on development. Pipelines with arrows are used to show the process flow, for example, the line between the Initiative Set and the Development Set shows that through the Selection Process, some initiatives may become projects.

Construct Causal Loop Structure of the Model

Once we understand the main constructs and processes of the IT ecosystem model, the next step is to construct a causal loop structure to link all the major variables we have identified and show the relationships among them.

According to Westerman and Henry (2006), effective information technology governance is an ecological process of variation, selection, and retention in an ongoing struggle for resources. Figure 6 below describes those processes and the relationships among key variables involved, using the System Dynamics standard causal loop diagramming tool, which marks the impacts of change in value of one variable to that of another by "+" or "-" If the change to the other variable is in the same direction, then we use the "+" sign, otherwise, use "-" . In a typical causal loop diagram, one of the most important aspects is to identify the impacts of changes in values of one variable to itself in a complete loop structure. Choosing any variable in a loop structure, loop polarity is determined by navigating along the loop until the full circuit is completed. Positive loops are those that the change will reinforce the change initially made. Negative loops are those that the change made will counteract the initial change made.

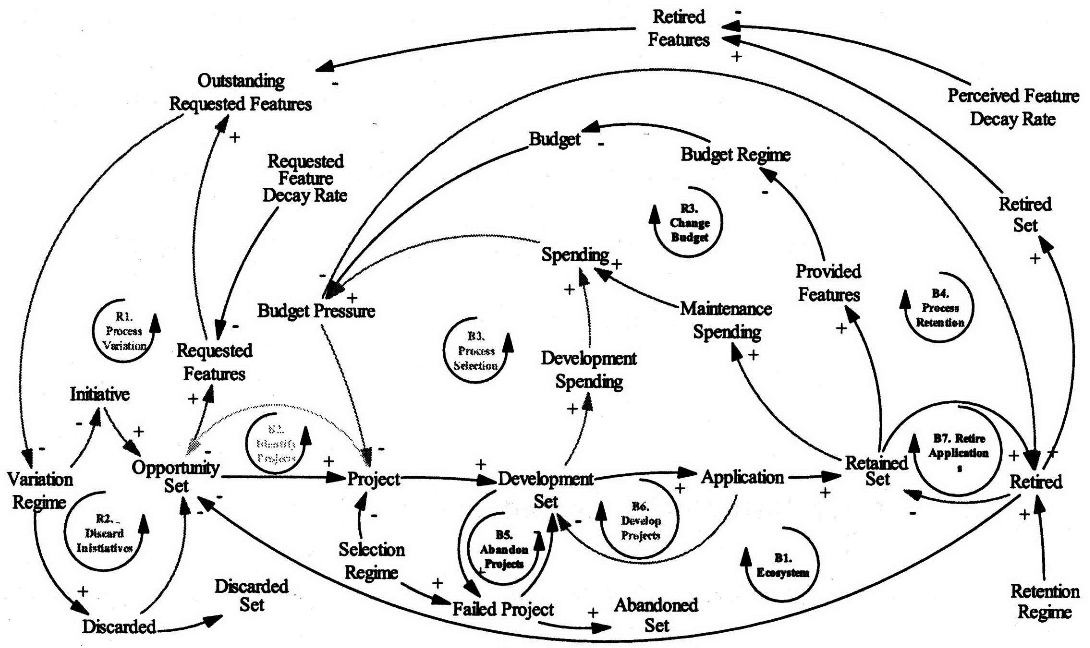


Figure 6: IT ecosystem causal loop diagram

Although the IT governance processes are continuous, without explicit starting and ending points, to simplify our analysis I will start from the negative loop B1 (Ecosystem) in the diagram. Basically, without constraints, all initiatives in the Proposal Set will become development projects and finally applications. However, because of the budget constraint and possibility of failed projects shown here, the ability for IT to develop new projects and maintain existing applications is restrained. Negative loop B3 (Process Selection) will reduce the number of development projects once the system face budget pressure. In addition, negative loop B4 (Process Retention) may speed up the retirement when budget pressure is high. B3 and B4 are self correcting negative loops that help maintain the ecosystem to relatively stable states in terms of budget consumption, features maintained and new project development. Given a fixed amount of budget, we can see from this causal loop diagram, that there are several competitions for resources in the IT ecosystem. Projects in the Development Set will compete on which one to keep and which one to fail. Applications in the Retained Set will compete on which one to maintain and which one to retire. Between the Development Set and the Retained Set, there is also competition on how much IT budget will be spent on the specific effort.

In the mean time, there is also a positive loop R1 (Process Variations) in the model. Business constantly has the need for maintaining existing features and developing new features that require IT resources. Those requests, if unsatisfied, will lead to more Initiatives or Requests that will be accumulated in the Request Set, determined by the Variation Regime. The more outstanding features business is looking to acquire, the more proposals. Given the framework I model the ecosystem dynamics, IT has limited budgets while facing seemingly unlimited business requests, without Variation Regime or Budget Regime change, IT will always be a bottleneck. Positive loop R1 (Process Variation) reflects this scenario as it is almost always the case that only a subset of proposals could be developed and maintained even if we do have significant levels of IT budget available. Furthermore, another positive loop R2 (Discard Initiative) reinforces the effect of the positive loop R1 in that, the higher the number of outstanding requested features, the weaker the variation regime is, therefore the system will create fewer discarded initiatives. These two positive loops show the situation where IT requests will create even more requests.

Finally, depending on the budget regime, there is potentially another positive loop R3 (Change Budget) in the model. If the budget regime is set to grow budget along with the growth of the number of features provided, the faster the growth in the number of features, the more increase in the budget for the next period, thus possibly further more delivered features.

Although I discuss those dynamic loops as if they were independent to each other; however, they actually work simultaneously in the IT ecosystem. Some variables may even be part of several loops. For example, the Development Set is part of both the Ecosystem loop (B1) and the Process Selection (B3), as indicated by the causal loop diagram.

Build Stock Flow Diagram for the IT Ecosystem Model

I constructed a stock flow diagram so that we can look into the model in more details in term of the natures of those variables that construct the IT ecosystem, before I can design and implement a simulation application.

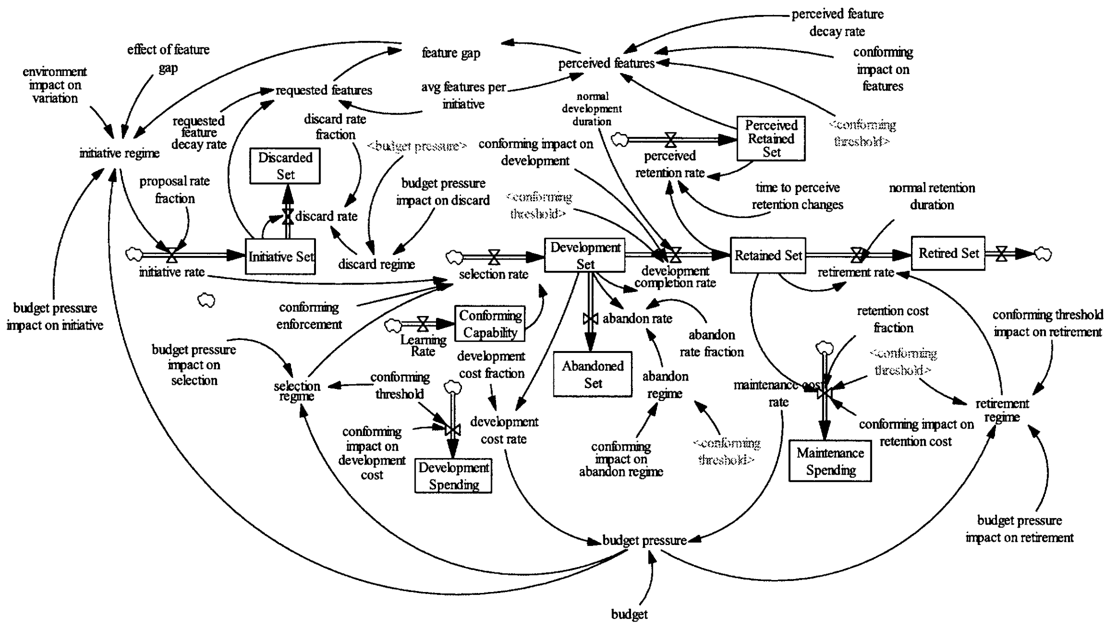


Figure 7: IT ecosystem stock flow diagram

The model consists of several major sections that represent variation, selection, and retention processes and some minor sections that center around those major sections. Each section includes stock, flow and auxiliary variables and their interactions determine dynamic behaviors of the ecosystem. We will discuss those key stock, flow and exogenous variables in the next chapter where we perform detailed design of the IT ecosystem model. Stock flow diagram helps determine the data structures to use in our system design later.

Chapter 4: Information Technology Ecosystem Simulation Model Development

Goals and Scope

The purpose of developing the simulation model is to formalize the proposed framework based on the understanding on the dynamics of IT ecosystem discussed in prior sections. The design and application should at least:

- Consist of major constructs and relationships discussed in the proposed framework.
- Reflect propositions or behaviors proposed by the model.
- Help simulate different scenarios consistent with observations from our experiences.
- Allow future expansions and upgrades

In this phase of design and implementation, the simulation model focuses on the structures and dynamic behaviors of Selection Process and Retention Process. The first reason to do so is that these are the two processes falling into the direct control or influence of an IT organization and are the focus from an IT organization's perspective. In addition, the working paper (Westerman and Henry, 2006) and my discussions with Dr. Westerman have covered not only qualitative but also some quantitative aspects on how the system would work. In this sense, we have information on model formalization for those sections. Last but not least, even though I just designed but did not implement the variation process in feedback loop structures, I did implement the variation section in such a way that a user can to some degree change how variation process is performed through the application setting changes. For example, a user can specify how initiatives would be proposed through changing system settings to allow customization on attributes of those initiatives created. Westerman and Henry (2006) also argue that IT is not in a role of dictating how business would create variations, but rather of utilizing selection process effectively to improve the structure and effectiveness of the IT portfolio assets. In this sense, not specifically implementing the variation process as a loop structure internal to the model leaves the room for flexibility for a user to decide how variation process works.

General Assumptions

Without doubt, IT ecosystem is one of the most complex systems that we can image. There are potentially hundreds of variables and many more processes or loops that we can model. In order to make the model implementation more manageable while still reflect the nature or core of its dynamic behaviors, we would necessarily build the model under some assumptions.

1. IT governance takes a sequence of actions to progress from variation to retention to selection during one simulation period.
2. In each simulation period, the IT governance goes through each of the major variation, selection, retention processes only once.
3. All Perceived Features are indifferent to users in our analysis process. Under this setting, the higher is the number of provided features, the more satisfaction for business users.

4. Every project will take at least one period (in our case, one quarter) to finish its development.
5. If Normalization option is chosen, then all projects will have the same conforming value as that of the Conformance Threshold parameter after this normalization process. In this case, conformance factor is not going to take part in the selection process. Rather, other parameters, such as development cost, maintenance cost, and number of features etc. will determine whether a variation will be selected in the ecosystem.
6. Assume at the beginning of the simulation, all stock variables are initialized at the value of zero.

The purposes of the above simplifications were not to weaken the power of our simulation model; rather just to make the model easier to implement. Furthermore, some assumptions may be easily relaxed in the future development to make the model more flexible.

Formalization of the Model

In this section, we detail major constructs, their formulation, and descriptions of processes using these constructs to construct our model.

Variation

The Variation concept is central to our model and contains the following fields that determine its characteristics:

- Variation Identifier: Variation Identifier is a unique number created by the simulation program to identify a Variation. This identifier is used for convenient track of the Variation along different phases of processes in the ecosystem.
- Conformance Value: Conformance Value is a value between zero and one. Zero means the Variation will be totally using non-conforming technologies to implement and operate, while one meaning using only conforming technologies. This attribute is implemented as a random variable of a Normal Distribution with adjustment to prevent the value from being below zero.
- Base Development Unit Cost: Base Development Unit Cost, modeled as a random variable of a Normal Distribution, stands for money spent on a Variation per period when it is in the development stage as a Project. This cost is called base because it has not been adjusted by the Impact of Conformance value on the variation.
- Base Maintenance Unit Cost: Base Maintenance Unit Cost, modeled as a random variable of a Normal Distribution, stands for money spent on a Variation per period when it is in the maintenance stage as an Application. This cost is called base because it has not been adjusted by the Impact of Conformance value on the variation.
- Base Number of Features: Base Number of Features of a Variation, which is not adjusted by the Conformance Impact on the value of the

variation, is modeled as a random variable drawn from a Normal Distribution.

- Conforming Effect on Development Unit Cost: Conforming Effect on Development Unit Cost is a value drawn from a Normal Distribution (mean at one) that is used to adjust upon Based Development Unit Cost to derive Working Development Unit Cost.
- Conforming Effect on Maintenance Unit Cost: Conforming Effect on Maintenance Unit Cost is a value drawn from a Normal Distribution (mean at one) that is used to adjust upon Based Maintenance Unit Cost to derive Working Maintenance Unit Cost.
- Conforming Effect on Number of Features: Conforming Effect on Number of Features is a value drawn from a Normal Distribution (mean at one) that is used to adjust upon Based Number of Features of a variation to derive Working Number of Features.
- Working Development Unit Cost: Working Development Unit Cost, used as development unit cost for each period when a variation is in the development phase, is an adjusted value based on the Base Development Unit Cost. Since the impact of conformance of a variation to development cost is not certain, we adjust the value by a random value.
- Working Maintenance Unit Cost: Working Maintenance Unit Cost, used as actual maintenance unit cost for each period when a Variation is in maintenance phase, is an adjusted value based on Base Maintenance

Unit Cost. It is commonly observed that the more conforming a variation is, the lower the maintenance cost in general.

- Working Number of Features: Working Number of Features, used as the actual number of features for each Variation in model calculation as if it will be developed and operated using conforming technologies, is an adjusted value based on Base Number of Features. The less conforming is the original variation, the lower of the number of features after the adjustment.
- Expected Development Duration: Expected Development Duration, modeled as a random variable of a Normal Distribution, is the number of expected periods needed to develop and deliver a project.
- Time Proposed: Time Proposed is a variable recording the period in which a variation is proposed in the simulation.
- Time Selected: Time Selected is a variable recording period in which a variation is selected as a development project and moved into the Development Set. This value could be different from the value of the Time Proposed, if a Delay in Reviewing Initiative happened after the variation was proposed. This value would not be populated if the variation was discarded.
- Time Discarded: Time Discarded is a variable recording the period in which a project was discarded from the Initiative Set and moved into the Discarded Set.

- Time Deployed: Time Deployed is a variable recording the period in which a project finished development and was deployed as an application into the Retained Set. This value helps us apply Retirement Regime later
- Time Abandoned: Time Abandoned is a variable recording the period in which a project was abandoned from the Development Set. Abandoned projects went into the Abandoned Set.
- Time Retired: Time Retired is a variable recording the period in which an application retired from the Retained Set and went into the Retired Set. A retired application stops consuming resources.
- Development Spending: Development Spending is a variable recording accumulated development spending on a project through its life cycle. It is the sum of all Working Development Unit Cost incurred developing the project, from the period when the project was selected until current period, or the period when either the project was delivered or abandoned, whichever was earlier.
- Maintenance Spending: Maintenance Spending is a variable recording accumulated maintenance spending on an application when it is maintained and operated. It is the sum of all Working Maintenance Unit Costs incurred after the application was deployed.
- Base Failure Probability: Base Failure Probability is a random variable describing how likely a project may fail during the development phase. A failed project will be abandoned.

- Conforming Effect on Failure Probability: Conforming Effect on Failure Probability is a value from a Normal Distribution (mean at one) that is used to adjust upon Based Failure Probability to derive Working Failure Probability for a project.
- Working Failure Probability: Working Failure Probability, used as the actual likelihood a project may fail, is an adjusted value based on the Base Failure Probability and the Conforming Effect on Failure Probability of the project.

Proposal Set

Proposal Set keeps accumulating all proposals from business units, as the result, it has only one inflow but no outflow. The formula to calculate the number of proposals in it is simply:

$$O_t = \sum_0^{t-1} o_i$$

Where o_i is the rate of proposals at period *i*.

Request Set

As long as business units request new initiatives, they flow into the Request Set. Request Set only depletes its content when some variations retire. In our model, Request Set R is a stock variable representing the accumulation of all the active requests that have ever been proposed. At time t, the value of the Request Set is the integral of all net requests, difference between the rate of proposals and that of retirements, from all the previous periods. Since our model is not based on continuous changes, but rather allows only discrete ones, the formula to calculate the number of requests in it is as follows at time *t*:

$$R_t = \sum_0^{t-1} o_i - r_i$$

Where o_i is the rate of proposals and r_i the rate of retirement respectively for period *i*.

Initiative Set

Initiative Set contains all the active requests that still need to determine whether further development work should be perform upon them. In addition to the same inflow and outflow as these for the Request Set, Initiative Set has two additional outflows, new projects that go into the Development Set and discarded initiatives that go into the Discarded Set. The formula for the number of initiatives in it at time *t* is given by the following formula:

$$I_t = \sum_0^{t-1} o_i - r_i - p_i - d_i$$

Where o_i is the rate of proposals, r_i the rate of retirement, p_i the rate of new projects, and d_i the rate of discarded initiatives respectively for period *i*.

Discarded Set

The Discarded Set has only one inflow, discarded initiatives, in our model. Accordingly, the formula on the number of discarded initiatives is simply:

$$D_t = \sum_0^{t-1} d_i$$

Development Set

The **Development Set** tracks all projects that under development stage for an IT organization. The change in number of projects for a period for this set depends on new projects coming into it, the abandoned or failed project going out of it, and the completed or finished projects going out of it. Therefore, the formula for number of projects in the **Development Set** is the following:

$$P_t = \sum_0^{t-1} p_i - a_i - f_i$$

Where p_i is the rate of new projects, a_i the rate abandoning failed projects, and f_i the rate of completed or finished projects respectively for period *i*.

Retained Set

The **Retained Set** has all finished active projects that provide with features for the business. Accordingly, the formula to calculate the number of application in the **Retained Set** is:

$$T_t = \sum_0^{t-1} f_i - r_i$$

Where f_i is the rate of finished new projects and r_i the rate of retirement of applications respectively for period *i*.

Abandoned Set and Retired Set

Similar to the **Discarded Set**, the **Abandoned Set** and the **Retired Set** have no outflows for them. We simply give their formulas here as the following:

$$A_t = \sum_0^{t-1} a_i$$

$R_t = \sum_0^{t-1} r_i$

Mathematically, we also have the following two equations in terms of relationships for those stock variables at any given period:

$O_t = Q_t + R_t$ (Proposals are the sum of all requests and all retired) and

$Q_t = I_t + D_t + P_t + A_t + T_t$ (requests is the sum of initiatives, discarded, projects, abandoned projects, and retained applications).

Development Spending

The value of Development Spending for a Period is the sum of Working Development Unit Cost for all active projects in the Development Set.

$$DS_t = \sum_{j=1}^{N} DC_{j,t}$$

Where $DC_{j,t}$ is Working Development Unit Cost on project j for period t.

Number of Features under Development

The value of Number of Features under Development for a period is the sum of Working Number of Features for all active projects in the Development Set.

$$FD_t = \sum_{j=1}^{N} FP_{j,t}$$

Where $FP_{j,\,t}$ is the value of Working Number of Features for project j for period t.

Weighted Average Spending by Development Features

Accordingly, the value of Weighted Average Spending by Development Features

$$\text{WADS}_t = \sum_{j=1}^{N}(FP_{j,t} \times DC_{j,t}) \div \sum_{j=1}^{N} FP_{j,t}$$

Where $\text{FP}_{j,t}$ is the Number of Development Features of project j for period t; $\text{DC}_{j,t}$ is Working Development Unit Cost of project j for period t.

Maintenance Spending for a Period

The value of Maintenance Spending for a period is the sum of Working Maintenance Unit Cost for all active applications in the Retained Set.

$$\text{MS}_t = \sum_{j=1}^{N} MC_{j,t}$$

Where $\text{MC}_{j,t}$ is Working Maintenance Unit Cost on application j for the period t.

Features Perceived

The value of Features Perceived for a period is the sum of Working Number of Features for all active applications in the Retained Set.

$$\text{FM}_t = \sum_{j=1}^{N} FT_{j,t}$$

Where $\text{FT}_{j,t}$ is the value of Working Number of Features for application j for period t.

Weighted Average Spending by Maintained Features

Accordingly, the value of Weighted Average Spending by Maintained Features

$$\text{WAMS}_t = \sum_{j=1}^{N} (FT_{j,t} \times MC_{j,t}) \div \sum_{j=1}^{N} FT_{j,t}$$

Where $FT_{j,t}$ is the Number of Maintained Features for application j for period t; $MC_{j,t}$ is Working Maintenance Unit Cost of project j for period t.

Weighted Average Conformance by Maintained Features

Accordingly, the value of Weighted Average Conformance by Maintained Features

$$\text{WAMC}_t = \sum_{j=1}^{N} (FT_{j,t} \times CT_{j,t}) \div \sum_{j=1}^{N} FT_{j,t}$$

Where $FT_{j,t}$ is the Number of Maintained Features of application j for period t; $CT_{j,t}$ is Conformance of application j for period t.

The simulation application has three levels in its process hierarchy. The highest level is the simulation which contains one or more iterations which in turn each has one or more running periods. One running period is characterized as the duration in which the IT governance processes (variation, selection and retention etc.) each operates once. For example, for a particular IT organization, it will go through the major IT governance process one full cycle once a month. A period is therefore one month long. I use multiple iterations in simulation to help mitigate data generation bias. At the summary phase, results from multiple iterations for the same period will be pooled together to calculate averages to be used as the measures of our simulation results. The execution of simulation for the same period over multiple iterations will use same policy or regime settings.

For example, a base simulation case in our research has 100 iterations, each comprising of 80 periods. In this design, each period stands for one month in time scale. During each month, variation, selection, and retention processes each will happen once guided by regime settings for the period. In a different period, distinct regime settings may be used. Multiple periods can also share a set of same regime settings determined by the user. Simulation will run 100 times before it finishes. In order to find out the number of features after the simulation for all 80 periods, we basically find the arithmetic mean from the 100 iterations on all the values of number of features for the same period. Figure 6 shows the high level system work flow for our simulation processes.

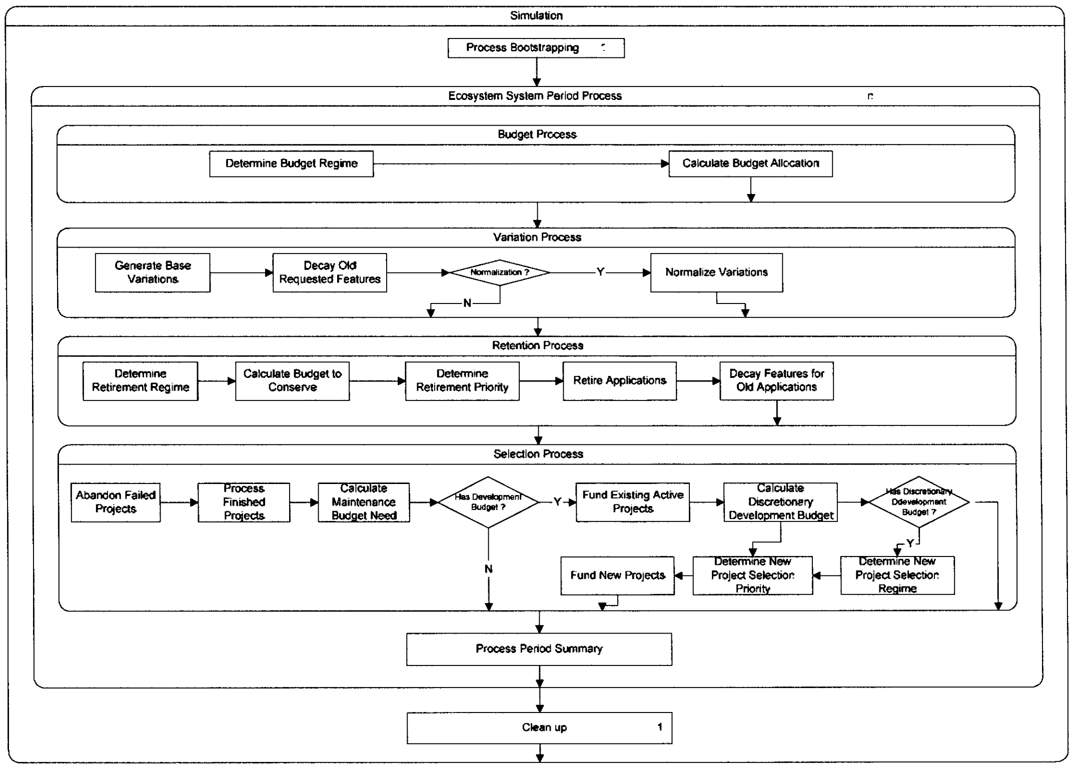

Simulation
Process Bootstrapping
Ecosystem System Period Process
Budget Process
Determine Budget Regime
Calculate Budget Allocation
Variation Process
Generate Base Variations
Decay Old Requested Features
Normalization ?
Y
N
Normalize Variations
Retention Process
Determine Retirement Regime
Calculate Budget to Conserve
Determine Retirement Priority
Retire Applications
Decay Features for Old Applications
Selection Process
Abandon Failed Projects
Process Finished Projects
Calculate Maintenance Budget Need
Has Development Budget ?
Y
N
Fund Existing Active Projects
Calculate Discretionary Development Budget
Has Discretionary Ddevelopment Budget ?
Y
Determine New Project Selection Regime
Determine New Project Selection Priority
Fund New Projects
Process Period Summary
Clean up

Figure 8: IT governance system process flow chart

Variation Process

Variation processes model generation of variations from business units for a period. It includes the following main processes:

Generate Variations

Each period, business units present scores of proposals to start the variation process. The number of variations created for each period is determined by running a random number generator. All attributes of a variation described in the pervious section are populated.

To simplify the implementation, all data that have random nature are generated from the Gaussian random number generator. However, it is very easy to substitute other types of random number generator (e.g. Uniform) for the Gaussian. To allow maximum flexibility in data generation, each such random number variable has its own dedicated generator whose properties are determined by random number generation seed, mean, and standard deviation that all can be specified by the user before running the simulation. In this step, simulation uses the configuration parameters to create certain number of variations as potential project candidates. Also in this step, if we want to adjust values of some parameters by the conforming factor of the variation and conforming threshold, we will normalize values of those parameters and use adjusted values of them such as Working Number of Features and Working Development Cost etc for calculations in the simulation. If the user chooses to perform the Normalization, the following processes will happen:

Adjust Development Unit Cost

We will use Conforming Impact on Development Unit Cost to adjust development cost. Since this adjustment can bring up or bring down the development unit cost and we scale the Conforming Impact on Development Unit Cost using a normal distribution with mean of one, the adjusted Working Development Unit Cost is evaluated simply using the following formula

Working Development Unit Cost = max (0, Base Development Unit Cost × Conforming Effect on Development Cost)

At the first glance, the formula may seem linear to the value of Base Development Unit Cost. However, since the value of Conforming Effect on Development Cost is a random variable, the value of the Working Development Unit Cost will be the result of multiplication of two random numbers.

Adjust Maintenance Unit Cost

Using Conforming Impact on Maintenance Unit Cost to evaluate the value as follows:

Working Maintenance Unit Cost = Base Maintenance Unit Cost × (1 -.max (0, Conformance Threshold - Conformance of the Variation))

The basic intuition for this adjustment is that the lower the conforming value of a variation; the more reduction in its maintenance cost after we make the variation to be conforming. For example, if the Conformance value of the variation is 0.4 while the Conformance Threshold is 0.6, then the value of the Working Maintenance Unit Cost becomes Conforming Effect on Maintenance times Base Maintenance Unit Cost adjusted by a factor of 0.8 (1 – (0.6-0.4)), a maintenance cost reduction of 20%. In this formulation, if a variation has a conformance value higher than the Conformance Threshold, the adjustment will not have any effect.

Adjust Number of Features

I used Conformance value of the variation and Conformance Threshold to adjust the number of features should user force to normalize the parameters.

Working Number of Features = Conforming Effect on Features × max (0, Base Number of Features × (Conformance of the Variation / Conforming Threshold))

Basically, the more conforming a variation is, the less feature will be lost, if we perform normalization.

Adjust Failure Probability

Each project will have a Base Failure Probability, populated by a random number generator. However, if the user chooses to perform normalization on variation data, then we will adjust it to derive Working Failure Probability as follows:

Working Failure Probability = max (0, Base Failure Probability + Conformance Impact on Failure Probability × (100% – Variation Conformance Value))

As we can see, the higher the variation Conformance value, the less likely would a project fail per this adjustment.

Adjust Conformance on Variations

If the user chooses to normalize variation as described above, then all variations will have the same value of Conformance as that of the Conformance Threshold parameter for the period. In this case, Conformance values of variations will not have impact in the optimization process since virtually all of them are the same.

Retention Process

Retention processes find out which applications to retire for the Retained Set. In this step, simulator will perform the following:

<u>Check Budget Amount needed to conserve through Retirement for the period</u>

If no retirement percentage is specified, then stop the retention process here. In this case, the ecosystem will not retire any applications. However, if the user decides that certain percentage of applications will retire for each period, the application will calculate the minimum amount of budget that needs to be conserved through retirement. This amount equals

$$\text{Maintenance Cost} \times \text{Retire \% of maintenance cost}$$

<u>Prioritize Retirement of Applications</u>

We sort applications according to the retirement regime rule for the period. The sorting logic can be chosen by Conformance (the lower is the Conformance of a variation, the earlier in the list for it to retire) or by maintenance cost (the higher is the maintenance cost for an application, the earlier for it to be in the list), or by ROI (the lower is the value, the earlier for it to retire) which is calculated roughly with the following formula:

$$\text{ROI} = \text{Working Number of Features} / \text{Working Maintenance Cost}$$

<u>Retire Application following the Retirement Priority List</u>

The sorting process described in the previous step basically created a priority list for retirement. Here we check the budget conserved after the retirement of an application from the top of this retirement priority list. If cumulated amount of budget conserved from retirement so far is equal to or more than the amount targeted to conserve, stop the retirement process. Otherwise, repeat this step against the next application. When deciding whether an application should retire, we also look at the application and consider retiring it only if its Service Time is longer than the value of the value of the Delay in Retiring Threshold parameter. This delay models the situation that realizing the loss of value of an application and taking the action to retire it will need some time. The value of Service Time for an application is calculated as the following:

Service Time = Current Time – Time Deployed

Selection Process

Selection processes decide which projects to abandon and which initiatives should be selected as new development projects, as well as find out which initiatives to discard for a period.

Check Abandonment Regime

An active project in the Development Set will consume development resources. An on-going project may fail and be thrown into the Abandoned Set after the Abandonment Regime check. But some projects may finish as applications and make it into the Retained Set to provide with Perceived Features. The abandonment rule used in the simulation is to find projects that have higher Working Failure Probability values than that of the Allowed Failure Probability parameter and that have been in the Development Set (Current time – Time Selected) longer than the value of the Delay in

Abandoning Project parameter. The reason to have a delay in abandonment is that we will give any project at least a minimum time under development for possible delivery.

Deliver Finished Projects

This process checks the Development Set and moves all finished projects into the Application Set. The rule to determine whether a project should be delivered is to assume that every project will be completed if it has been allocated development budget in more periods than Expected Development Duration value for the variation. In each period, we check whether a project will be allocated development budget. If true, we will add one period of development time to its working periods. Under this assumption, if a project has a working period for development with a value larger than or equal to the expected development duration, it will go into the Retained Set and become a delivered application.

Check IT Budget for the Period

Budget stands for the amount of monetary resources that a firm has to invest in IT for a period. In reality some other types of resources (technologies, patents, or government permits etc.) will also influence effectiveness and efficiency of IT. However, in order to make the model easier to manage and understand at this research stage, our design of the model does not consider the effects of those other factors. It is also reasonable to assume that we can, in most cases, possibly use monetary resources to acquire other types of resources we need to develop IT capability. Through the control of the amount of budget allocation, we can perform simulations with different scenarios in terms of how munificent an environment in which the IT organization operates. The higher is the amount of the budget allocation, the more abundant is the IT resources. In our simulation model, the values of budget variables may be fixed or adjusted through the simulation periods according to different decision rules as the following:

i) Straight-line growth (percentage change can be specified before the start or during the simulation run). We allow for zero or negative numbers for no growth or resource shrinking scenarios.

   Budget (t_1) = Budget (t_0) × (1 + Budget Increase Percentage)

ii) Budget increase at the same percentage growth as that of number of perceived features increases. This is a proxy to growing as a % of revenue in reality and can roughly be formulated as:

   Budget (t_2) = Budget (t_1) × (Number of Features (t_1) ÷ Number of Features (t_0))

Calculate Committed Maintenance Budget for the Next Period

I use the value of the sum of Working Maintenance Unit Cost on those that are in the Retained Set as the measure of committed maintenance budget. In practice, most organizations have the highest priority on support for existing applications and call them lights-on. This value will be the light-on budget that an organization needs to fund first in maintaining active applications for the next period. The calculation of Committed Maintenance Budget is given when we discuss how to get MS_t in the previous section.

Calculate Committed Development Budget for the Period

I use the value of the sum of Working Development Unit Cost on those that are already in the Development Set as the committed development budget for the period. This portion of budget will have higher priority to fund before an IT organization can allocate resource to select new projects for the period. The calculation of Committed Development Budget is given when we discuss how to get DS_t in the previous section.

Calculate the Amount of Discretionary Development Budget

In the previous steps, we have already determined the budget allocation and calculated the maintenance budget and the committed development budget needed for the period, so we can find out the discretionary budget that can be allocated for new project development as follows:

Discretionary Budget Amount = Budget Allocation - Maintenance Budget Amount - Committed Development Budget Amount

Selection of new projects will only happen if we have a positive value for Discretionary Development Budget.

While performing budget calculations, the simulation uses budget lower and upper bounds, two parameters set by the user to help determine whether we are below or above the budget usage. If the committed budget given as

Committed Budget = maintenance budget + committed development budget

for a period is lower than the value of the Budget Lower Bound, then we have discretionary budget for new project development. On the other hand, if the committed budget for a period is higher than the Budget Upper Bound if one more new project is selected, then there will be no selection of new project for the period. We use the lower and upper bounds to better reflect the reality that, budget allocation and control in most organizations would have some flexibility within some certain range.

Check Selection Regime

Our model first checks the number of time periods for a variation to have been in the Initiative Set, moves a variation into the Discarded Set, if it has been in the Initiative Set for too long (longer than the value of the Delay in Discarding Initiative Threshold since it was in the Initiative Set), e.g. when the following is true

Current Time – Time Proposed > Delay in Discarding Initiative

The intuition behind this discard regime rule is that if the variation has been in the Initiative Set long enough but had not been selected as a new development project, either it has low priority for the organization or it is impossible to develop and maintain the variation effectively and/or efficiently under the current environment; therefore, the variation is deemed as out of scope.

In the mean time, consider a variation as a potential project candidate only if it has been in the Initiative Set longer than the value of the Delay in Reviewing Initiative Threshold, because an Initiative will not be reviewed immediately right after it gets into the Initiative Set due to delays in the process. The length of the open window for an initiative to be chosen into the Development Set is therefore the value of

Delay in Discarding Initiatives – Delay in Review Initiatives

Among those initiatives that have passed the review regime and conformance checks, projects will be chosen by the selection priority specified by the user. New projects can be chosen by the sequence in time initiatives are proposed, by conformance, by return on investment, or by number of features provided. As we have discussed, selection of an initiative as a new project should not lead to budget overrun above the

Upper Budget Bound. Selection regime related to the selection process for a period can be specified as one of the following alternatives:

- Selecting Projects by Proposed Time: If this option is checked, it requires selecting projects from the Initiative Set by the order of the time that initiatives are proposed.
- Selecting Projects by Expected Features: If this option is checked, it gives higher priority to initiatives that provide the highest number of features.
- Selecting Projects by ROI: This requires spending the development budget on initiatives that have the highest value of ROI (ratio of Working Number of Features over Working Maintenance Unit Cost).

In conjunction to using one of the above selection criteria, the user can also choose to apply the Forcing Conforming rule. If checked, only initiatives that have Conformance values higher than that of the Conformance Threshold will be considered to make it to the development stage. For example, if the user requires that selection process apply a conformance threshold at 0.5, any variations with lower conformance values will not be considered as project candidates, even through they may fit well for other selection rules.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 5: Design and Development of the Simulation and Analysis Tools

This chapter discusses the architecture and design and development for the simulation and analysis tools of the IT ecosystem applications.

System Architecture and Main Components

The application suite, developed based on client server architecture, consists of two applications that specialize in simulation and analysis respectively. The high level system modules are shown in the following diagram (Figure 9).

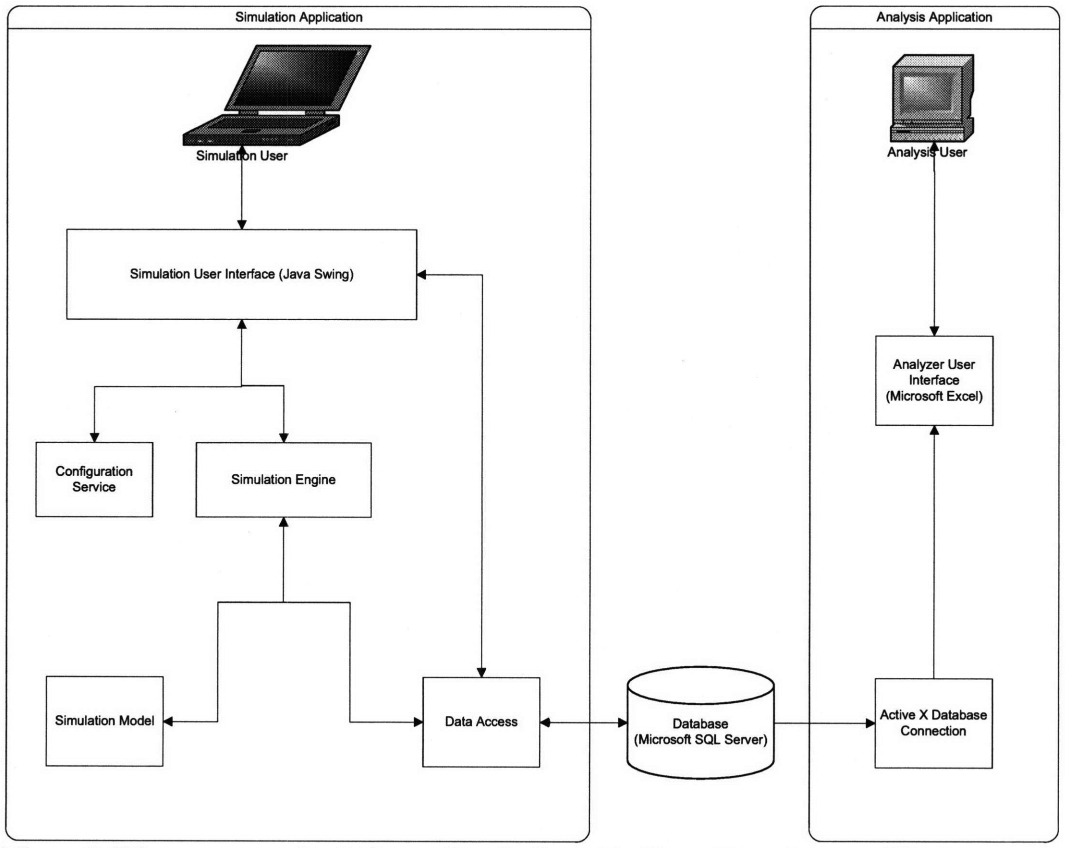

Simulation Application
Simulation User
Simulation User Interface (Java Swing)
Configuration Service
Simulation Engine
Simulation Model
Data Access
Database (Microsoft SQL Server)
Analysis Application
Analysis User
Analyzer User Interface (Microsoft Excel)
Active X Database Connection

Figure 9: IT governance simulation and analysis application suite systems architecture

The simulation application (left section in the diagram) was implemented in Java based technologies (Swing and JDBC). Data created in the execution of simulations are saved into a Microsoft SQL Server database for later use. The first reason implementing simulation application as a Java based application is due to the versatility of the technology to develop the user interface function and simulation capability through well defined application framework such as Swing for GUI and JDBC for data access. Another reason we choose the technology is that Java has strong third party (mostly free) component support and robust build-in data structures that help design and implement the application. For example, to achieve better performance and scalability, a Java connection pool component DBBroker, implemented by some open source software developers at the opensource.devdaily.com, is used for data access between the simulation application and the Microsoft SQL database server.

The analysis application, which is implemented as a Microsoft Excel Visual Basic Application (VBA), allows users easy and flexible access to the data created in the simulation step. Since data analysis will be performed best through tables and chart illustrations, Microsoft Excel is one of the best tools that suit the need.

The software packages used to develop the applications mentioned above are most either open source tools (such as Java and DBBroker) or free tools for research and academic use (such as Microsoft SQL Server 2005 Express Edition). The only commercial software used, Windows operating system and Microsoft Excel, is ubiquitously installed on almost all computers used by users or potential users of the applications as I know. System requirements to run the applications and software package used are listed in appendix.

Ecosystem Database

Data generated by the simulation application are saved in the ecosystem database tables which will be accessed later to perform analysis by the analysis tool. Data related to simulation configurations and simulation results are normalized and partitioned into several database tables, balancing the need for easy data access and the requirement for satisfactory performance.

Simulation Table

The Simulation table is used to save simulation level configuration information when user runs the simulation. One simulation run will create a new entry in this table. The entries saved will be used by the analysis tool to create a list for simulations against which a user can choose to run analysis.

Summary Table

The Summary table contains simulation results that will be used by the analysis tool to perform scenario analysis. Each period of simulation will create a new entry as simulation result data inserted into this table.

Budget Table

The Budget table saves how IT budget is allocated among maintenance and development efforts. The table also saves budget lower bound and budget upper bound specified by the user for each period. If we need to track details on how IT budget is allocated and assigned, this table will be very handy.

PeriodOption Table

The PeriodOption table saves the user input options for running one simulation period. Data on configurations and regime settings for each period are saved in this table.

Variation Table

The Variation table has all the variation data created through the variation process.

VariationStatus Table

Some information associated with variation data will be changed in different period during a simulation run. For example, Working Number of Features and Working Development Unit Cost of a variation may be different if the user chooses to perform Normalization. Therefore, we use the VariationStatus table to save those period dependent values for variations. By inspecting details in this table, we can find out status changes (e.g. proposal → project → application→) for any variation.

StatusCode Table

This is the table used to translate variation status description into status code or vise verse. The purpose of using this table, rather than save the names of variation status in the VariationStatus table directly, is to reduce duplicate data to make future change on status description easier.

The following database table diagram (Figure 10) describes the relationships among those tables. In this diagram, the links between two tables reflect relationships.

The table on the side of a link that has a key sign means that some field(s) in this table is used as a foreign key in the table on the other side of the link. The relationships help maintain data referential integrity and improve data access performance. For example, field SimulationID in the Simulation table is a foreign key of the PeriodOption table. Any entry in the PeriodOption table needs to have a corresponding entry in the Simulation table that has the same SimulationID, before it can be created.

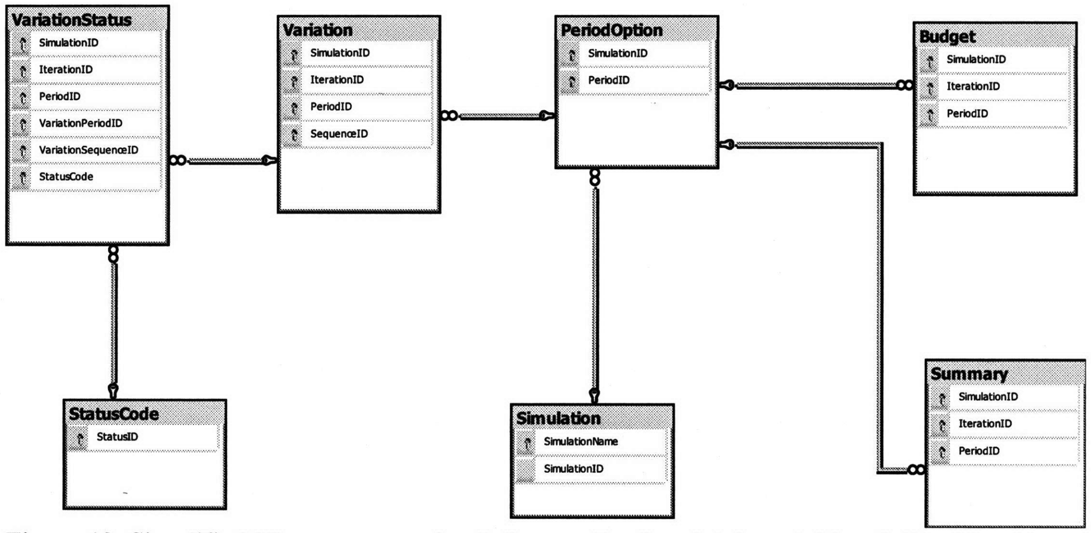


Figure 10: Simplified IT governance simulation application database table relationships diagram

For simplicity and clarity reason, I only show the key fields of each table here. To see more detail, a diagram that shows all the table fields is included in appendix.

Through adopting database technology to persist simulation data, we can decouple the simulation and the analysis applications in design and development so that we can adopt more flexible design, achieve better performance and use best of the breed on tools to perform either task.

Simulation Application Screens and Main Operations

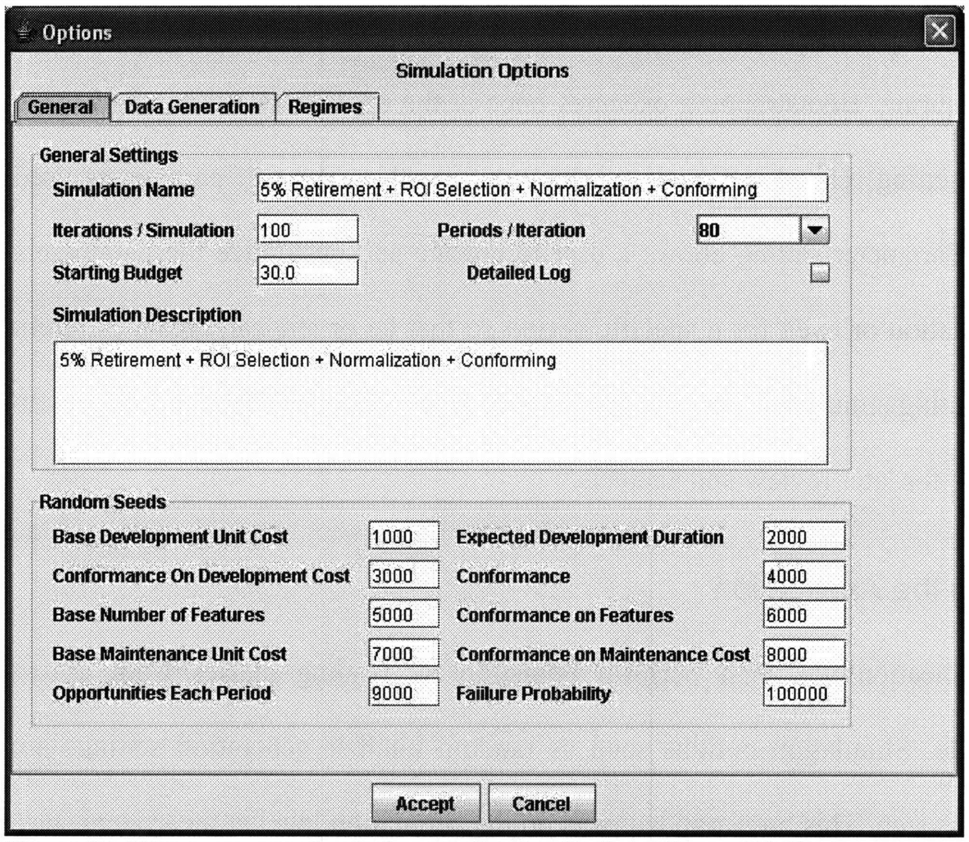
Performing a simulation is a three step operations of bootstrapping, parameter setting, and executing. Bootstrapping executes once at the application startup and loads the application setting related information which defines the values of parameters used for simulation. Parameter setting allows a user to change and customize the parameters used for a simulation or even for a specific period so that he or she can create different scenarios. Executing simulation runs the application, generates result data and save them into the database for later review and analysis.

Bootstrapping the Application

The application will first perform bootstrap by reading entries from a text configuration file. Simulation options such as random number generation settings are configured in this step. This bootstrap happens on the simulation level at the system start. The simulation settings from the pervious simulation run will be loaded as the default setting.

Setting Simulation Parameters

A user will be able to make changes on simulation settings after the bootstrapping. He or she can modify general simulation configurations from the "General" tab (Figure 11) under the Options menu by specifying the name used for the simulation run and periods to perform the simulation as well as the starting budget allocation and logging detail level for the simulation. The user can also fill in a description and set random number generator seeds for the simulation run. Attributes set on this page will not change during one simulation run.



General Settings

Simulation Name	5% Retirement + ROI Selection + Normalization + Conforming		
Iterations / Simulation	100	Periods / Iteration	80
Starting Budget	30.0	Detailed Log	☐

Simulation Description

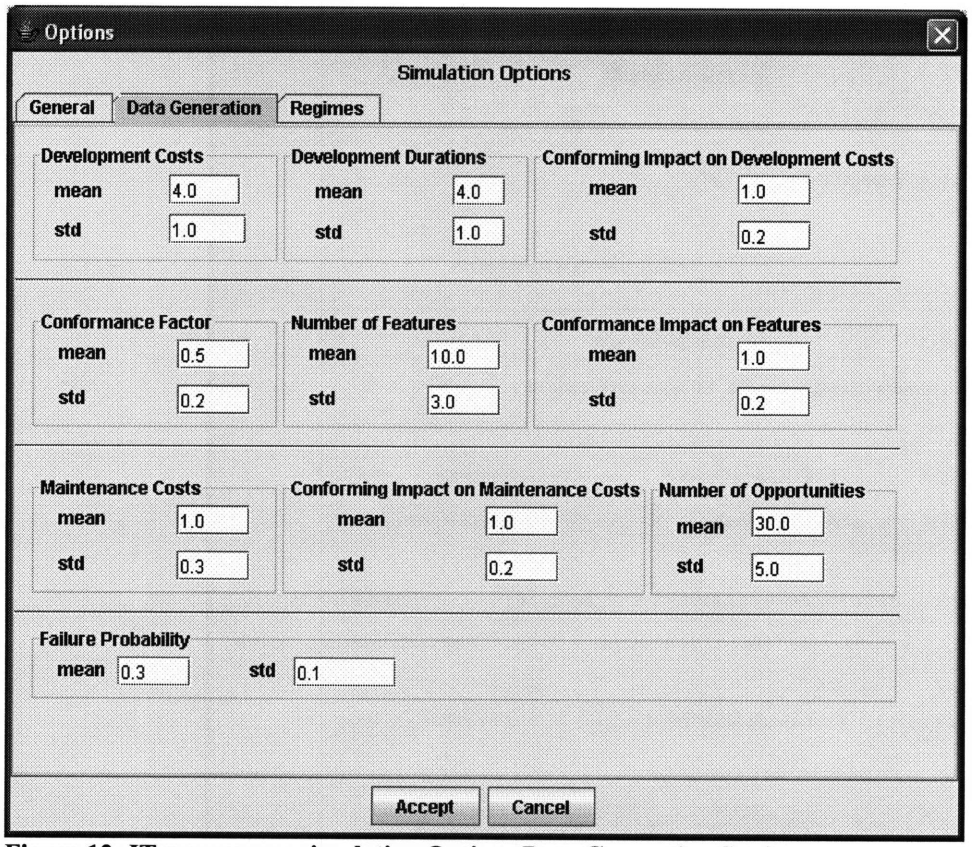
5% Retirement + ROI Selection + Normalization + Conforming

Random Seeds

Base Development Unit Cost	1000	Expected Development Duration	2000
Conformance On Development Cost	3000	Conformance	4000
Base Number of Features	5000	Conformance on Features	6000
Base Maintenance Unit Cost	7000	Conformance on Maintenance Cost	8000
Opportunities Each Period	9000	Faiilure Probability	100000

Figure 11: IT governance simulation application Options General Settings screen

The user can set mean and standard deviation values for random number generators used for the simulation run from the "Data Generation" tab (Figure 12).



Options

Simulation Options

General | Data Generation | Regimes

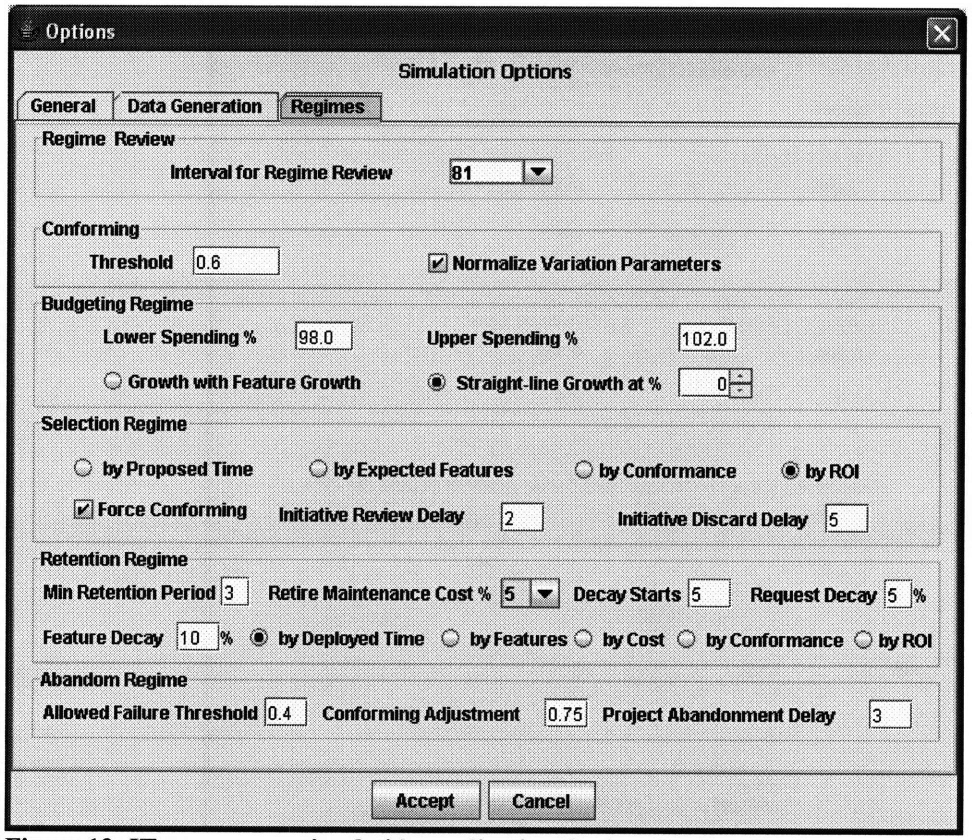
Development Costs			**Development Durations**			**Conforming Impact on Development Costs**		
mean	4.0		mean	4.0		mean	1.0	
std	1.0		std	1.0		std	0.2	
Conformance Factor			**Number of Features**			**Conformance Impact on Features**		
mean	0.5		mean	10.0		mean	1.0	
std	0.2		std	3.0		std	0.2	
Maintenance Costs			**Conforming Impact on Maintenance Costs**			**Number of Opportunities**		
mean	1.0		mean	1.0		mean	30.0	
std	0.3		std	0.2		std	5.0	

Failure Probability
mean 0.3 std 0.1

Accept Cancel

Figure 12: IT governance simulation Options Data Generation Settings screen

The user can also set all the IT ecosystem regime parameter values from the "Regimes" tab (Figure 13). Interval for Regime Review is used to allow the user to intervene during a simulation execution. If a number N which is less than the total periods of simulation is specified here, for every N periods during the simulation run, the execution will temporally be suspended to allow the user to use the configuration screen to make some changes that may alter the simulation results for the following periods. In the sample screen, since the total number of periods for the simulation is 80 and the user sets a value of 81 here, the simulation will not be interrupted before it is finished.



Regime Review

Interval for Regime Review: 81

Conforming

Threshold: 0.6 [✓ Normalize Variation Parameters]

Budgeting Regime

Lower Spending %: 98.0 Upper Spending %: 102.0

Growth with Feature Growth [✓ Straight-line Growth at %] 0

Selection Regime

by Proposed Time by Expected Features by Conformance [✓ by ROI]

[✓ Force Conforming] Initiative Review Delay: 2 Initiative Discard Delay: 5

Retention Regime

Min Retention Period: 3 Retire Maintenance Cost %: 5 Decay Starts: 5 Request Decay: 5 %

Feature Decay: 10 % [✓ by Deployed Time] by Features by Cost by Conformance by ROI

Abandom Regime

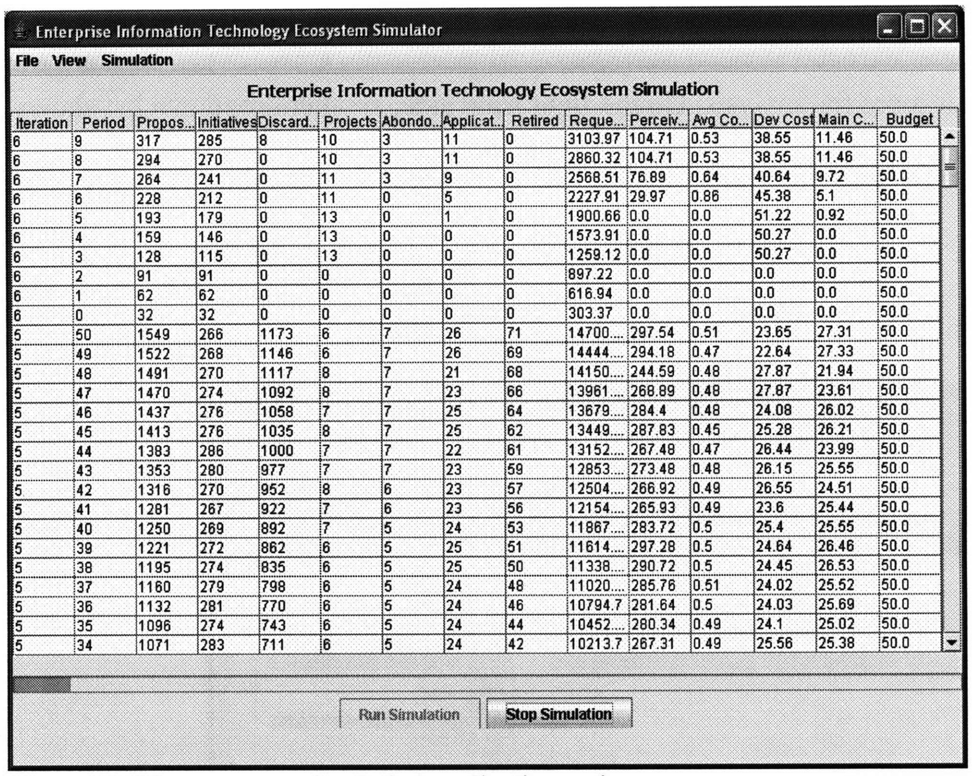
Allowed Failure Threshold: 0.4 Conforming Adjustment: 0.75 Project Abandonment Delay: 3

Figure 13: IT governance simulation application Options Regime Settings screen

However, if we set the Interval for Regime Review to 10 instead, every 10 periods of the simulation run, the user can change parameters on the Data Generation and/or Regimes tabs. This flexibility allows us to implement some more complex test scenarios.

Executing a Simulation Run

Once the simulation setting is completed, the user can run the simulation. Each period, some key simulation results for the period will be printed out on the screen for the user to review.


Enterprise Information Technology Ecosystem Simulator
File
View
Simulation
Enterprise Information Technology Ecosystem Simulation
Run Simulation
Stop Simulation

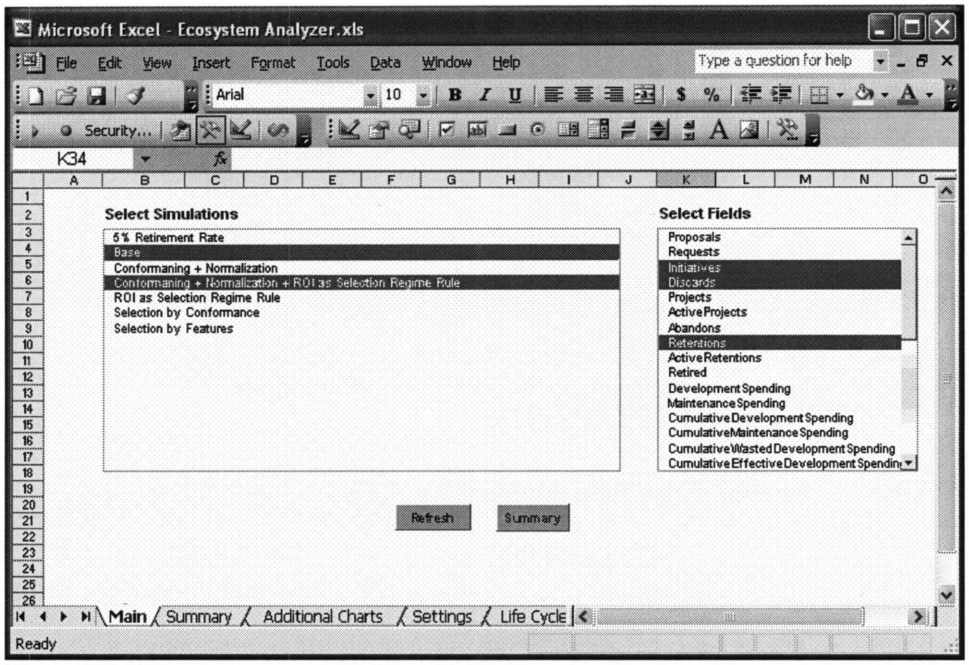
Iteration	Period	Propos...	Initiatives	Discard...	Projects	Abondo...	Applicat...	Retired	Reque...	Perceiv...	Avg Co...	Dev Cost	Main C...	Budget
6	9	317	285	8	10	3	11	0	3103.97	104.71	0.53	38.55	11.46	50.0
6	8	294	270	0	10	3	11	0	2860.32	104.71	0.53	38.55	11.46	50.0
6	7	264	241	0	11	3	9	0	2568.51	76.89	0.64	40.64	9.72	50.0
6	6	228	212	0	11	0	5	0	2227.91	29.97	0.86	45.38	5.1	50.0
6	5	193	179	0	13	0	1	0	1900.66	0.0	0.0	51.22	0.92	50.0
6	4	159	146	0	13	0	0	0	1573.91	0.0	0.0	50.27	0.0	50.0
6	3	128	115	0	13	0	0	0	1259.12	0.0	0.0	50.27	0.0	50.0
6	2	91	91	0	0	0	0	0	897.22	0.0	0.0	0.0	0.0	50.0
6	1	62	62	0	0	0	0	0	616.94	0.0	0.0	0.0	0.0	50.0
6	0	32	32	0	0	0	0	0	303.37	0.0	0.0	0.0	0.0	50.0
5	50	1549	266	1173	6	7	26	71	14700....	297.54	0.51	23.65	27.31	50.0
5	49	1522	268	1146	6	7	26	69	14444....	294.18	0.47	22.64	27.33	50.0
5	48	1491	270	1117	8	7	21	68	14150....	244.59	0.48	27.87	21.94	50.0
5	47	1470	274	1092	8	7	23	66	13961....	268.89	0.48	27.87	23.61	50.0
5	46	1437	276	1058	7	7	25	64	13679....	284.4	0.48	24.08	26.02	50.0
5	45	1413	276	1035	8	7	25	62	13449....	287.83	0.45	25.28	26.21	50.0
5	44	1383	286	1000	7	7	22	61	13152....	267.48	0.47	26.44	23.99	50.0
5	43	1353	280	977	7	7	23	59	12853....	273.48	0.48	26.15	25.55	50.0
5	42	1316	270	952	8	6	23	57	12504....	266.92	0.49	26.55	24.51	50.0
5	41	1281	267	922	7	6	23	56	12154....	265.93	0.49	23.6	25.44	50.0
5	40	1250	269	892	7	5	24	53	11867....	283.72	0.5	25.4	25.55	50.0
5	39	1221	272	862	6	5	25	51	11614....	297.28	0.5	24.64	26.46	50.0
5	38	1195	274	835	6	5	25	50	11338....	290.72	0.5	24.45	26.53	50.0
5	37	1160	279	798	6	5	24	48	11020....	285.76	0.51	24.02	25.52	50.0
5	36	1132	281	770	6	5	24	46	10794.7	281.64	0.5	24.03	25.69	50.0
5	35	1096	274	743	6	5	24	44	10452....	280.34	0.49	24.1	25.02	50.0
5	34	1071	283	711	6	5	24	42	10213.7	267.31	0.49	25.56	25.38	50.0

Figure 14: IT governance simulation application main screen

The above screen shot (Figure 14) is taken during a simulation that has 80 periods for each of the 100 iterations. It shows that the simulation has just completed the fifth iteration and is running on the sixth for period ten (results of those from period nine have just been shown on the screen). Some key indicators such as number of projects and number of perceived features are shown in the table. The blue bar at the lower portion of the screen shows the progress of the simulation. User can also terminate the current simulation by pushing the Stop Simulation button.

Analysis Tool Screens and Operations

Once the simulation data have been created in the database tables, a user can access them through a Microsoft Excel Visual Basic Application (VBA). There are two dropdown lists in the user main screen of the application (Figure 15). On the left hand side is a list showing all the simulation runs that have been completed from which a user can choose one or more for analysis. On the right hand side, another dropdown list shows all the simulation result fields from which a user can choose to see values. The user can pick one or more simulation runs from the dropdown list on the left and then select one or more fields from the list on the right before viewing any details.



Select Simulations

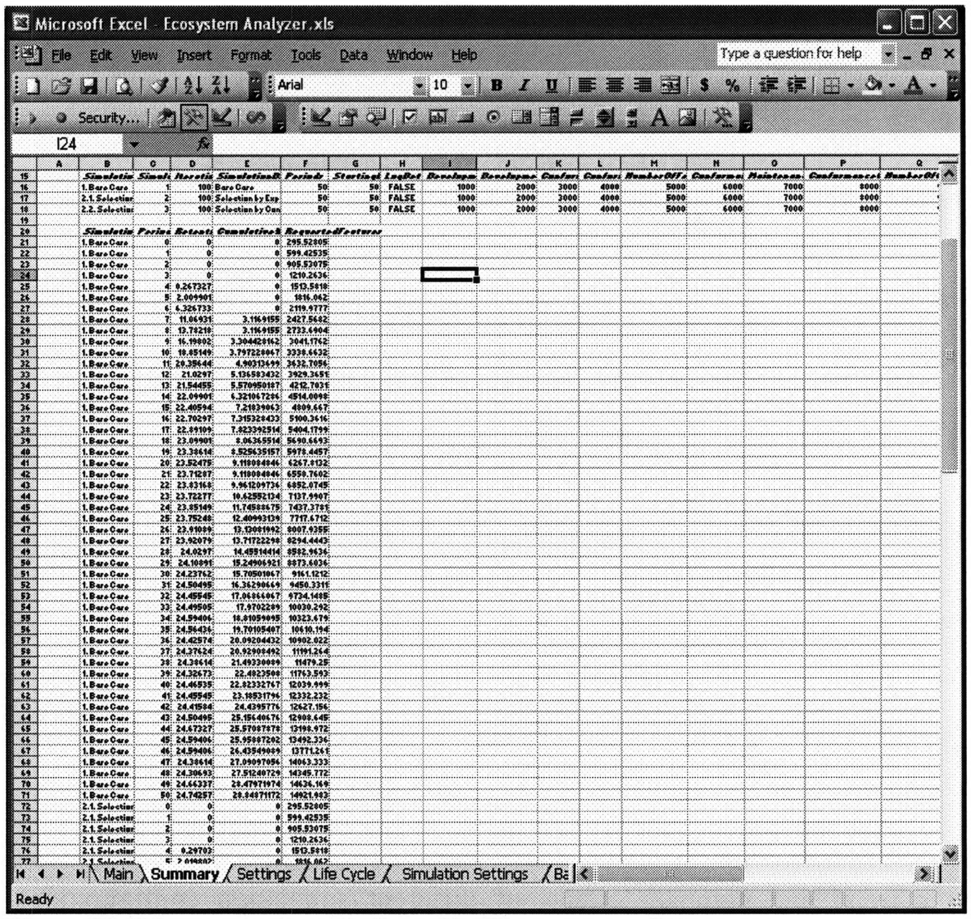
- 5% Retirement Rate
- Base
- Conformaning + Normalization
- Conformaning + Normalization + ROI as Selection Regime Rule
- ROI as Selection Regime Rule
- Selection by Conformance
- Selection by Features

Select Fields

- Proposals
- Requests
- Initiatives
- Discards
- Projects
- ActiveProjects
- Abandons
- Retentions
- ActiveRetentions
- Retired
- DevelopmentSpending
- MaintenanceSpending
- CumulativeDevelopmentSpending
- CumulativeMaintenanceSpending
- CumulativeWastedDevelopmentSpending
- CumulativeEffectiveDevelopmentSpendin

Figure 15: IT governance analysis application main screen

The Refresh button allows the user to update the two dropdown lists to show what he or she currently has in the database. The Summary button will bring up a summary screen that shows details for the selected field values for the selected simulation runs. If multiple simulation entries are selected, then all the selected fields for all of those selected simulations will be shown in the summary page shown next (Figure 16).


Microsoft Excel - Ecosystem Analyzer.xls

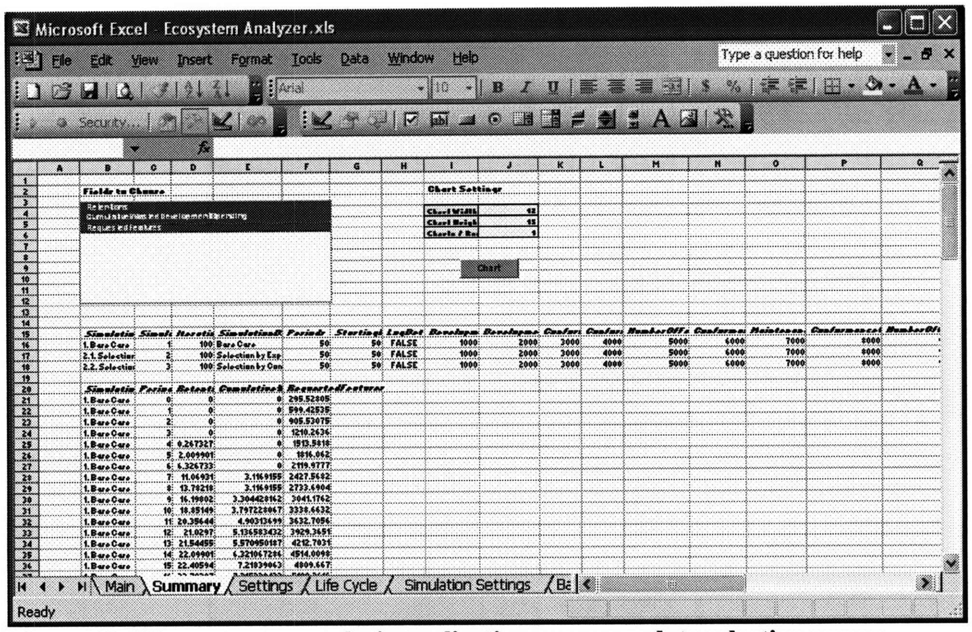
I24

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
15		*Simulatio*	*Simul*	*Iteratio*	*SimulationD*	*Periods*	*StartingI*	*LogDet*	*Developm*	*Developme*	*Confor*	*Confor*	*NumberOfFe*	*Conformai*	*Maintenan*	*Conformancet*	*NumberOf*
16		1. Base Case	1	100	Base Case	50	50	FALSE	1000	2000	3000	4000	5000	6000	7000	8000	
17		2.1. Selection	2	100	Selection by Exp	50	50	FALSE	1000	2000	3000	4000	5000	6000	7000	8000	
18		2.2. Selection	3	100	Selection by Con	50	50	FALSE	1000	2000	3000	4000	5000	6000	7000	8000	
19																	

	A	B	C	D	E	F
20		*Simulatio*	*Perioc*	*Retenti*	*CumulativeR*	*RequestedFeatures*
21		1. Base Case	0	0	0	295.52805
22		1. Base Case	1	0	0	599.42535
23		1. Base Case	2	0	0	905.53075
24		1. Base Case	3	0	0	1210.2636
25		1. Base Case	4	0.267327	0	1513.5818
26		1. Base Case	5	2.009901	0	1816.062
27		1. Base Case	6	6.326733	0	2119.9777
28		1. Base Case	7	11.06931	3.1169155	2427.5682
29		1. Base Case	8	13.78218	3.1169155	2733.6904
30		1. Base Case	9	16.19802	3.304428162	3041.1762
31		1. Base Case	10	18.85149	3.797228067	3338.6632
32		1. Base Case	11	20.35644	4.90313699	3632.7056
33		1. Base Case	12	21.0297	5.136583432	3929.3651
34		1. Base Case	13	21.54455	5.570950187	4212.7031
35		1. Base Case	14	22.09901	6.321067286	4514.0098
36		1. Base Case	15	22.40594	7.21839063	4809.667
37		1. Base Case	16	22.70297	7.315328433	5100.3616
38		1. Base Case	17	22.89109	7.823392514	5404.1799
39		1. Base Case	18	23.09901	8.06365514	5690.6693
40		1. Base Case	19	23.38614	8.525635157	5978.4457
41		1. Base Case	20	23.52475	9.118084846	6267.8132
42		1. Base Case	21	23.71287	9.118084846	6558.7602
43		1. Base Case	22	23.83168	9.961209736	6852.0745
44		1. Base Case	23	23.72277	10.62552134	7137.9907
45		1. Base Case	24	23.85149	11.74588675	7437.3781
46		1. Base Case	25	23.75248	12.40993139	7717.6712
47		1. Base Case	26	23.91089	13.13081992	8007.9355
48		1. Base Case	27	23.92079	13.71722298	8294.4443
49		1. Base Case	28	24.0297	14.45514414	8582.9636
50		1. Base Case	29	24.10891	15.24906921	8873.6036
51		1. Base Case	30	24.23762	15.70501067	9161.1212
52		1. Base Case	31	24.50495	16.36290669	9450.3311
53		1. Base Case	32	24.45545	17.06866067	9734.1485
54		1. Base Case	33	24.49505	17.9702289	10030.292
55		1. Base Case	34	24.59406	18.81059095	10323.679
56		1. Base Case	35	24.56436	19.70105407	10610.194
57		1. Base Case	36	24.42574	20.09204432	10902.022
58		1. Base Case	37	24.37624	20.92908492	11191.264
59		1. Base Case	38	24.38614	21.49330089	11479.25
60		1. Base Case	39	24.32673	22.4823508	11763.593
61		1. Base Case	40	24.46535	22.82332767	12039.999
62		1. Base Case	41	24.45545	23.18531796	12332.232
63		1. Base Case	42	24.41584	24.4395776	12627.156
64		1. Base Case	43	24.50495	25.15640676	12908.645
65		1. Base Case	44	24.67327	25.57087878	13198.972
66		1. Base Case	45	24.59406	25.95887202	13492.336
67		1. Base Case	46	24.59406	26.43549089	13771.261
68		1. Base Case	47	24.38614	27.09097056	14063.333
69		1. Base Case	48	24.30693	27.51240729	14345.772
70		1. Base Case	49	24.66337	28.47971974	14636.169
71		1. Base Case	50	24.74257	28.84871172	14921.983
72		2.1. Selection	0	0	0	295.52805
73		2.1. Selection	1	0	0	599.42535
74		2.1. Selection	2	0	0	905.53075
75		2.1. Selection	3	0	0	1210.2636
76		2.1. Selection	4	0.29703	0	1513.5818
77		2.1. Selection	5	2.019802	0	1816.062

Figure 16: IT governance analysis application summary data screen

At the summary page, the user can view major simulation settings as well as the field values for the selected simulation runs. From this summary page, the user can further select certain fields of interest and draw time series charts for easy analysis and comparison (Figure 17).


Microsoft Excel - Ecosystem Analyzer.xls
File Edit View Insert Format Tools Data Window Help
Type a question for help
Arial 10
Security...
Chart
Main Summary Settings Life Cycle Simulation Settings
Ready

Fields to Chart

Retentions
CumulativeWastedDevelopmentSpending
RequestedFeatures

Chart Settings

Chart Width	12
Chart Height	15
Charts / Row	1

Simulatio	Simula	Iteratio	SimulationD	Periods	StartingI	LogDet	Developm	Developme	Confor	Confor	NumberOfFe	Conforma	Maintenan	Conformanceat	NumberOf
1. Base Case	1	100	Base Case	50	50	FALSE	1000	2000	3000	4000	5000	6000	7000	8000	
2.1. Selection	2	100	Selection by Exp	50	50	FALSE	1000	2000	3000	4000	5000	6000	7000	8000	
2.2. Selection	3	100	Selection by Con	50	50	FALSE	1000	2000	3000	4000	5000	6000	7000	8000	

Simulatio	Period	Retenti	Cumulative	RequestedFeatures
1. Base Case	0	0	0	295.52805
1. Base Case	1	0	0	599.42535
1. Base Case	2	0	0	905.53075
1. Base Case	3	0	0	1210.2636
1. Base Case	4	0.267327	0	1513.5818
1. Base Case	5	2.009901	0	1816.062
1. Base Case	6	6.326733	0	2119.9777
1. Base Case	7	11.06931	3.1169155	2427.5682
1. Base Case	8	13.78218	3.1169155	2733.6904
1. Base Case	9	16.19802	3.304428162	3041.1762
1. Base Case	10	18.85149	3.797228067	3338.6632
1. Base Case	11	20.35644	4.90313699	3632.7056
1. Base Case	12	21.0297	5.136583432	3929.3651
1. Base Case	13	21.54455	5.570950187	4212.7031
1. Base Case	14	22.09901	6.321067286	4514.0098
1. Base Case	15	22.40594	7.21839063	4809.667

Figure 17: IT governance analysis application summary data selection screen

If multiple simulation runs had been selected from the first input screen, then same field from different simulation runs will be drawn in the same chart so that the user can easily see the trend and compare effects of different simulation configurations (Figure 18).

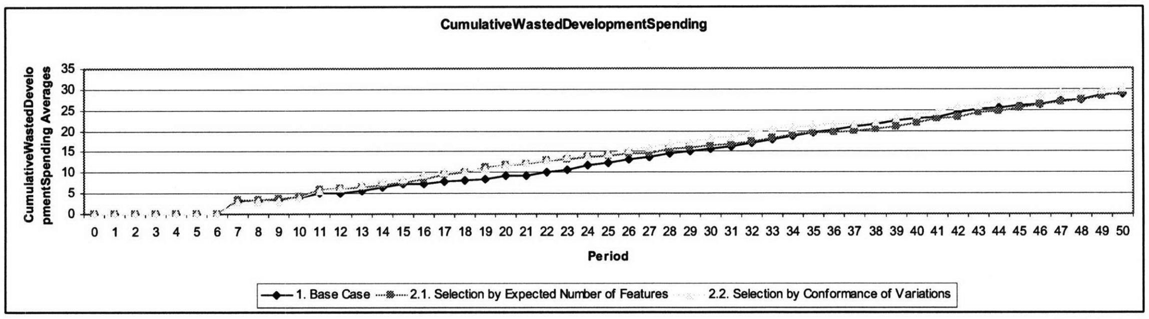


Figure 18: Sample simulation result chart – multiple runs

As shown above, lines for different simulations use different colors and different line markers to distinguish from one other.

If several fields are selected, then multiple charts will be used, one for each field (Figure 19, Figure 20 and Figure 21).

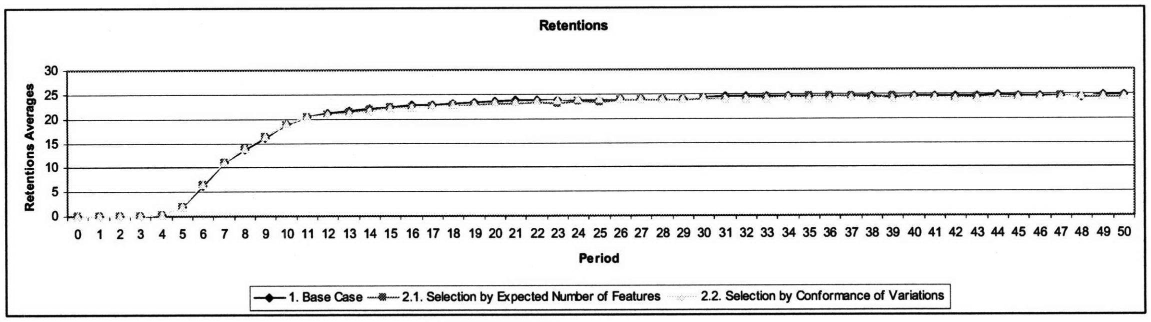


Figure 19: Sample simulation result – Retentions

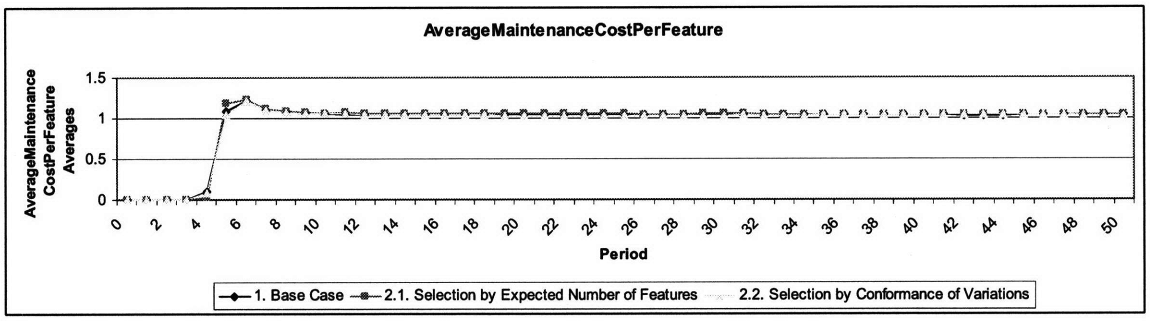


Figure 20: Sample simulation result – Average maintenance cost by feature

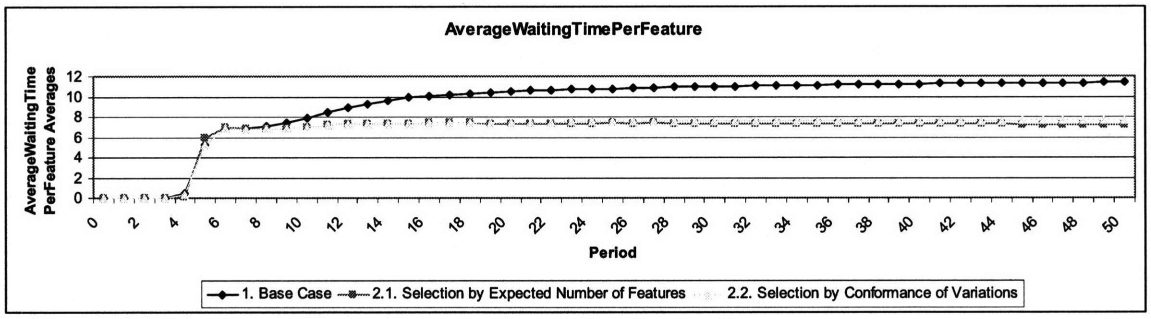


Figure 21: Sample simulation result – Average waiting time by feature

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 6: Simulation Scenarios and Analysis

This chapter details the results running some simulation scenarios. I will discuss purposes to perform simulations, the criteria to measure simulation results and some assumptions of those scenarios. Finally simulation scenarios will be presented along with comparative analysis of these scenarios.

Purposes of running simulations and IT ecosystem performance measures

The purposes of performing simulations and analysis are to research:

- How variation, selection and retention work in the dynamic IT ecosystem
- How policy regime changes affect performance of the IT ecosystem and why the specific impacts happen after the regime changes
- How IT strategies can be aligned with business strategies, serving the organization effectively and efficiently
- How IT can align its own process regimes
- How IT may benefit from some practice changes from business
- How different mechanisms, alone or in combination, hinder or help the effectiveness of the set of developed applications

In order to evaluate simulation results and compare performances of different scenarios, we first need to decide measurable criteria for comparing and evaluating simulation results. Those criteria are in two categories.

The first category measures how effectively an IT organization provides services to business units. We will look into the following measures:

- Number of Features Provided: This measure tells how many features that are provided to business units by finding the total number of features associated with all applications in the Retained Set. All else equal, the more features provided by IT, the more effective the IT organization and IT services are.
- Number of features/projects under Development, Development Spending and Percentage of Budget Spent in Development: these parameters measure how well an IT organization develops new capability. Relatively speaking, the more resources spent on developing new projects, the better IT is able to align with the business in the changing world, because a higher number of new features/projects under development and higher portion of the IT budget spent on development mean that IT has better capability to fulfill new requests from business units rather than merely spend most of its effort on maintenance items.
- Cumulative Effective Development Spending: Cumulative Effective Development Spending is defined as total Cumulative Development Spending excluding Cumulative Wasted Development Spending, which is defined as cumulative spending on projects that were never delivered. Higher value of this measure means less waste on failed projects, all other things equal.
- Total Spending: This parameter measures the total costs of ownership developing and maintaining IT asset portfolio for a period. In the fixed IT budget scenarios, all cases will probably have same or similar amount of total spending.

- Average Waiting Time to a New Feature: This parameter measures the average speed delivering a new feature by IT to business units. Shorter waiting time means more timely service from the IT organization and better client satisfaction.
- Feature Satisfaction Ratio: It is calculated by using the ratio of Number of Features Perceived over Number of Features Requested. It is another measure on the IT service level. The higher the value, the higher proportion for IT to satisfy requested features from business units.

Another category of criteria tells how effectively IT maximizes the output from the resource consumption. We will look at the following:

- Average Maintenance Cost by Feature: It is calculated by averaging the maintenance cost over number of features maintained on applications in the Retention Set.
- Average Development Cost by Feature: Similar to the average maintenance cost per feature variable, except the average development cost per feature measures against projects in the Development Set.

While there are many parameters that can be changed to perform our simulations, we are particularly interested in investigating impacts of two types of change factors related to our model.

The first are those over which an IT organization can control or upon which IT at least may have significant influences. We will focus on the selection regimes and retention regimes. In our model, several policy variables are related to the selection/retention regimes. We will focus on impacts related to how projects are chosen from initiatives and how applications retire. We assume that different selection/retention policies will lead to different system behaviors. For example, if we select projects by Return on Investment (ROI), we would expect to have relatively better user satisfaction measures for the organization in the long run. Another aspect we will look into is on normalization and forcing conforming related settings in the selection process. As we often observed, adopting conforming technologies in development help an organization maintain and operate applications more easily and at lower costs.

The second category of change factors relate to business strategies and environmental changes which IT usually can NOT directly control. Rather, these factors affect how well the IT organization operates. Westerman and Henry (2006) particularly mentioned how munificence of the environment will influence the IT processes. There are several factors in this category. The first one is the budget regime that includes budget allocation for each period as a fixed amount and budget growth pattern (Straight-line growth or growth with number of feature increases). The basic assumption on IT budget allocation is that the higher the IT budget or the higher growth rates of it, the better effectiveness of IT service, as IT can take advantage of more abundant resources to provide better and faster services, even though the efficiency measures may or may not become better. Other factors in the business and environment category are the variation pattern in terms of cost structure (average development unit cost and average maintenance unit cost by feature) and average number of features of a variation (implemented through changes in random number generator settings) for a period. As we have discussed, IT is not in the driver seat as far as the variation process is concerned.

However, that does NOT mean the IT organization is in a totally powerless position. Selection and retention regimes adopted by an IT organization could influence business in its behaviors related to variation creations. By understanding how proposals go through the IT development stages, business is encouraged or discouraged in performing some practices in the generation of variations. In this sense, IT can help business units improve through inducing them to propose variations that have higher quality relative to the ecosystem. I assume the better the quality of proposals from the variation process, the better results the IT organization can achieve. The main simulation scenarios that I will test in this section are summarized in Table 2.

Category	Regime	Measure
IT can control or have great influence	Selection Regime	ROI Normalization + Forcing Conforming
	Retirement Regime	Retire 5% for each period
IT may have some influence or IT can NOT control	Budget	Fixed amount allocation Increase at a fixed percentage Increase with feature growth
	Variation	Number of features each project Conformance values of proposals

Table 2: Simulation scenarios summary

Simulation Scenario 1: Set up and Run the Base Simulation Scenario

The main parameters for the baseline scenario are highlighted as follows:

- Budget growth: uses fixed budget for all the periods.
- Selection regime: chooses new projects from initiatives by times they were proposed, first come first served. Normalization and forcing conforming rules are NOT activated.
- Retention regime: disables retirement of implemented applications.
- Feature decays: sets requested feature decay to start from period five (5) with a decay rate at 5% on Perceived Features and a 10% on Requested Features per period respectively.

The following two lists of parameters detail simulation settings for the base scenario. The first is for the simulation configuration and the parameter values set up here will stay the same for a simulation run (Table 3).

Variable	Sample Value	Explanation
SimulationName	Base	Name of the simulation
Iterations	100	Number of runs for the simulation
SimulationDescription	Base Line Test	Description of the simulation
Periods	80	Periods for one run
StartingBudget	30	Budget allocation for the starting period
LogDetails	N	Whether log intensively while running the simulation
DevelopmentUnitCostSeed	1000	Random seed for Development Cost
DevelopmentDurationSeed	2000	Random seed for Development Duration
ConformanceOnDevelopmentCostSeed	3000	Random seed for Conformance Impact on Development Cost
ConformanceSeed	4000	Random seed for Conformance
NumberOfFeaturesSeed	5000	Random seed for Number of Features
ConformanceOnFeaturesSeed	6000	Random seed for Conformance Impact on Number of Features
MaintenanceUnitCostSeed	7000	Random seed for Maintenance Cost
ConformanceOnMaintenanceUnitCostSeed	8000	Random seed Conformance Impact on Maintenance Cost
NumberOfOpportunitiesEachPeriodSeed	9000	Random seed for Number Opportunities for each period
FailureProbabilitySeed	10000	Random seed for Failure Probability

Table 3: Simulation settings for the base case

The second set of parameters control the settings for one simulation period and can be changed during a simulation execution if the user chooses to do so. Once changed, the modified values will be used for the remaining periods of the simulation runs, unless they are changed again by the user.

Category	Variable	Specification	Value	Explanation
Simulation Setting	**ReviewInterval**	Interval of periods the application allow the user to change simulation settings	81	Application allows the user to change period setting after every 81 periods in execution
Budget	**StraightlineBudgeting**	Whether to use straight line budget	N	Use Straight-line budget
	BudgetStraightlineChangePortion	Percentage of growth on budget	0	Straight-line budget increase at 0%
	LowerBudgetLimit	Lower bound of budget consumption allowed in percentage value	98	Lower budget consumption is 98%
	UpperBudgetLimit	Upper bound of budget consumption allowed in percentage value	102	Upper budget consumption is 98%
	GrowingBudgetWithFeatures	Whether to grow budget by feature growth	N	Do not grow budget by features delivered
Conforming	**NormalizingVariationsByConformance**	Normalization of variations	N	Do not normalize variations
	ConformanceOnFailure	Adjustment of conforming value on a project failure possibility	0.5	Failure probability increases 0.5 units for every unit a project is less below the Conforming Threshold. For a project 10% below the threshold, its failure probability value increases 5%
	ConformingThreshold	Minimum conforming value of a variation to have to be qualified as a project candidate when forcing conforming in selection process is applied	0.6	Conforming Threshold value is 0.5
Variation	**ConformanceMean**	Mean value used to generate Conformance random values for variations	0.5	Mean value 0.5 is used to generate Conformance random values for variations
	ConformanceStd	Std value used to generate Conformance random values for variations	0.2	Std value 0.2 is used to generate Conformance random values for variations
	DevelopmentCostMean	Mean value used to generate Development Cost random values for variations	3	Mean value 4 is used to generate Development Cost random values for variations
	DevelopmentCostStd	Std value used to generate Development Cost random values for variations	0.8	Std value 1 is used to generate Development Cost random values for variations

	DevelopmentDurationMean	Mean value used to generate Development Duration random values for variations	4	Mean value used to generate Development Duration random values for variations
	DevelopmentDurationStd	Std value used to generate Development Duration random values for variations	1	Std value used to generate Development Duration random values for variations
	MaintenanceCostMean	Mean value used to generate Maintenance Cost random values for variations	0.6	Mean value used to generate Maintenance Cost random values for variations
	MaintenanceCostStd	Std value used to generate Maintenance Cost random values for variations	0.1	Std value used to generate Maintenance Cost random values for variations
	NumberOfFeaturesMean	Mean value used to generate Number of Features random values for variations	10	Mean value used to generate Number of Features random values for variations
	NumberOfFeaturesStd	Std value used to generate Number of Features random values for variations	4	Std value used to generate Number of Features random values for variations
	NumberOfOpportunitiesMean	Mean value used to generate Number of Opportunities random values for variations	30	Mean value used to generate Number of Opportunities random values for variations
	NumberOfOpportunitiesStd	Std value used to generate Number of Opportunities random values for variations	5	Std value used to generate Number of Opportunities random values for variations
	FailureProbabilityMean	Mean value used to generate Failure Probability random values for variations	0.3	Mean value used to generate Failure Probability random values for variations
	FailureProbabilityStd	Std value used to generate Failure Probability random values for variations	0.1	Std value used to generate Failure Probability random values for variations
	ConformanceImpactOnDevelopmentCostMean	Mean value used to generate Conformance Impact on Development Cost random values for variations	1	1 is used as mean value when generate Conformance Impact on Development Cost random numbers
	ConformanceImpactOnDevelopmentCostStd	Std value used to generate Conformance Impact on Development Cost random values for variations	0.2	0.2 is used as std value when generate Conformance Impact on Development Cost random numbers
	ConformanceImpactOnFeaturesMean	Mean value used to generate Conformance Impact on Number of Features random values for variations	1	1 is used as mean value when generate Number of Features random numbers

	ConformanceImpactOnFeatureStd	Std value used to generate Conformance Impact on Number of Features random values for variations	0.2	0.2 is used as std value when generate Number of Features random numbers
	ConformanceImpactOnMaintenanceCostMean	Mean value used to generate Maintenance Cost random values for variations	1	1 is used as mean value when generate Conformance Impact on Maintenance Cost random numbers
	ConformanceImpactOnMaintenanceCostStd	Std value used to generate Maintenance Cost random values for variations	0.2	0.2 is used as std value when generate Conformance Impact on Maintenance Cost random numbers
Retention	**RetiringByDeployedTime**	Whether to retire applications in the order of deployment time. If Y, applications deployed earliest will retire first	N	N means deployment time of an application is not a factor in determining the priority retiring the application
	RetriingMostExpensive	Whether to first retire the most expensive applications from the retained set	N	N means maintenance cost of an application is not a factor in determining the priority retiring the application
	RetiringByFeatures	Whether to retire applications that have least number of features first	N	N means number of features of an application is not a factor in determining the priority retiring the application
	RetiringByConformance	Whether to retire applications that have lowest conformance values first	N	N means conformance value of an application is not a factor in determining the priority retiring the application
	RetiringByROI	Whether to retire applications by ROI in descending order. The ones that have lowest RIO will retire first	N	N means ROI of an application is not a factor in determining the priority retiring the application
	RetiringPortionOfMaintenanceCost	Retire applications to free at least to portion of maintenance spending	0	No reclaim of maintenance spending
	MinRetentionPeriods	The minimum number of periods that an application will be kept	3	An application will be kept for at least three periods
Selection	**SelectingByProposedTime**	Whether to select projects by the time they were proposed	Y	Select projects by the time proposed
	SelectingByFeatures	Whether to select projects by their expected number of features	N	N means number of expected features of a variation is not a factor in determining the priority to make it a project
	SelectingByROI	Whether to select projects by their ROI. The higher the RIO of a variation, the earlier it will be selected	N	N means RIO of a variation is not a factor in determining the priority to make it a project

	SelectingByConformance	Whether to select projects by conformance value. The higher ones will have higher priority	N	N means conformance of a variation is not a factor in determining the priority to make it a project
	DelayInReviewingInitiatives	Minimum number of periods that an initiative stays in the Initiative Set before being considering as a project candidate	2	An Initiative will stay in the Initiative Set for at least two periods before being considered as a project candidate
	DelayInDiscardingInitiatives	Minimum number of periods that an initiative stays in the Initiative Set before being discarded	5	A variation will not be discarded for at least five periods after it is in the Initiative Set
	DelayInAbondoningProjects	Minimum number of periods that a project stays in the Development Set before being considered for failure check	3	A project will not be abandoned for at least three periods after it is in the Development Set
	MaxAllowedFailurePossibility	The maximum allowed failure possibility for a project to be deemed as a failure	0.4	A project will be considered a failure if failure probably is more than 0.4
	ForcingConforming	Whether to select only conforming initiatives as projects	N	N means there is no required minimum for a variation to be considered as a project candidate
Feature Decay	DecayStartPeriod	When feature decay starts after a proposal is requested or after an application is deployed	5	Feature decay starts after five periods
	PerceivedFeatureDecayRate	Percentage of decay on number of perceived features	5	Each period, decay 5% of number of perceived features
	RequestedFeatureDecayRate	Percentage of decay on number of requested features	10	Each period, decay 10% of number of requested features

Table 4: Period settings for the base case

The parameters highlighted in bold fonts in the two tables are the ones against which we will test impacts of changes in this section on some scenarios. However, in this base simulation scenario, we do not change any of the period settings.

In this base simulation scenario, we see three distinct phases in terms of how IT budget was spent. In the first several periods, 100% of the IT budget was allocated to development of new projects because there were NO finished applications to maintain yet. The number of projects under development was initially constrained by the total IT budget allocated for the period and the value of the Delay in Reviewing Initiatives parameter only. The second phase is characterized by the fact that the IT budget was split between the development spending and the maintenance spending, with the development spending percentage going lower and the maintenance spending percentage going higher as retention of already-deployed applications used some resources that otherwise could go to development. The third phase had all IT budget spent on maintenance with no resources left at all for any project development.

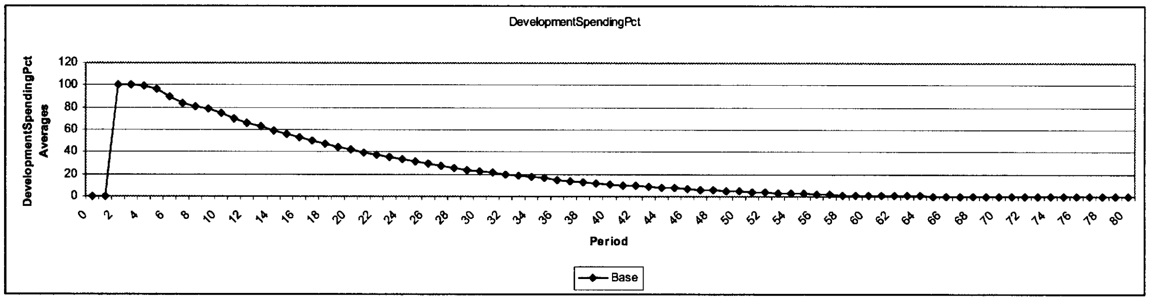


Figure 22: Base case – Development spending percentage

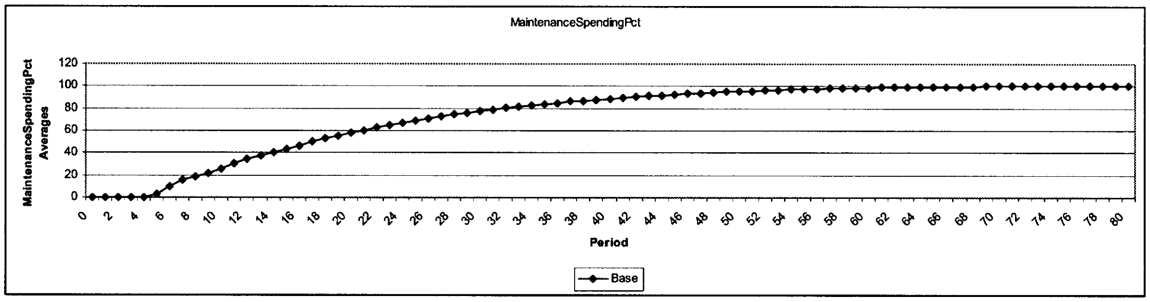


Figure 23: Base case – Maintenance spending percentage

Corresponding to the budget spending pattern, the number of projects jumped up quickly in the first stage, then continuously went lower and finally arrived at a zero level while the number of retained applications gradually went up all the way, before saturating the whole IT budget at around period 60.

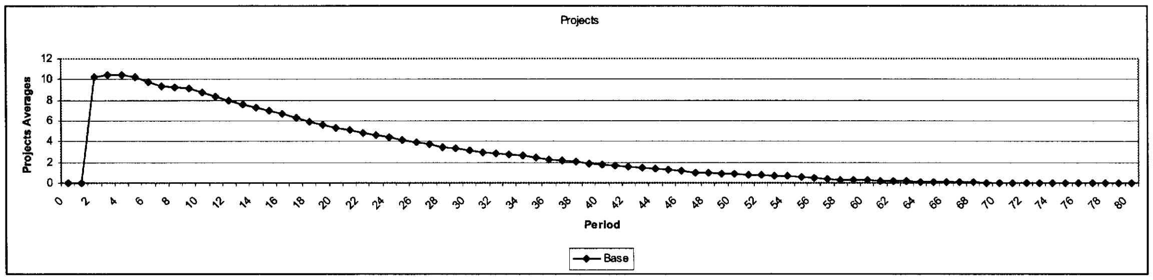


Figure 24: Base case – Projects

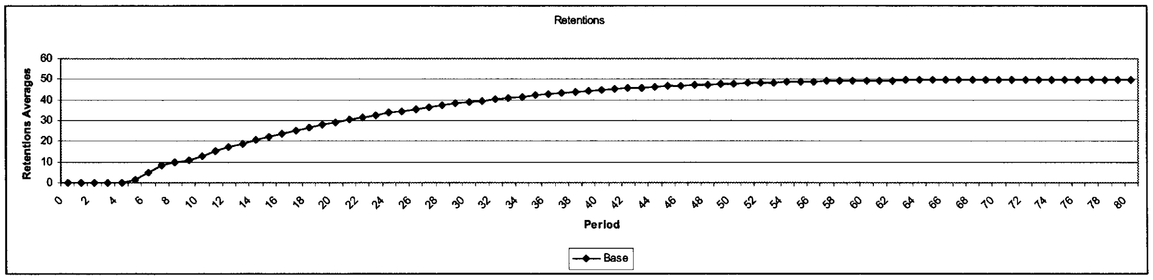


Figure 25: Base case – Retentions

In addition, although the Initiative Set increased its number of initiatives in the first phase, it quickly arrived at a certain level and stayed relatively constant thereafter. In the same time, the number of discarded initiatives kept going up. The level of Discarded Set stock variable seems to be growing in a linear fashion in this particular simulation setting, meaning that the change (increase) rate is relatively constant over the periods. The reason to have such patterns on the Initiative Set and the Discarded Set is due to the policy we used on discarding initiatives. Recall our policy is to discard initiatives if they have stayed in the Initiative Set longer than the value of Delay in Discarding Initiative parameter. Under this setting, if a proposal could not become a project in certain periods after it was requested by business, it would go into the Discarded Set. The Discarded Set in the model is a stock variable that only has inflow without any outflow such that its value will keep increasing, as we do NOT re-activate any discarded initiatives and get them back into the Initiative Set.

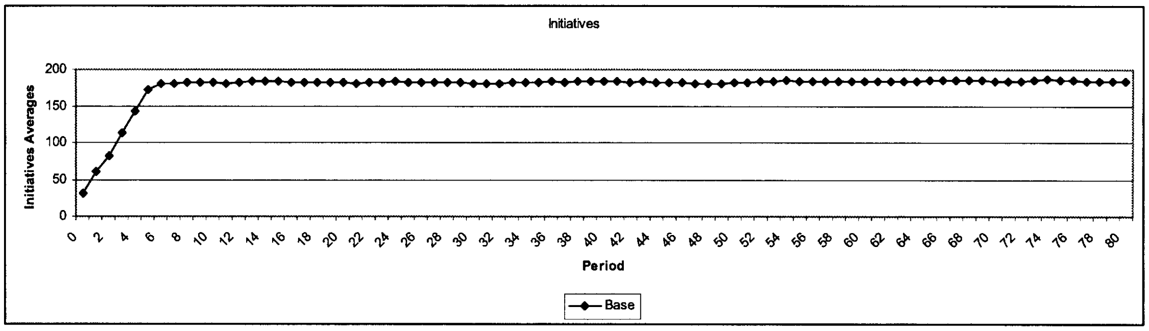


Figure 26: Base case – Initiatives

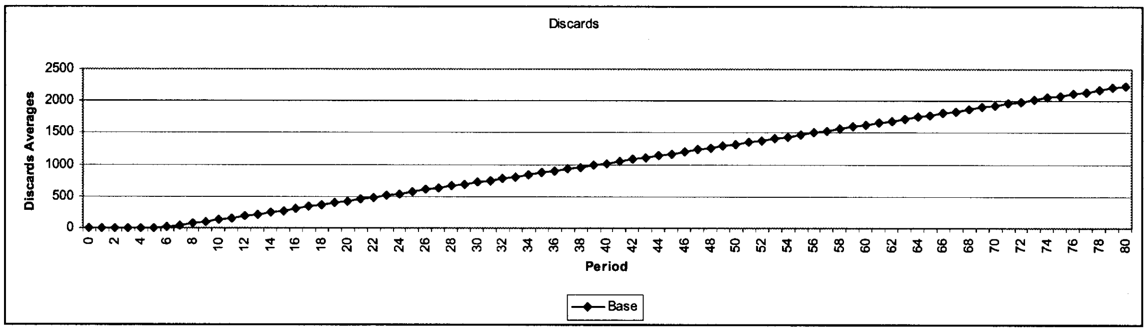


Figure 27: Base case – Discards

Number of applications in the Retained Set, after moving up in the first 60 periods, stopped increasing because there was no further development on projects after that period. In the mean time, because we did not retire any existing applications from the Retained Set, the number of applications was not decreasing either.

We can explain what happened by reviewing the IT ecosystem model and the simulation settings we had discussed for this base scenario in previous sections. To briefly recap, in this base simulation, we set the budget allocation at a fixed value (Straight-line increased at the rate of 0% for each period), so the budget pressure (which can be viewed as total IT spending over total IT budget) in the two balancing loops Process Selection (B3) and Process Retention (B4) are determined by the total amount of the development spending and the maintenance spending. In the first phase, because there had been neither development projects nor retained applications in the system, no budget pressure existed. In such an environment, we quickly see new development projects move into the Development Set right after the simulation ran after the period specified by the value of the Delay in Reviewing Initiatives parameter. However, once there were projects in the Development Set, the development spending associated with them would increase the budget pressure. In the later period of the simulation, some projects would fail and move into the Abandoned Set while others would finish their development work and go into the Retained Set as applications, which would also consume the IT maintenance budget. The higher the number of projects in the Development Set, the more outflows into the Retained Set and into the Abandoned Set. This explains why the number of projects in the Development Set kept going down in the second phase, as more IT budget went to the maintenance allocation bucket. Because finished projects went into the Retained Set, the number of applications kept going up in this phase. Since we chose, in this base case, not to retire applications and to give maintenance spending the highest priority in IT budget allocation, eventually, all budget went into the maintenance work with no resources left for development of unfinished projects or selection of any new projects in the third phase.

In terms of the number of perceived features provided by IT, we had the value initially went up, but only to see it go lower quickly. There are two factors determining changes in the number of features perceived or provided. The first is the features associated with applications, especially with new applications going into the Retained Set as feature inflow. The other is the decay outflow on features from those old applications – representing the fact that older functionality tends to be seen as less useful in most organizations over time. In the earlier periods, the inflow was dominant because few applications were old. However, as time went by, the Retained Set was stuck with legacy applications that would decay features each period, leading to an overall decline of perceived features in the Retained Set, especially when there was no retirement allowed to conserve budget for any new feature development.

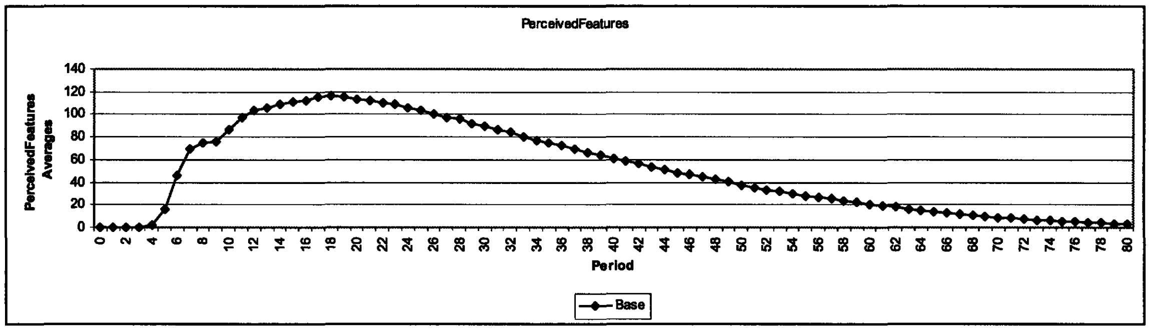


Figure 28: Base case – Perceived features

Even though we had a decay rate on requested features that was twice as much as that on the perceived features in this base case simulation, it is no surprise that the Feature Satisfaction Rate would still go lower, because in most of the periods, outstanding features requested were many time more than perceived features provided by active applications. In this sense, decay in requested features failed to mitigate the declining trend of the Feature Satisfaction Ratio. If we look at the simulation data, we can see even at the peak of the development, maintained features were still less than 10% of what had been requested.

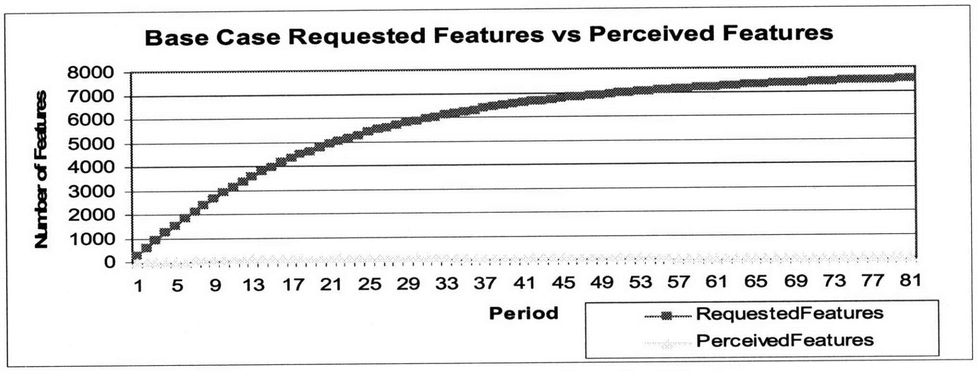


Figure 29: Base case – Requested features versus Perceived features

Does this mean IT should make it tougher for business to request new features such that the Features Satisfaction Ratio can be improved? I performed another simulation where I reduced the average number of variations requested from business units per period by half. However, I did not see significant improvement in the value of the Feature Satisfaction Ratio.

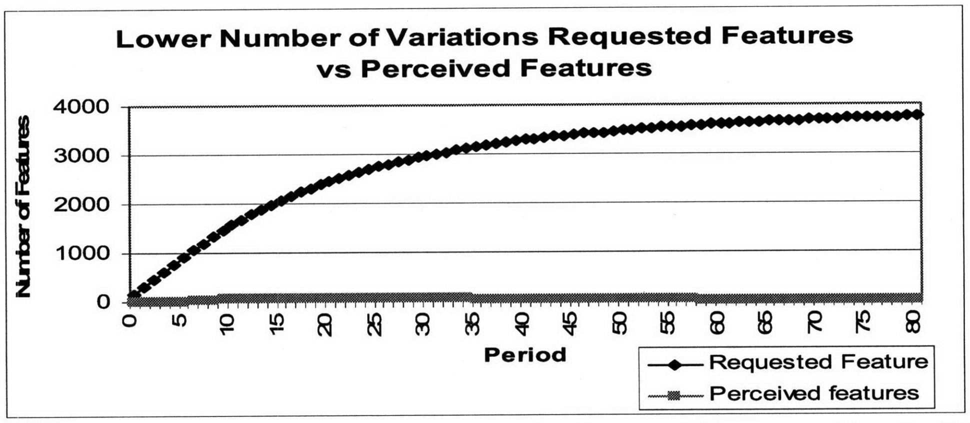


Figure 30: Base case – Lower Number of variation Requested Features vs Perceived Features

This is because we did not have new applications going into the Retained Set when maintenance used up the whole IT budget such that the decay of the perceived features quickly made the Number of Perceived Features very small. Even we had lower number of requested features; still the Feature Satisfaction Ratio would not improve. Suppressing user demand in such a scenario (no development on new projects) is probably not going to help improve IT service quality.

This basically reflected the fact that due to limited resources, IT probably will not be able to fulfill high portion of features that business would like to have.

The base case sends us a clear message: If the IT organization does NOT do its work optimizing its process, it will NOT be able to provide enough features to satisfy the business needs. More important, the Feature Satisfaction Ratio would be very low and the long term trend is to go even lower.

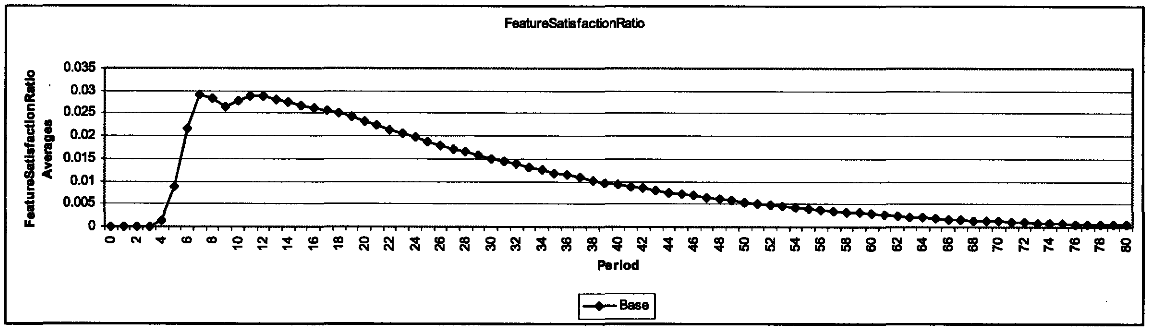


Figure 31: Base case – Feature satisfaction ratio

Simulation Scenario 2: Use Return on Investment (ROI) as the Selection Regime Rule

What if in the base case, rather than selecting new projects merely by the time they were proposed, we had used a "smarter" way to choose new projects. In this new simulation case, I adopt one of the most popular selection regime rules that many organizations use in making IT investment decisions: selecting new projects by expected ROI. To simplify our measure, I use the ratio between Working Number of Features and Working Maintenance Cost to calculate the value for ROI. The reason to use this ratio to measure ROI is that most of the IT application life cycle costs are maintenance cost. Recall we mentioned in our literature review section that the portion of maintenance costs normally counts 60%-80% of system life-cycle costs for complex systems (Banker and Slaughter 1997, Schneidewind 1987). By using this selection regime, I expect to see better selected Development Set and Retained Set.

Changing only the selection regime to use ROI does not seem to have changed the behavior of the ecosystem though. We still see similar picture in IT budget allocations, development projects, and the declining values in perceived features and feature satisfaction ratio as what we saw in the base case.

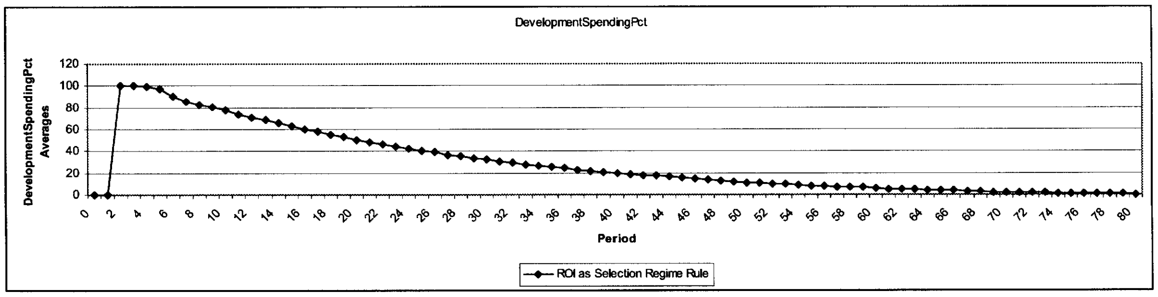


Figure 32: ROI selection regime – Development spending percentage

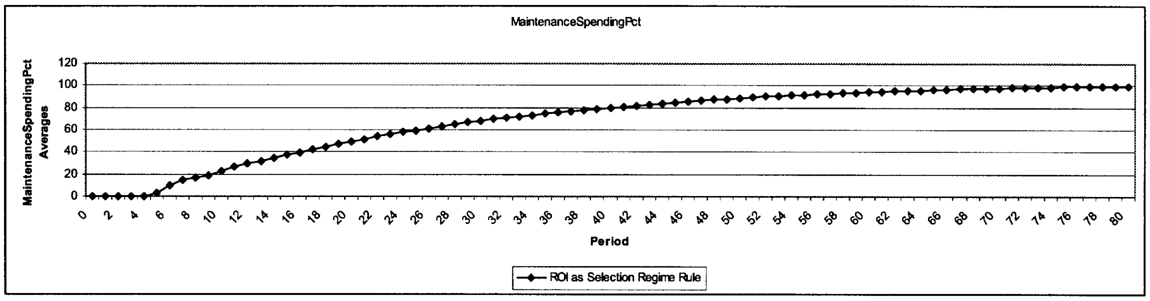


Figure 33: ROI selection regime – Maintenance spending percentage

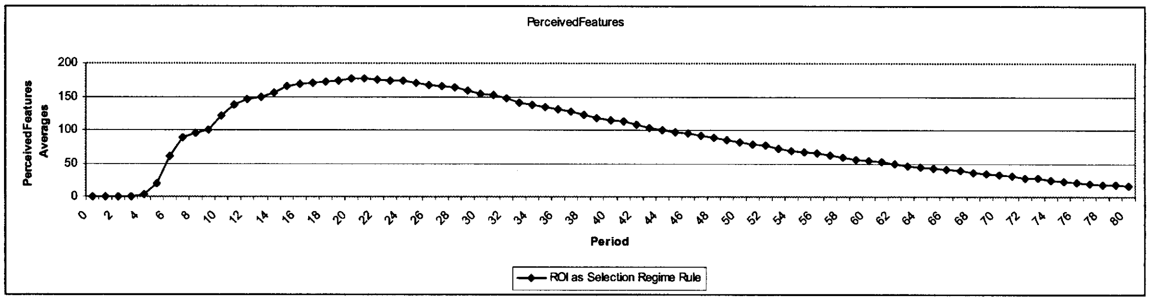


Figure 34: ROI selection regime – Perceived features

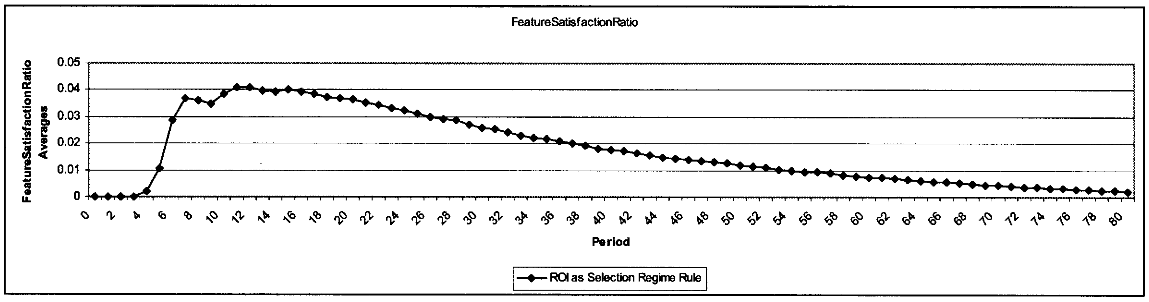


Figure 35: ROI selection regime – Feature satisfaction ratio

Does it mean that this selection regime using RIO will NOT make any difference? In order to find it out, I performed comparison with the base case. From now on, I will usually include the value of the same parameter of the base case in the same drawing chart of new simulation cases so that we can see more insights through the comparison.

Relatively speaking, selecting projects by ROI allowed the IT organization to do slightly more projects, spend slightly higher percentage of its budget on development and maintain slightly more applications. However, it resulted in much more perceived features, shorter waiting time for new features, and better Feature Satisfaction Ratio (although it still had a declining trend).

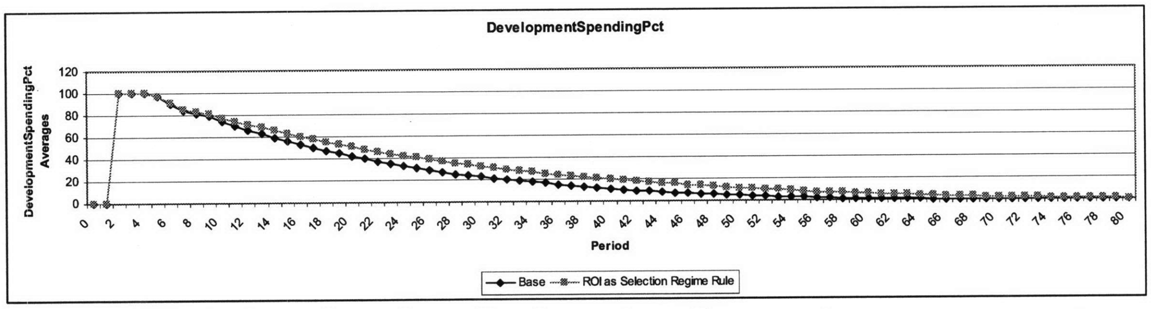


Figure 36: ROI selection regime vs base – Development spending percentage

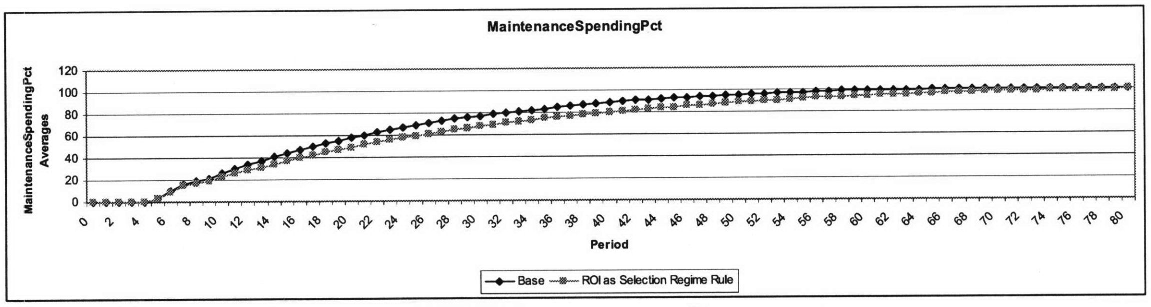


Figure 37: ROI selection regime vs base – Maintenance spending percentage

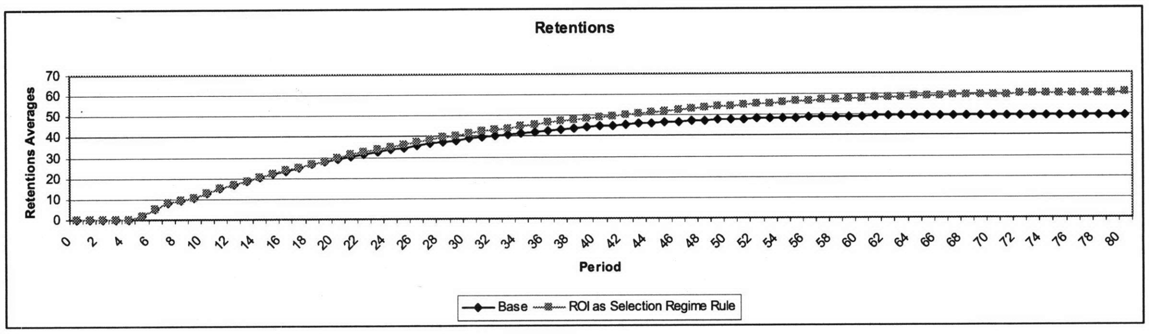


Figure 38: ROI selection regime vs base – Retentions

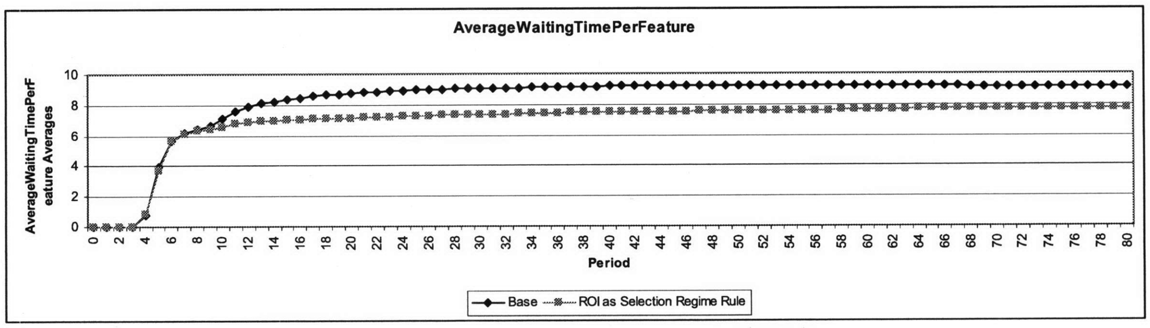


Figure 39: ROI selection regime vs base – Average waiting time by feature

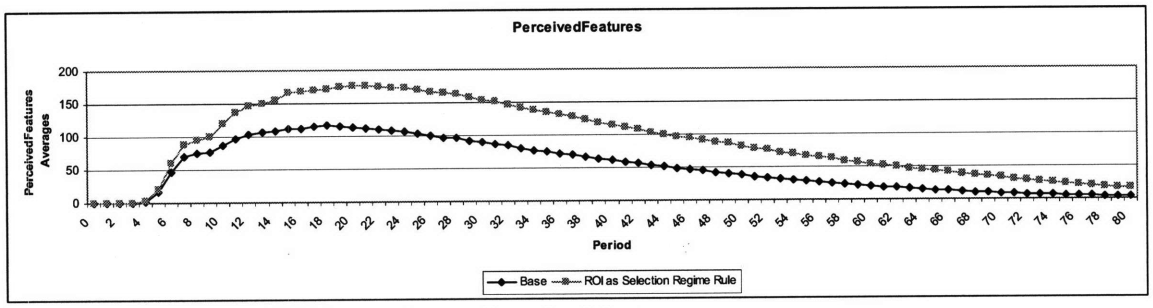


Figure 40: ROI selection regime vs base – Perceived features

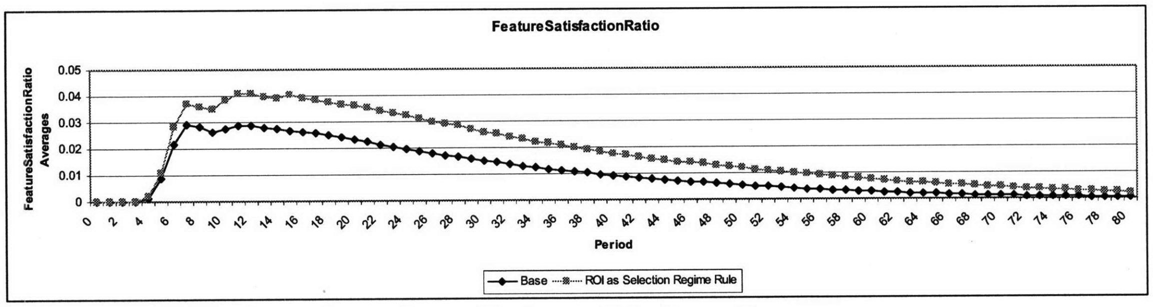


Figure 41: ROI selection regime vs base – Feature satisfaction ratio

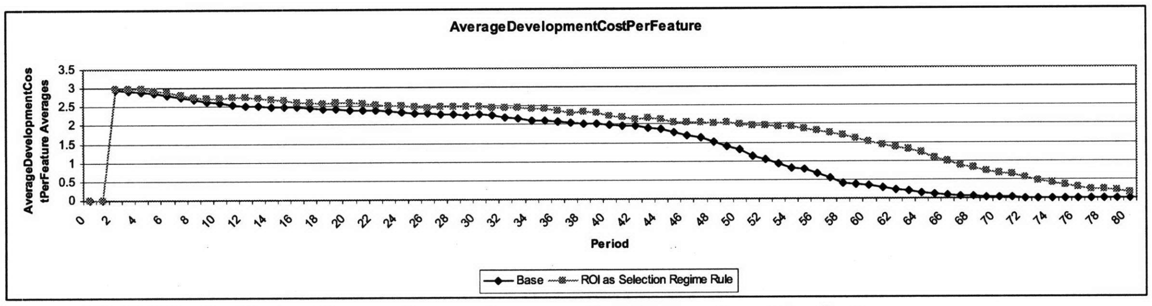


Figure 42: ROI selection regime vs base – Average development cost by feature

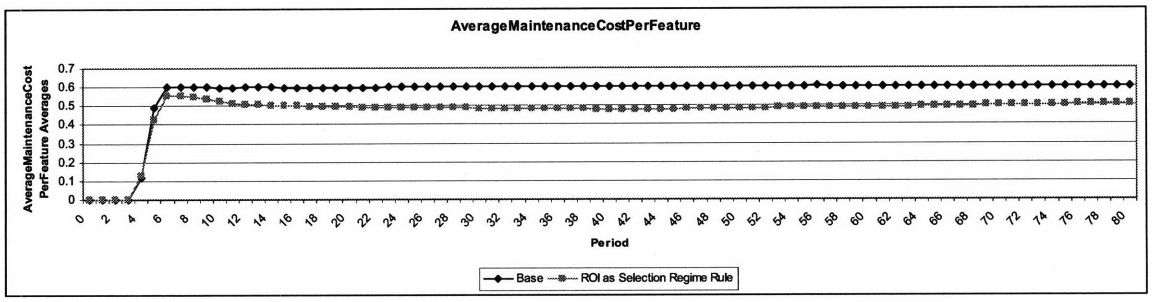


Figure 43: ROI selection regime vs base – Average maintenance cost by feature

The most distinctive character using ROI as the selection regime was that both the number of development projects and the number of retained applications were slightly higher than their counterparts in the base case, even with the same amount of budget allocation and the same budget growth regimes (in this case, constant budget allocation for all periods). This did not explain directly why we achieve better results. The explanation actually lies in the fact that because we now have more efficient Retained Set and Development Set in terms of average cost developing or maintaining a feature, the same amount of IT budget allocation will allow us to both maintain more existing features and develop more new features at the same time than otherwise using the base case settings. Another interesting finding is that the Number of Perceived Features was always higher than that of the base case for the same period in the simulations. In other words, ROI selection regime allows us to have better Number of Features from the very beginning and all the way through.

Unlike what we understand through our experiences, where we normally have to make trade-offs between efficiency and effectiveness or between the development spending and the maintenance spending, using ROI as the selection regime seems to help improve both development and maintenance to achieve a overall better ecosystem. It is encouraging that through our understanding of the IT ecosystem and use of appropriate IT regime policies, IT managers indeed could do better on both efficiency and effectiveness measures. This also justifies why many IT organizations use the ROI as the main selection regime to help determine how to fund IT projects.

Simulation Scenario 3: Introduce Normalization and Forcing Conforming

We have discussed, in prior sections, how conformance of a variation may affect its development costs, its maintenance costs, and its failure probability in development. Literatures (for example, Broadbent and Weill, 1999) and our own observations tell us that adjusting variations to make them more conforming to standard technologies will reduce its maintenance costs (because maintenance conforming technologies will be cheaper and easier) but may also reduce its features provided (we will have to forfeit some exotic function that only the non-conforming technologies can provide). While this observation may be true looking at individual project level, we need to see whether it holds well as a whole in a dynamic IT ecosystem.

To address this concern, I ran a new simulation that had the same settings as those in our base scenario except requiring normalization on requested variations and forcing conformance in selecting new projects.

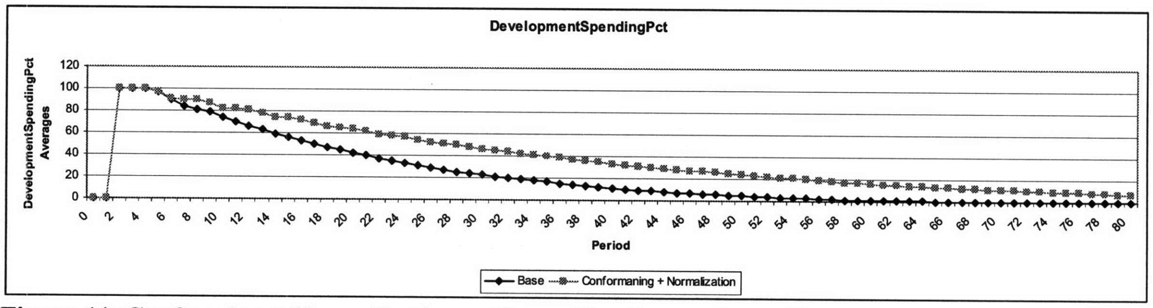


Figure 44: Conforming + Normalization vs Base – Development spending percentage

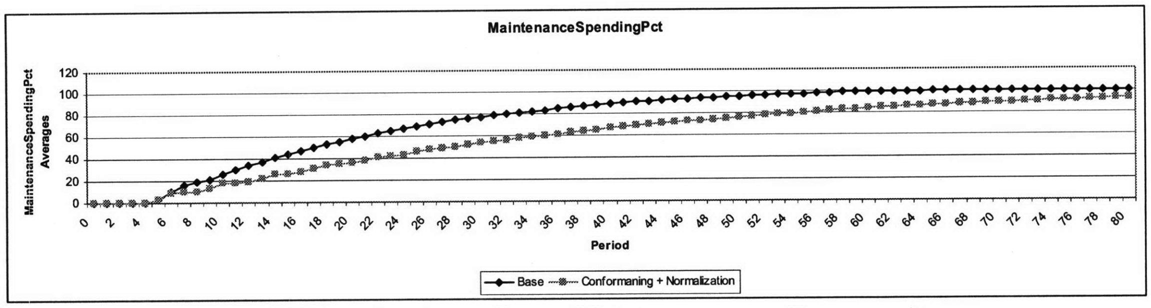


Figure 45: Conforming + Normalization vs Base – Maintenance spending percentage

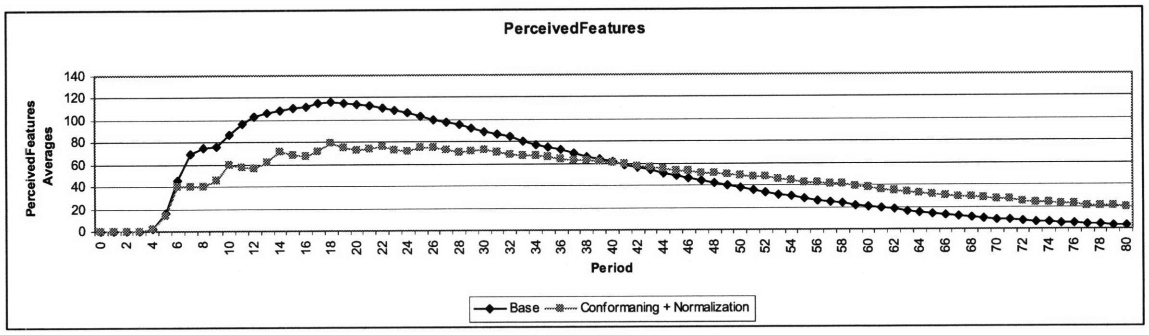


Figure 46: Conforming + Normalization vs Base – Perceived features

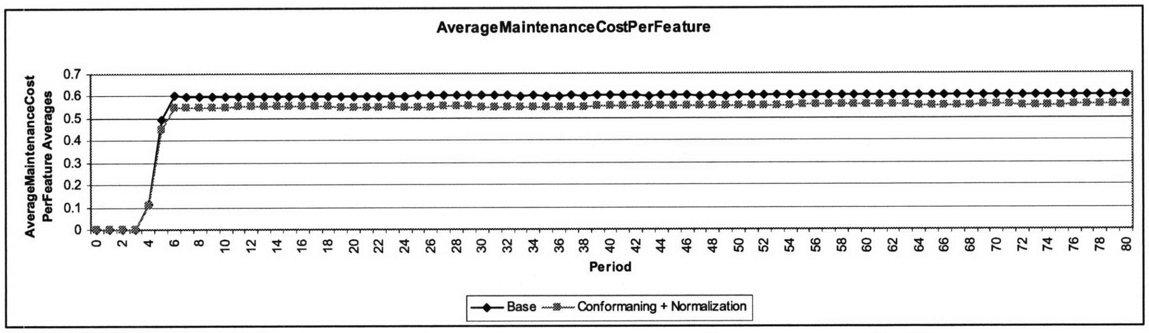


Figure 47: Conforming + Normalization vs Base – Average maintenance cost by feature

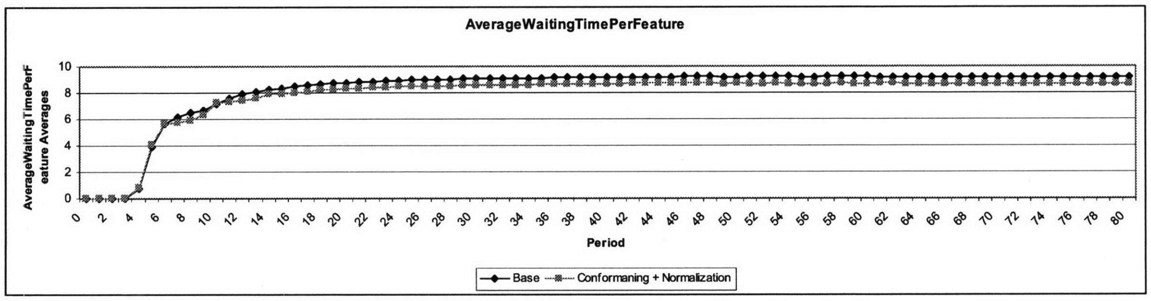


Figure 48: Conforming + Normalization vs Base – Average waiting time by feature

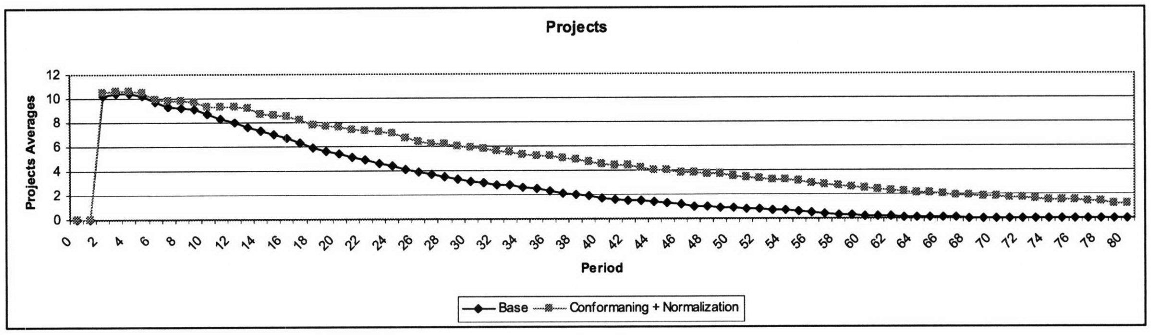


Figure 49: Conforming + Normalization vs Base – Projects

While this scenario did verify my expectation of lower overall maintenance costs than what we observed in the base case because the reduction of the maintenance costs on those originally non-conforming variations during the normalization process, it was some how counter-intuitive on the values of Number of Features Provided compared with the base case. In our simulation design, because we reduced the number of features while we make a non-conforming variation conforming through the normalization process as well, the number of features should be lower too as I initially thought. So what I had expected for this scenario should be lower maintenance costs and fewer features for the Retained Set in the IT ecosystem. What I actually observed was that before period 40, my prediction seemed to be correct; however, after the two lines on Numbers of Perceived Features (one for the base scenario and the other for this new scenario) crossed from that point, the new scenario had higher number of features. It seems that by adopting normalization and conforming in the selection process, we could have a slightly more efficient Retained Set in terms of average maintenance cost by feature (as I expected). At the same time, even though we had lower number of features in the earlier periods, we might eventually have more features in the long run (a little surprise to me). It seems that the benefit of adopting conforming technologies is not only a more efficient Retained Set, which affords us a bigger Retained Set in the long run, but also an increase in number of development projects because of more development budget coming from the saving of the maintenance efforts. In terms of number of features perceived, we see a "worse before better" scenario.

Simulation Scenario 4: Add Retirement into the Picture

In our base case, there is no retirement of any old applications from the Retained Set. In practice, every IT organization does retire out-of-date applications from the Retained Set, even few organizations explicitly force such practice. In some cases, however, some organizations will make retirement a policy. For example, they will increase the status of new requests that require retire one or more legacy applications to obtain funding for development of the new requests. Although these policies are seen as useful, the policymakers may not always fully aware of what gains or losses are associated with this policy. To find out what will happen if we introduce application retirement into the IT ecosystem, I performed a new simulation run with a 5% reduction rate for each period on maintenance cost through retirement of existing applications from the Retained Set. In this test, I retire applications by the time they were deployed. The earlier an application was delivered, the earlier for it to retire, while keeping other parameters the same as they were in the base scenario.

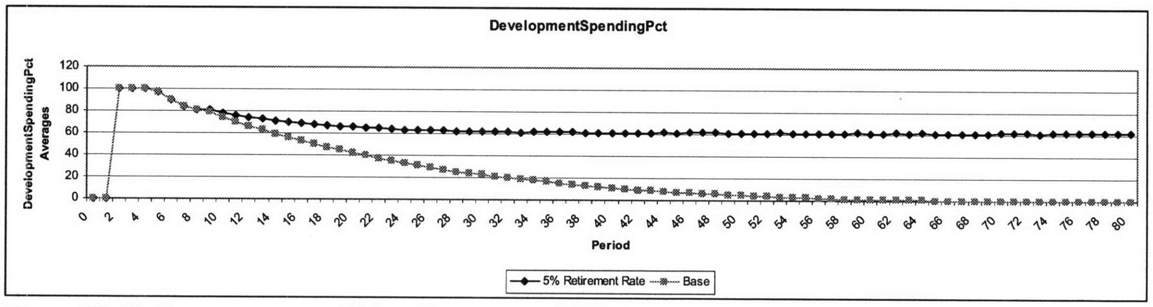


Figure 50: Retirement rate at 5% vs Base – Development spending percentage

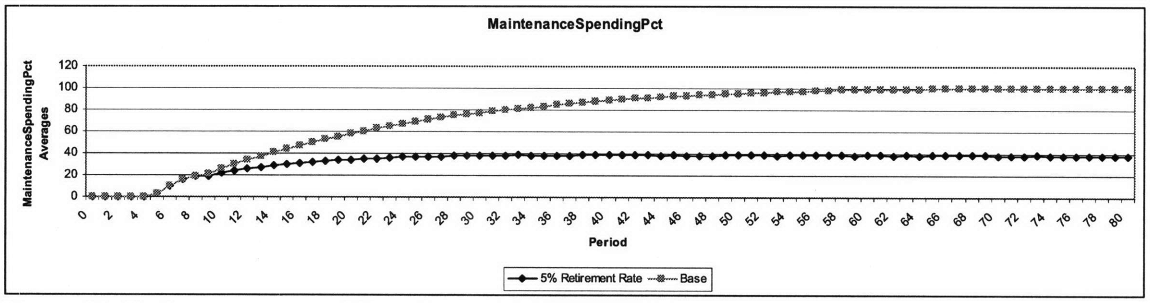


Figure 51: Retirement rate at 5% vs Base – Maintenance spending percentage

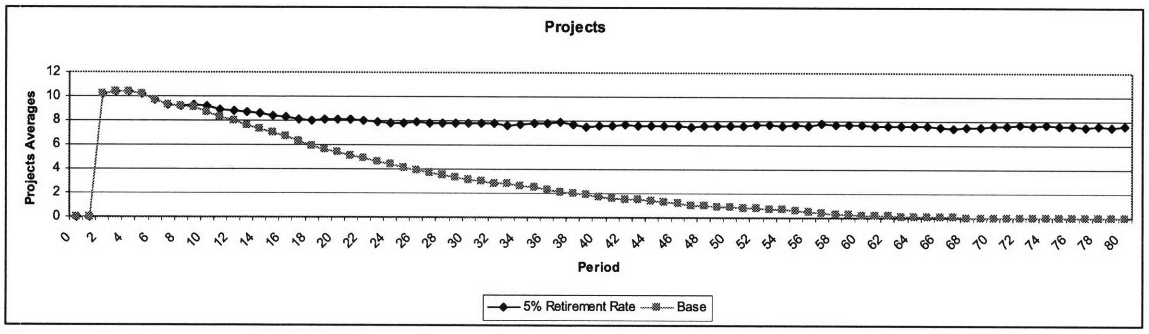


Figure 52: Retirement rate at 5% vs Base – Projects

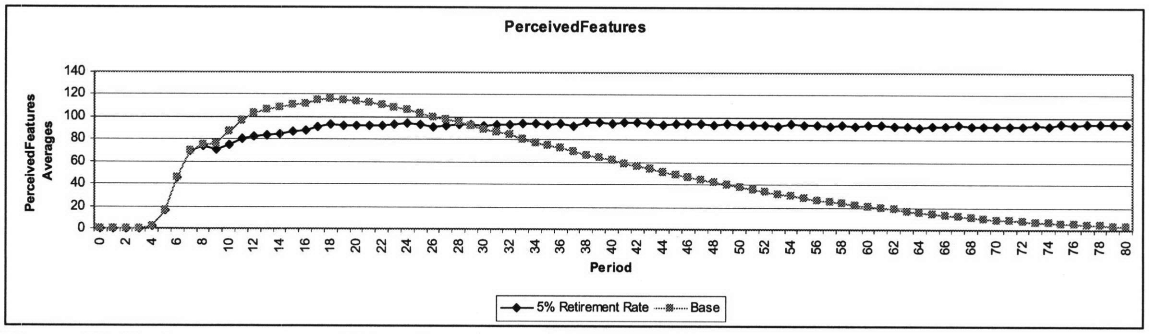


Figure 53: Retirement rate at 5% vs Base – Perceived features

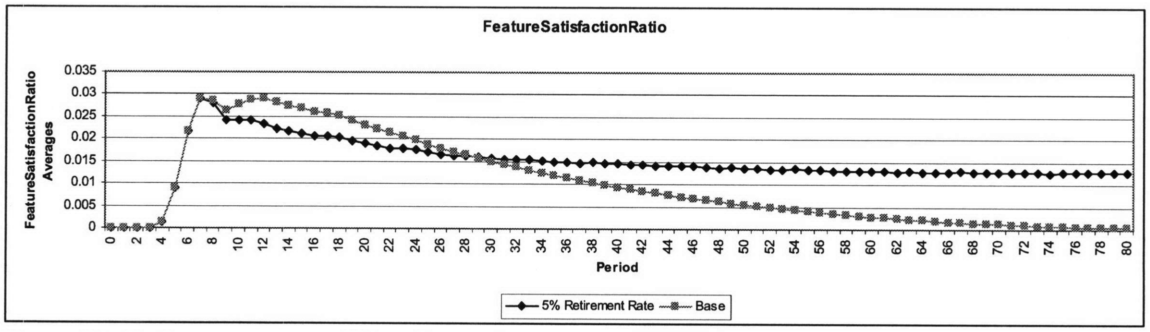


Figure 54: Retirement rate at 5% vs Base – Feature satisfaction ratio

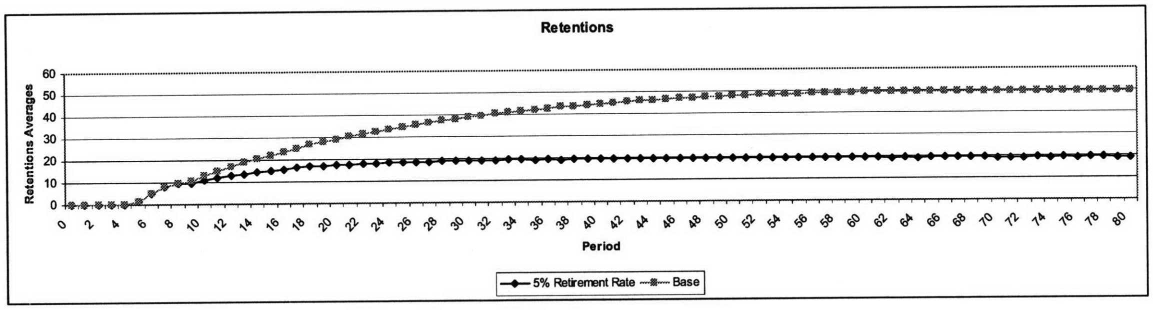


Figure 55: Retirement rate at 5% vs Base – Retentions

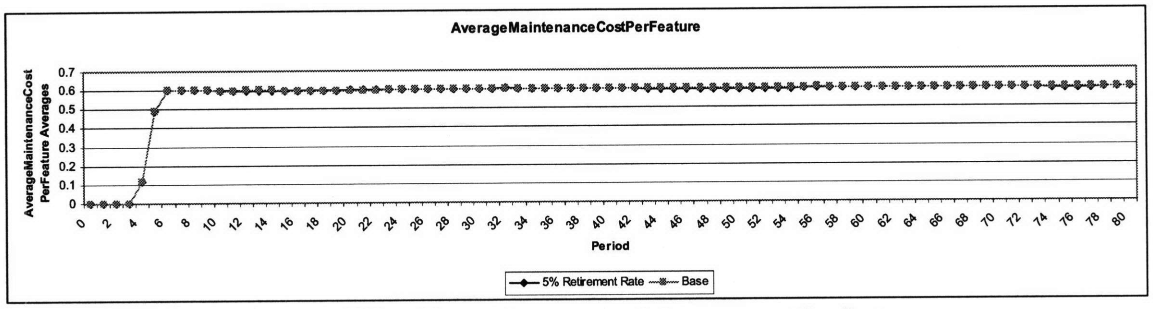


Figure 56: Retirement rate at 5% vs Base – Average maintenance cost by feature

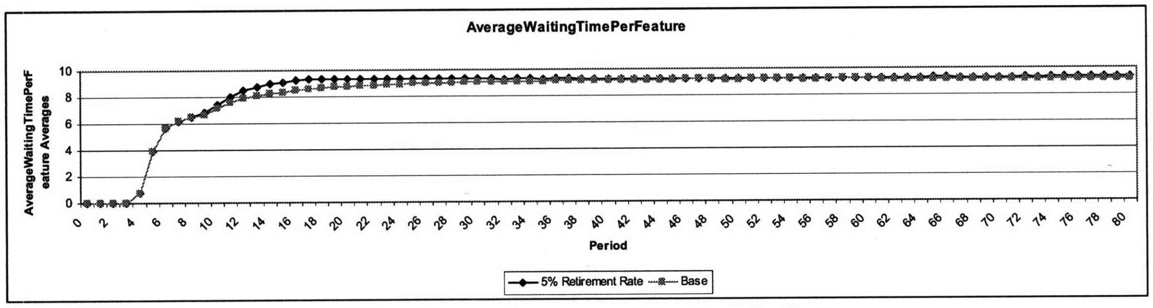


Figure 57: Retirement rate at 5% vs Base – Average waiting time by feature

The immediate observations are on the development spending and the maintenance spending. Rather than a 100% and 0% distribution of IT budget on the maintenance and the development with development getting zero in later periods, we see some sustained funding on the development effort now. Through the retirement process, the IT organization seems to be able to work on significantly more new projects and spend much higher percentage of its budget on development in the long term.

Probably more important, retiring old application actually helped dramatically increase Number of Features Perceived in the long term. The Feature Satisfaction Ratio in general is much higher and more stable as well.

Compared with the base case, the ecosystem is more effective. Even though the number of applications in the Retained Set is significantly less than that in the base scenario; however, the Retained Set contains much higher proportion of newer applications that have lower decay in their features and therefore can provide more functionality to business units.

Through performing retirement, two new major impacts are brought into the ecosystem. The first is that we will free some maintenance budget otherwise used by retired applications each period. The second impact, which is some how more implicit, is actually on the development activity. By analyzing the casual loop diagram, we can see by weakening the budget pressure in the Process Retention (Loop B4), we are actually improving the main loop Ecosystem (B1) as well. Common sense holds well here. A Retained Set that consists of mainly old and decayed applications and that requires significant maintenance effort is more of a liability than of a valuable asset. Even though in the first several periods, it seems that not retiring applications helped achieve slightly higher number of features in the Retained Set before serious feature decay happened; however, it is short lived. After several periods of run, when the feature decay effect was quickly kicked in and became more dominant, the Retained Set will quickly be stuffed with low value applications and become a resource drain when there was no retirement to free budget for new projects.

One interesting finding is that the values of the Number of Features Perceived and the Feature Satisfaction Ratio seem to have the similar "worse before better" pattern described in the scenario where normalization and forcing conforming is used as the selection regime rule (Scenario 3). We have very similar shape for the two sets of curves and the two lines (for Number of Features Perceived) crossed some where. Is there any difference between the two scenarios?

On the surface, it first seems that the scenario using normalization and forcing conforming was just less effective because the improvement to the base scenario happened in later period than that in the scenario when retirement was enforced; but this difference may due to our specific simulation setting. The fundamental difference here is that in the case of enforcing retirement, we guaranteed some portion of development budget for each period so that there would always be new features going into the Retained Set. While in the normalization and forcing conforming scenario, this is not the case. The policy just delay the time when the development work would stop.

If this is the case, is there any reason that IT managers would prefer to use normalization and forcing conforming regime to the retirement of old applications regime? The answer is yes. As we discussed, real world is more complex. It is possible that the IT organization may not have the luxury to retire any applications for some reasons (such as legal requirement). In this case, a different regime set that would provide some similar result of optimization to some degree but with worse scenario might be a legitimate alternative for a short period of time.

Simulation Scenario 5: Put it all together

Through investigation of the simulation scenarios 2, 3 and 4, in which we changed one (or one category of) policy regime relative to the base scenario and then compared their influence against the later, we can see different policy regimes may have different focus in terms of impacts on the IT ecosystem performance measures, some focusing on improving the effectiveness (eg. more perceived features through enforcing retirement policy) or on efficiency (lower maintenance cost through Normalization and Conforming selection regime), while others may be on both (more features at lower average maintenance cost through ROI selection regime). A logic question follows will be what if we can combine those policies, would we be able to make some further difference?

In the following new scenario, I combine the changes in all of the three policy rules into a new scenario as follows to see the overall influence to the IT ecosystem when they work together:

- The retirement rate: set at 5% per period
- Selection regime: uses ROI as the selection regime, performs normalization and forces conforming

The following figures put one measure in the same chart for all the five scenarios we have tested for easier comparison.

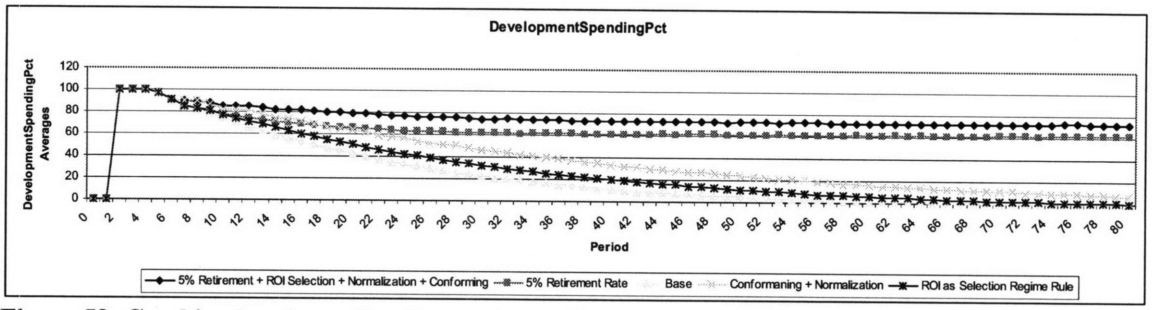


Figure 58: Combined regime – Development spending percentage

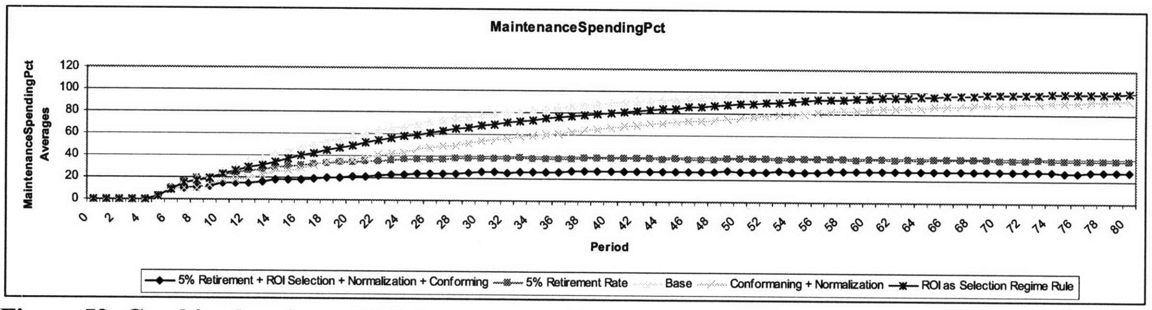


Figure 59: Combined regime – Maintenance spending percentage

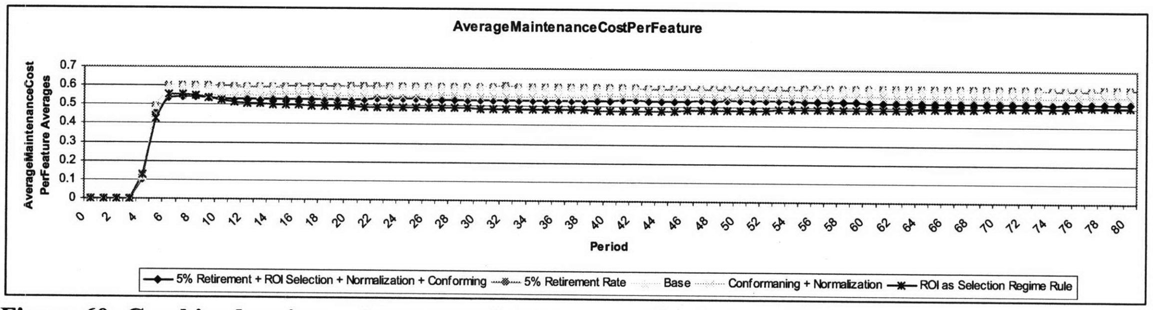


Figure 60: Combined regime – Average maintenance cost by feature

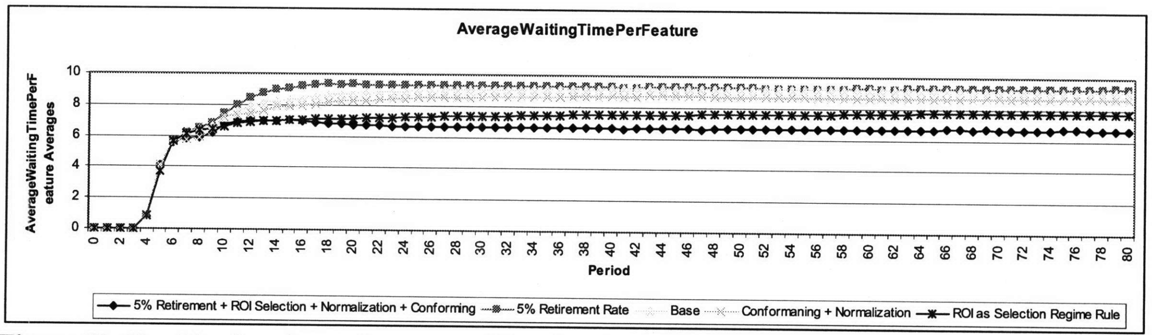


Figure 61: Combined regime – Average waiting time by feature

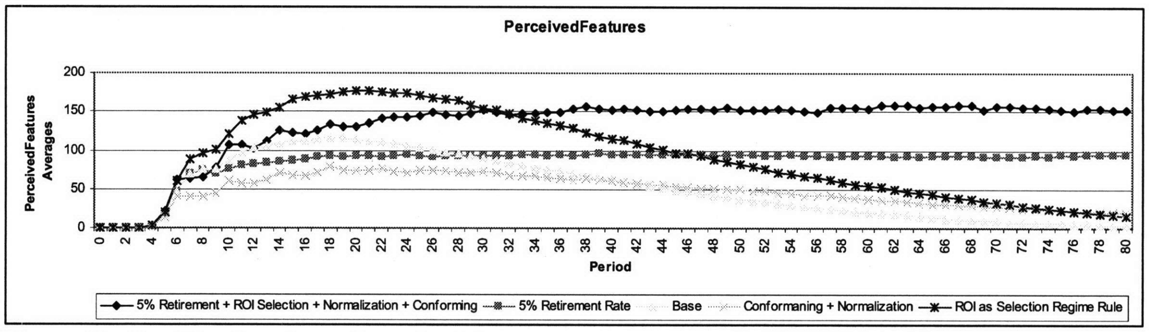


Figure 62: Combined regime – Perceived features

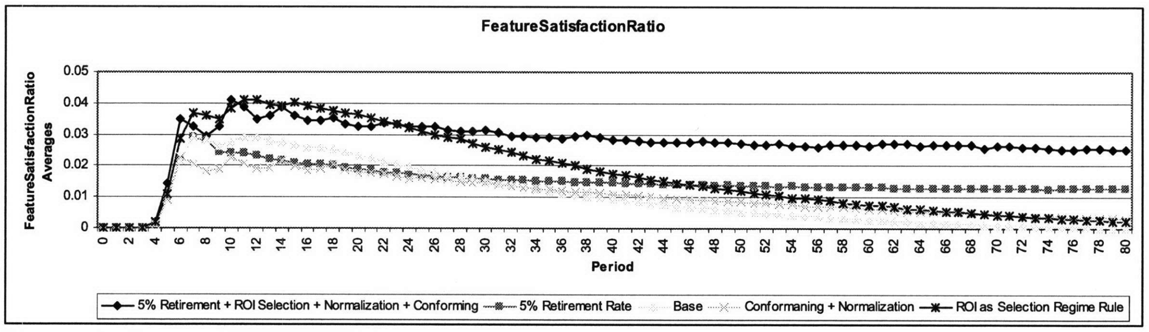


Figure 63: Combined regime – Feature satisfaction ratio

The most obvious observation is that enforcing the retirement of old applications seems to be a distinct change than the rest, as it has changed the pattern, not just the scale levels, of the system behavior. Through the retirement process, the IT ecosystem was able to keep a certain degree of development budget for each period and had much more stable values in Number of Features and Feature Satisfaction Ratio, which none of them was the case in the other scenarios. It seems that appropriate retirement of old applications is critical to having a healthy IT ecosystem in the sense of reviving the Retained Set with new and functioning features, as we can see the improvements happen no matter if we use only the base scenario combined with a retirement policy or if we change a bunch of regimes combined with a retirement policy.

For the impacts of the rest of the policy regimes, in which the basic system behaviors are similar under a fixed budget allocation without retirement of old applications assumption, I have the following observations:

- Selecting new projects by ROI helps increase the numbers of applications and features associated with the Retained Set, given the same resource consumption because of better use of IT budget in terms of development cost achieved in the Development Set and maintenance cost achieved in the Retained Set by feature. ROI selection regime helps enhance both overall IT system efficiency and effectiveness, as IT can provide more features to the business at lower average costs by feature. Using ROI as the selection regime rule seems to have better results than the base scenario in every major performance count.

- Adjusting variations through normalization process and only selecting new projects from those variations that are conforming help reduce the overall cost of IT operations. Given the same budget allocation, we can afford to develop slightly higher number of new projects because of the cost efficiency from lowered average maintenance cost on retained features. However, because of the normalization process, as we have discussed, the number of features provided may be reduced in some periods. Therefore, impact on Number of Features Perceived to the IT ecosystem is not a clear cut. In our particular simulation, it experienced a "worse before better" scenario.

- Overall, IT managers might be able to adopt combinations of "smarter" regimes to achieve better results. For example, in our particular simulation settings, it seems that using ROI as the selection regime particular helps increase the number of features and provides higher user satisfaction. Normalization and Forcing Conforming improves IT ecosystem efficiency through reduction on the maintenance costs, particular in the long term. IT managers can use those different regimes or combination of regimes to help achieve a particular goal. For example, this new scenario that combined several changes of different regimes seemed to have helped achieve a better IT ecosystem in the long term, with predominantly better measures in development/maintenance spending percentage, average waiting time by feature, perceived number of features and feature satisfaction ratio.

- One additional observation is that there might not be one regime that can beat all the rest in all the period and in all the performance measures. For example, we stressed that the key impact of retirement to the overall health of an IT ecosystem under a fixed IT budget allocation assumption. However, if we consider maximizing the Number of Features Perceived in the short term, it seems that no retirement on applications might be a better choice.

Simulation Scenario 6: Test on Budget Regime Impacts

While analyzing above simulation scenarios, I had observed one bothersome fact: The values of the Feature Satisfaction Ratio measure seemed to be quite low. Just as Westerman and Henry (2006) have pointed out, on the one hand, resources allocated to IT variation, selection and retention processes are limited; on the other hand, needs to improve business through IT initiatives seem unlimited. Even though IT could optimize its processes, it is still difficulty for IT to align with the growing business needs without an appropriate growth in IT resource allocation. System Dynamics theory calls this phenomenon “limit to success”. I will see how the amount of budget allocation, and probably more important, the IT budget growth patterns, will affect the IT ecosystem. We will see how IT can improve the Feature Satisfaction Ratio.

Change Budget Allocation with a Lower or Higher Fixed Amount

The first change is to adjust the IT budget allocation amount for each period by 30% in either direction (increase or decrease), relative to that of the base scenario. In this case, the IT budget is still fixed over periods, just at a higher or lower level.

System Dynamics tells us, system structures and relationships among its constructs determine the system behavior. Based on this understanding, I do not think changing the budget allocation amount only will change the system behavior as this change does not fundamentally alter either the structure or relationships. The simulation results confirmed my assumption.

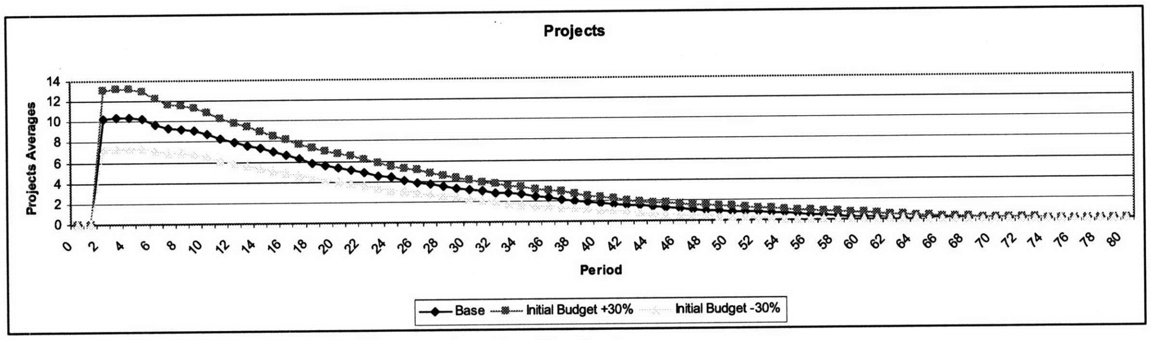


Figure 64: Fixed IT budget at different levels – Projects

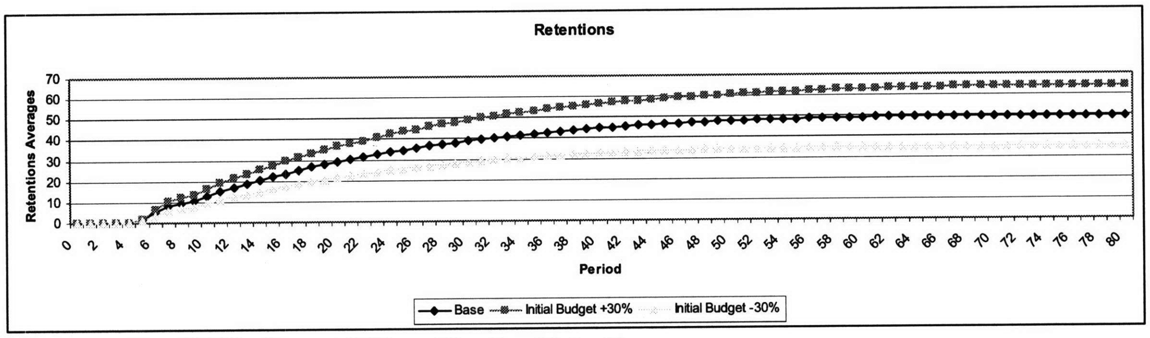


Figure 65: Fixed IT budget at different levels – Retentions

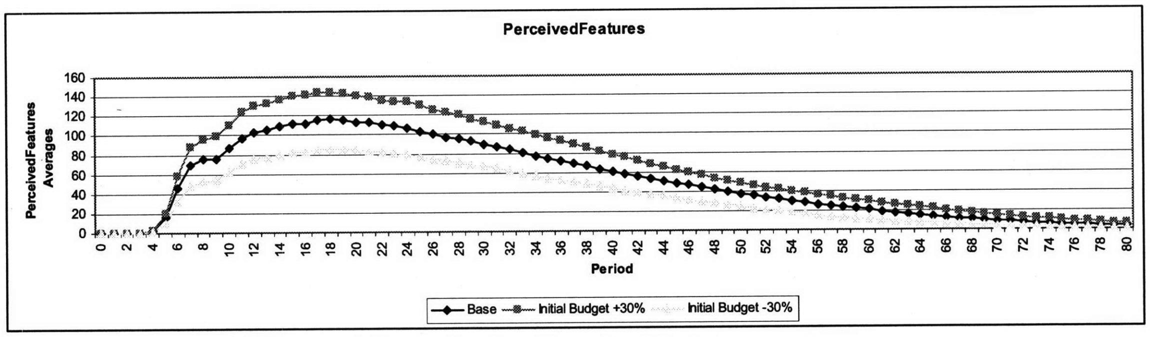


Figure 66: Fixed IT budget at different levels – Perceived features

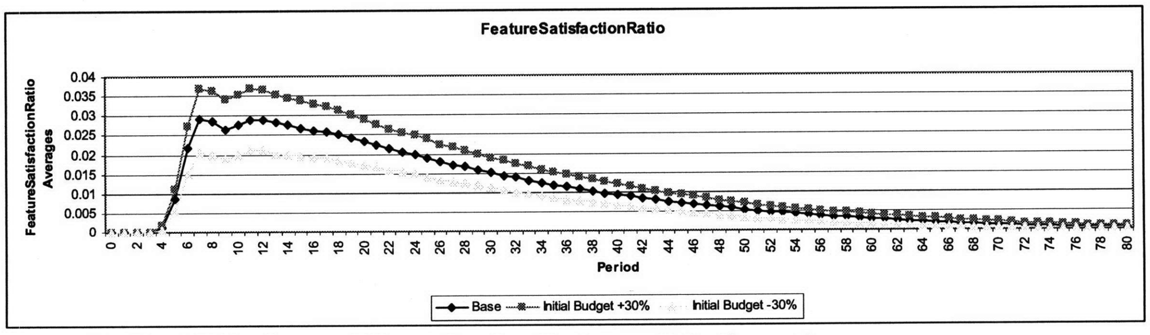


Figure 67: Fixed IT budget at different levels – Feature satisfaction ratio

As it is indicated, some efficiency and effectiveness measures changed their values compared with those in our base case scenario. For example, we will have different levels of projects, applications, and perceived features associated with each of the different budget allocations. However, the basic observations were still the same: Lower value in Number of Features Perceived and continuously lowered Feature Satisfaction Ratio.

I also performed another test with 30% increase in fixed budget allocation, in conjunction with adopting regime settings that we had set up for the scenario five (set retirement rate at 5%, used ROI as the selection regime rule, enabled normalization of variation and forced conforming).

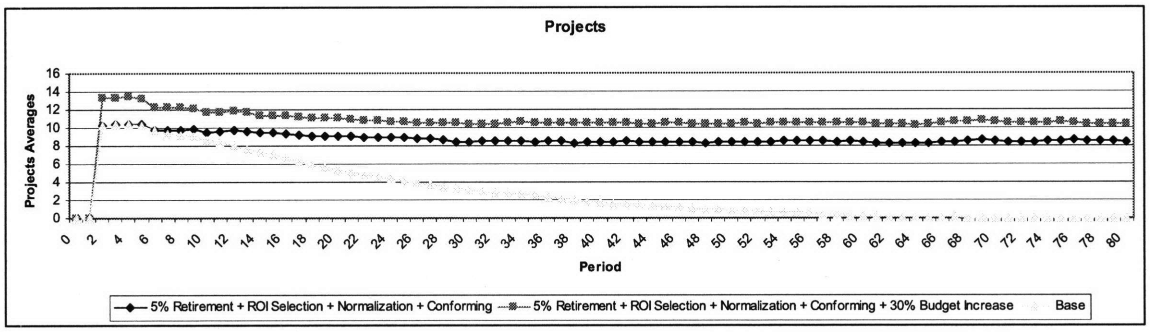


Figure 68: Combined vs Combined + 30% budget increase – Projects

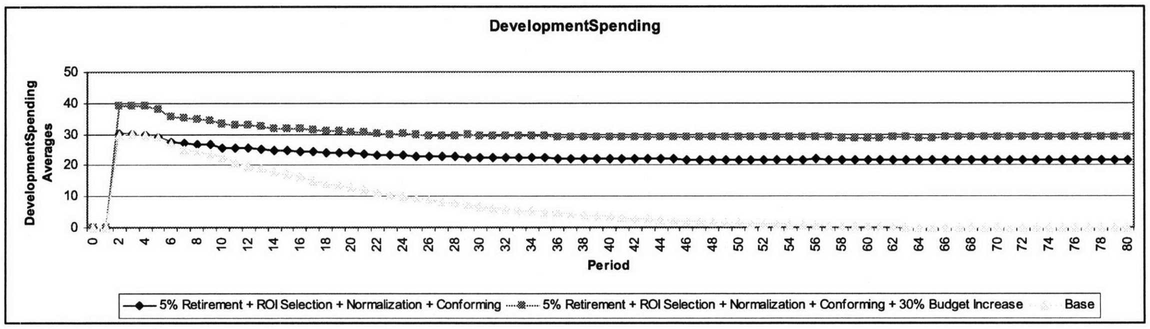


Figure 69: Combined vs Combined + 30% budget increase – Development spending

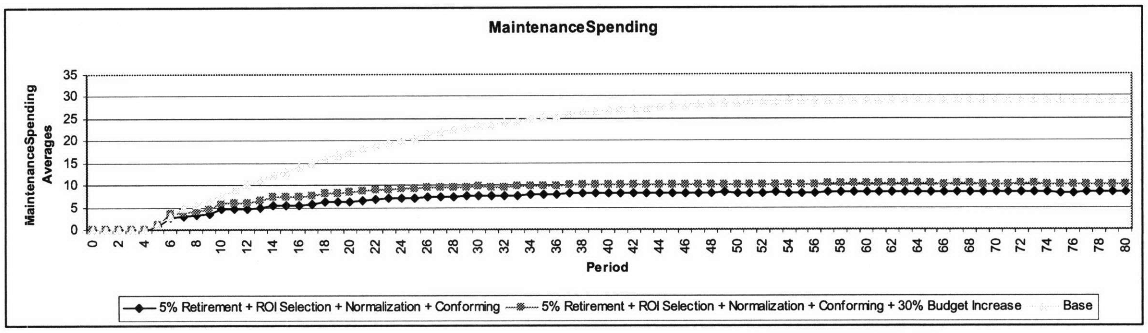


Figure 70: Combined vs Combined + 30% budget increase – Maintenance spending

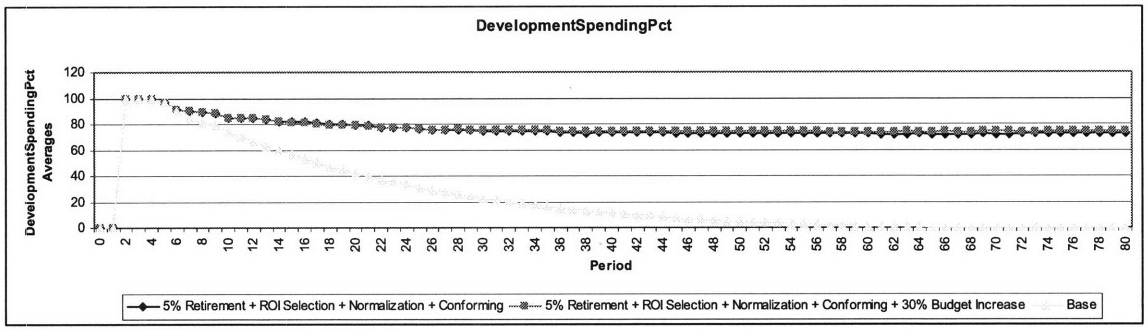


Figure 71: Combined vs Combined + 30% budget increase – Development spending percentage

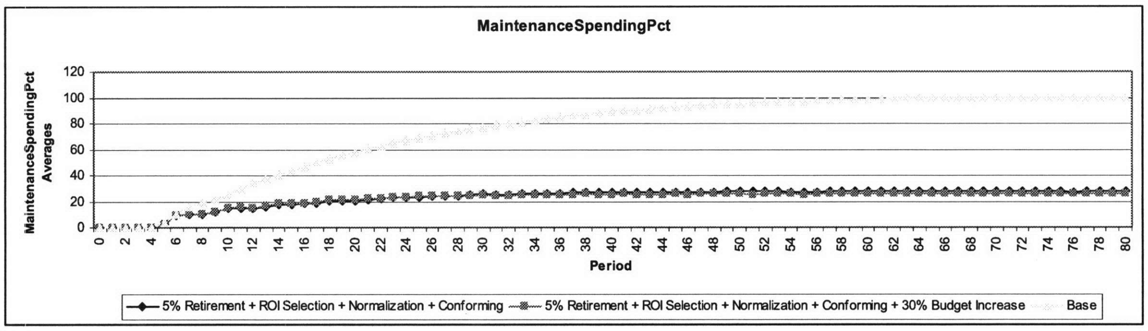


Figure 72: Combined vs Combined + 30% budget increase – Maintenance spending percentage

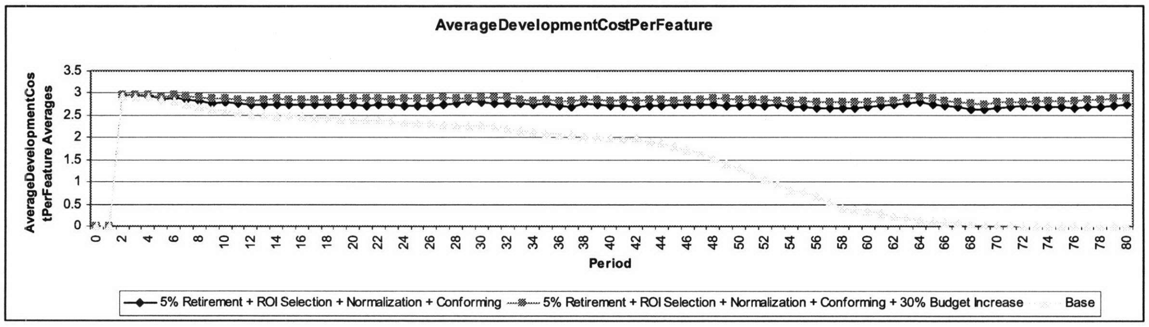


Figure 73: Combined vs Combined + 30% budget increase – Average development cost by feature

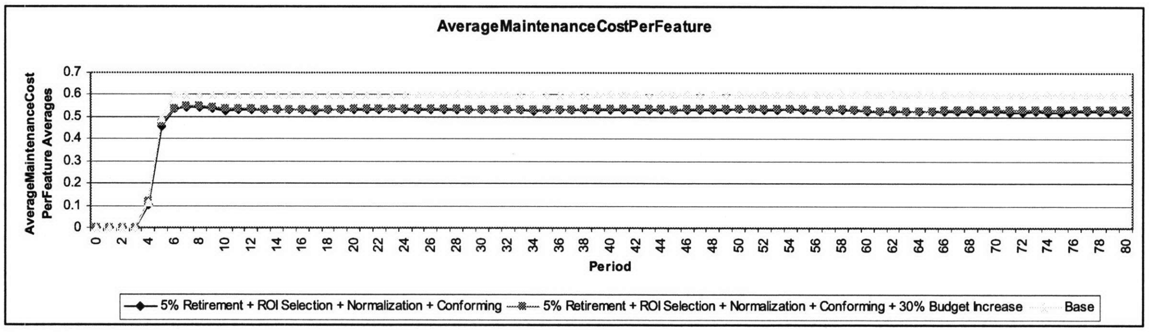


Figure 74: Combined vs Combined + 30% budget increase – Average maintenance cost by feature

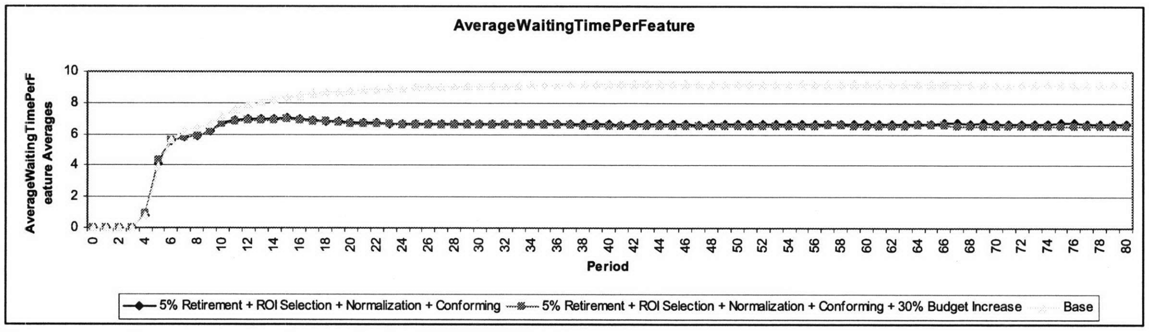


Figure 75: Combined vs Combined + 30% budget increase – Average waiting time by feature

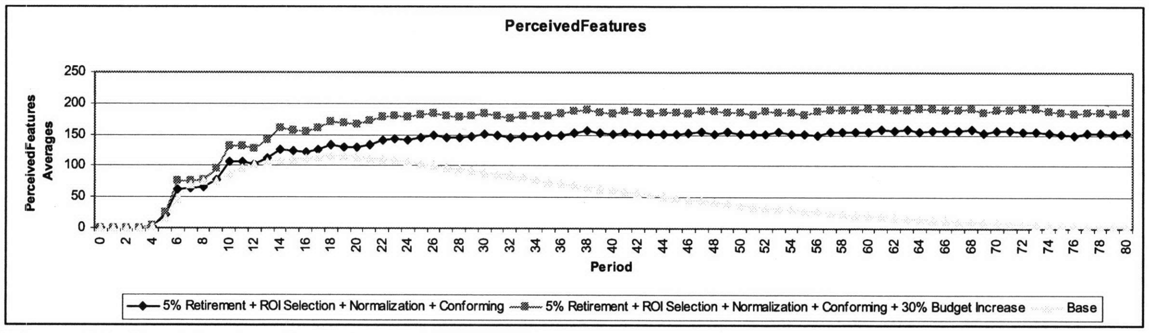


Figure 76: Combined vs Combined + 30% budget increase – Perceived feature

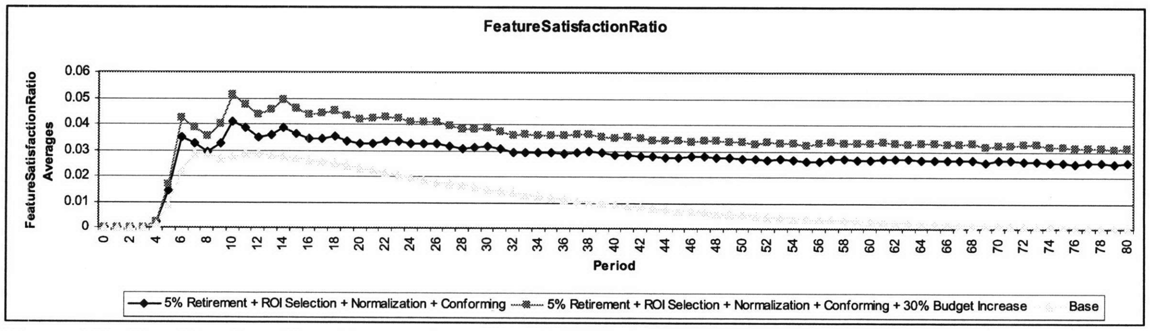


Figure 77: Combined vs Combined + 30% budget increase – Feature satisfaction ratio

This new test did not change the basic pattern that we observed before we increased fixed amount of budget allocation, even though some performance measures (such as Number of Features, Number of Projects, Feature Satisfaction Ratio) were further improved with increased resources available to the IT organization. The changes observed were mostly parallel and the percentages of budget in development and maintenance were also very similar before and after the budget increase. The only exceptions were that after the increase of the budget, the average development cost and the average maintenance cost by feature were slightly higher. Probably the IT governance process loosed its selection regimes when resources are more abundant; improving the effectiveness, while lowering slightly its efficiency measures.

In both of the above two scenarios, we did not account for the possible effect when budget allocation is increased, business units may tend to increase their number of requests (which is probably the case in real world). The improvement we observed might have to be discounted more once we consider this effect.

Change Budget Growth Regime

If changing budget allocation amount to a higher fixed level only will not change the fact that the Feature Satisfaction Ratio will not be fundamentally improved. What if we can make some other adjustments on how IT budget is allocated then see what the impacts might be? In this case, I will consider two different scenarios: grow budget amount at a predetermined fixed percentage each period and grow budget at the same growth rate as that of change rate in number of perceived features.

When we increased the budget amount by 10% for each period, it seemed that the system would not face budget pressure starting from period 35, after which we saw no significant changes in the numbers of variations for both the Initiative Set and the Discarded Set. The number of variations in the Initiative Set flattened out because abundant IT budget allowed all requests from the business units to become projects. In this simulation, since I set value of the Delay in Reviewing Initiative parameter to two (2) periods and had an average number of variations to be created at 30 for each period, the number of variations in the Initiative Set was close to 60 after period 35. In the mean time, no initiatives were discarded after period 35 because all reviewed initiatives went into the Development Set as projects.

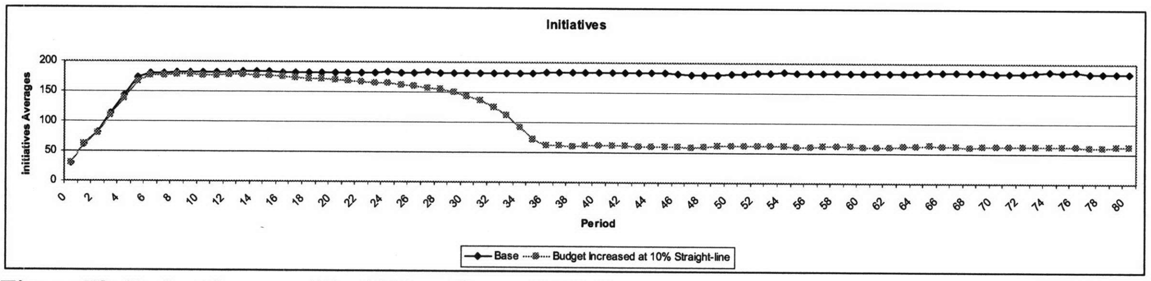


Figure 78: Budget increased by 10% vs Base – Initiatives

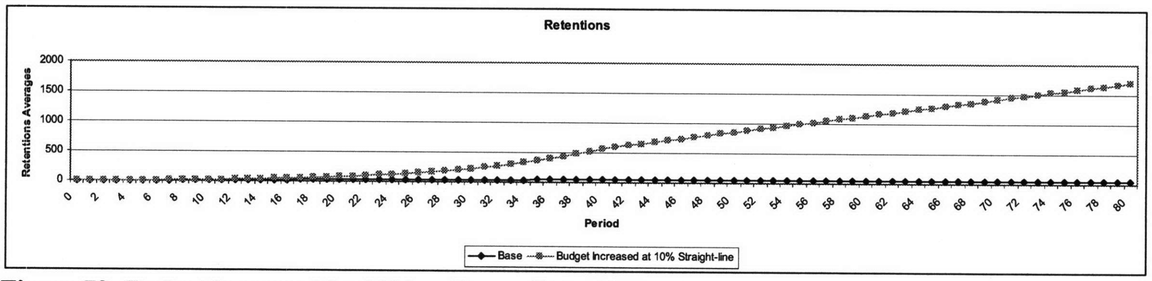


Figure 79: Budget increased by 10% vs Base – Retentions

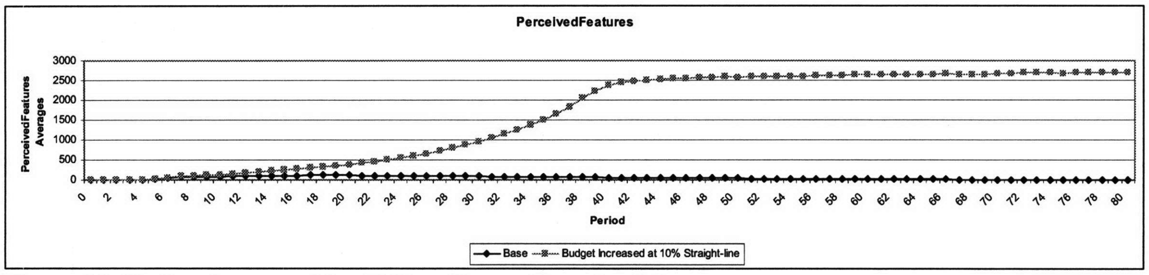


Figure 80: Budget increased by 10% vs Base – Perceived features

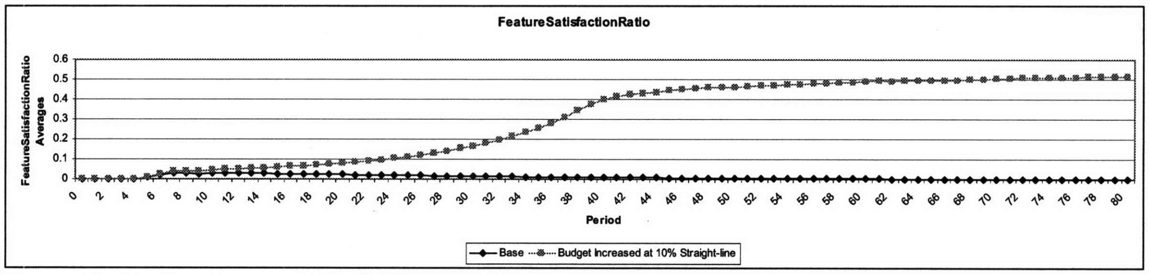


Figure 81: Budget increased by 10% vs Base – Feature satisfaction ratio

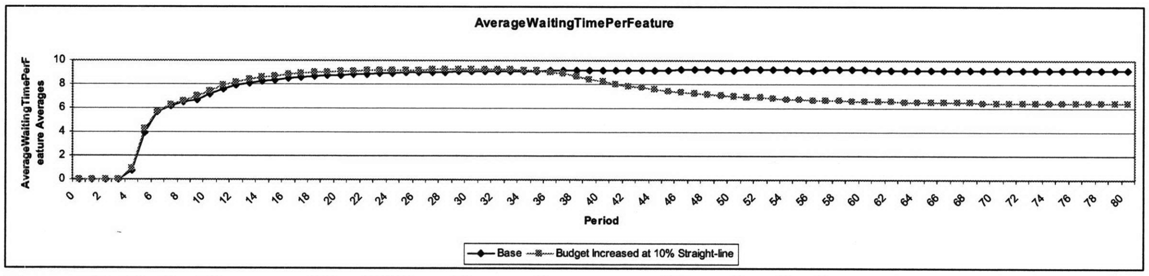


Figure 82: Budget increased by 10% vs Base – Average waiting time by feature

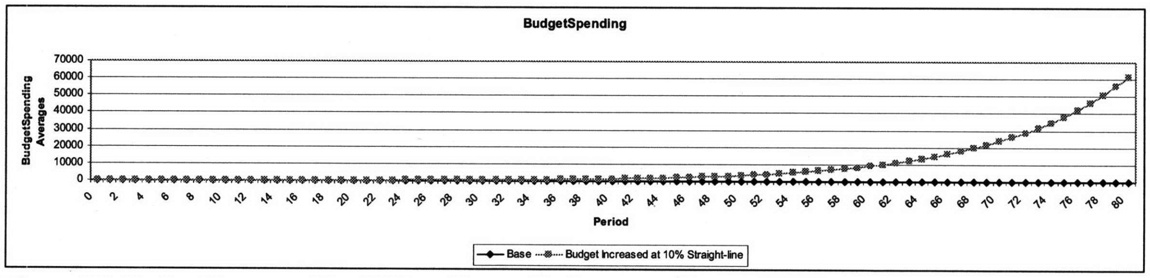


Figure 83: Budget increased by 10% vs Base – Budget spending

In such a resource abundant environment when all new proposals could virtually become projects, IT service level was much higher with increased Number of Features Perceived and Feature Satisfaction. We also had a apparently lower Average Waiting Time for new features in later periods. In addition, Feature Satisfaction

Ratios were stable at a much higher level. We also seem to have much more retained projects and retained features. On the other hand, the total resources (Budget Spending) consumed are much higher, especially for those late periods.

Another budget growth pattern is to allow IT to grow its budget allocation at the same rate as the change rate in number of features delivered. Some organizations use this strategy to link resource consumption directly to delivery or contributions from IT.

My observation is that the effectiveness of IT increases significantly because of the increased resources available to IT.

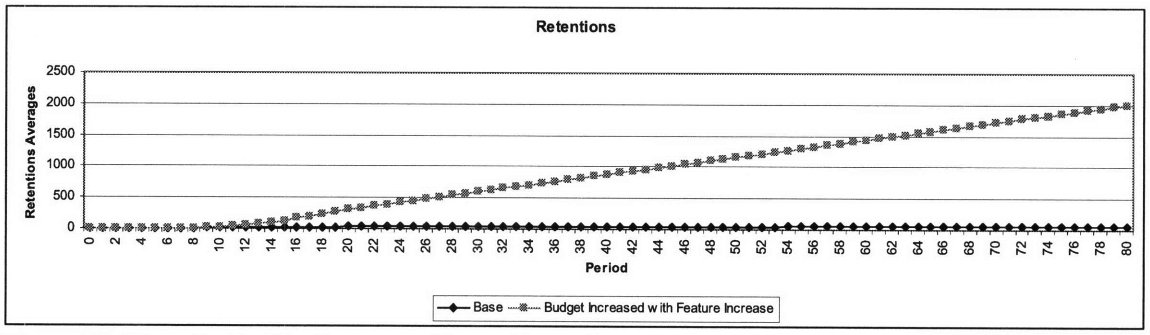


Figure 84: Grow budget by feature vs Base – Retentions

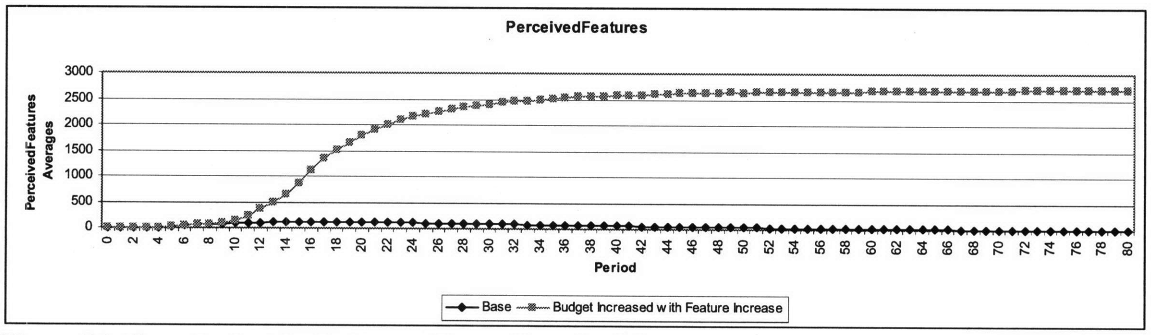


Figure 85: Grow budget by feature vs Base – Perceived features

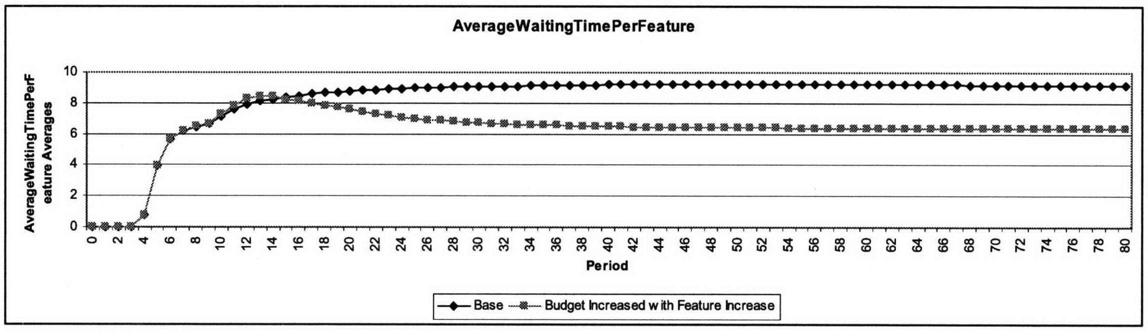


Figure 86: Grow budget by feature vs Base – Average waiting time by feature

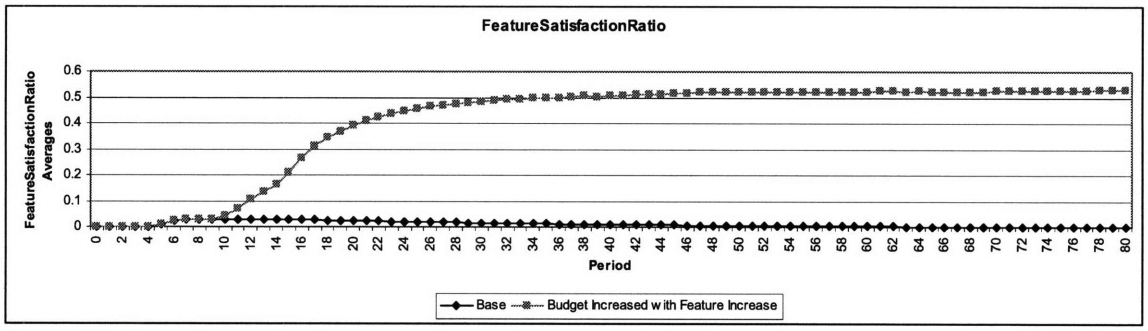


Figure 87: Grow budget by feature vs Base – Feature satisfaction ratio

In the base scenario, the lack of resources is the bottleneck of the system performance, the IT could have done more projects or keep more applications in the system, had the IT had more resources. This is why the service level had been improved in the scenarios when IT could grow its budget allocations either at a fixed percentage change or at the rate of the number of feature growth.

The final comparison on budget regime impacts is between growing budget with features provided versus growing budget at the 10% fixed rate for each period. In our case, the growth by feature growth regime further improved the effectiveness measures. If we look at our model, we can see that this is due to the slow ball effect (positive feedback) of the reinforcing loop Change Budget (R3). When the number of features goes up, IT will be allocated more budget while more budget allocation will further increase the number of features delivered or maintained for the next period. This is the positive impact to the improved IT service level.

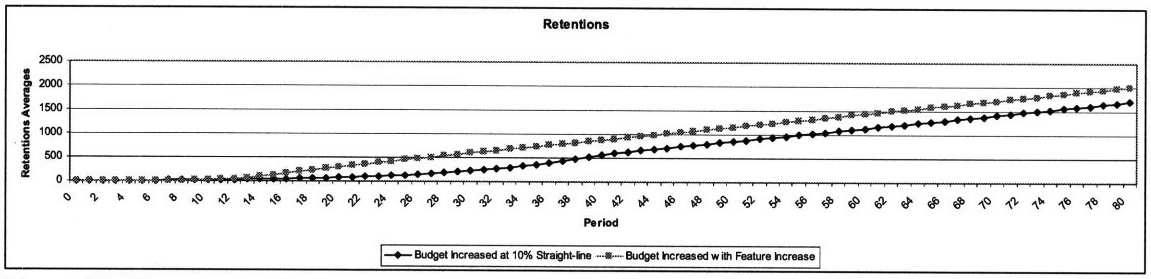


Figure 88: Grow budget by feature vs Grow budget by fixed percentage – Retentions

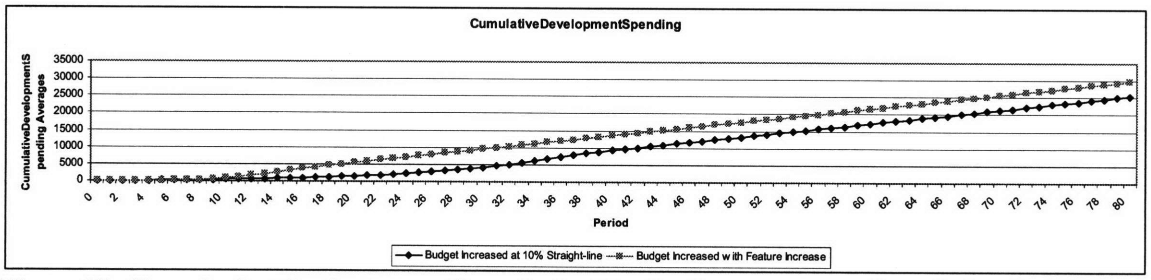


Figure 89: Grow budget by feature vs Grow budget by fixed percentage – Cumulative development spending

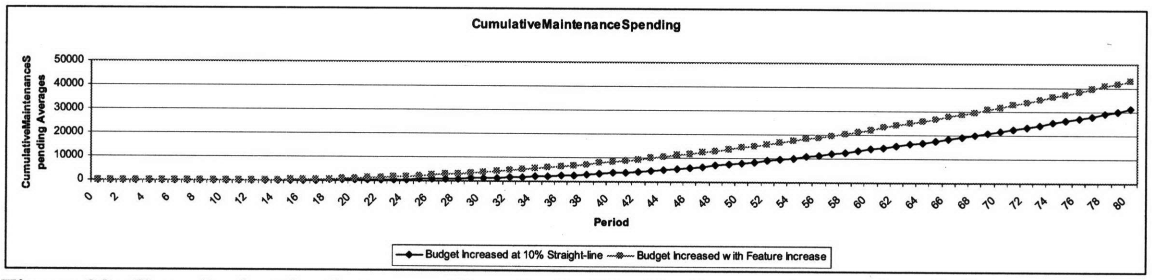


Figure 90: Grow budget by feature vs Grow budget by fixed percentage – Cumulative maintenance spending

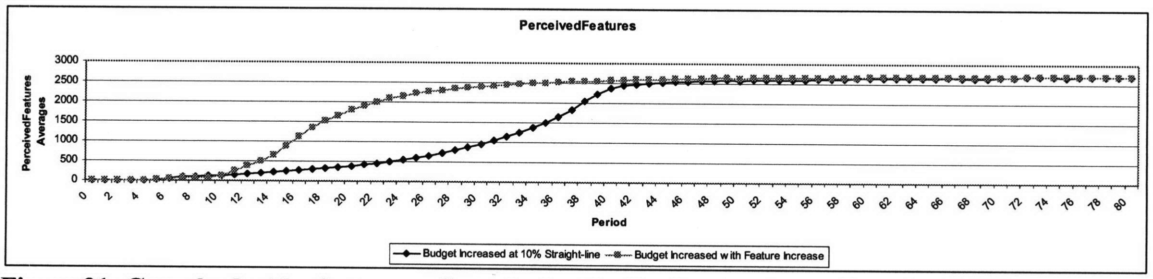


Figure 91: Grow budget by feature vs Grow budget by fixed percentage – Perceived features

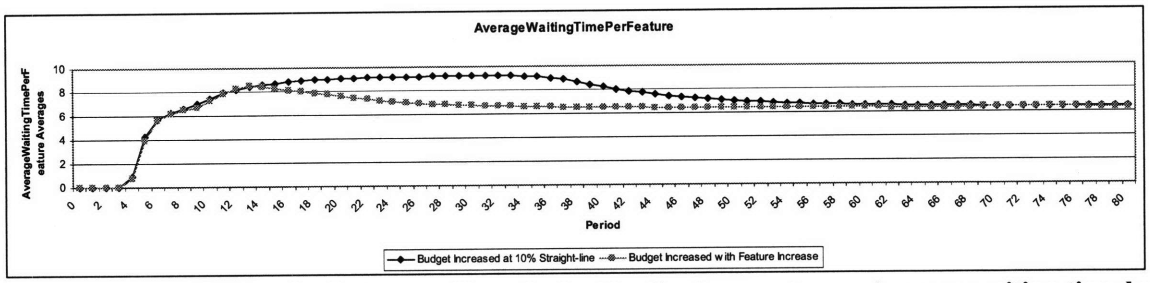


Figure 92: Grow budget by feature vs Grow budget by fixed percentage – Average waiting time by feature

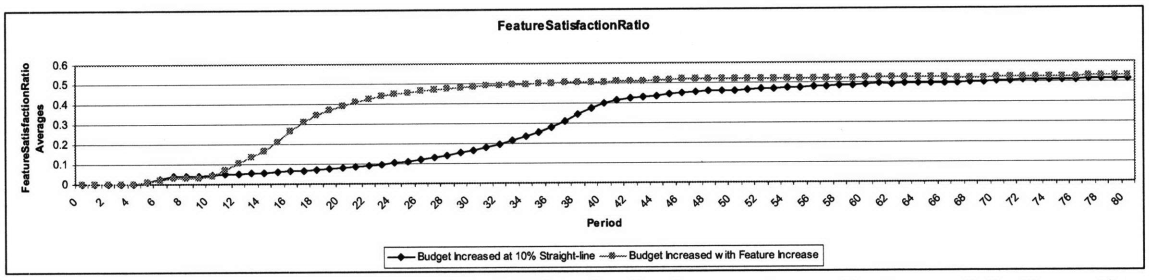


Figure 93: Grow budget by feature vs Grow budget by fixed percentage – Feature satisfaction ratio

However, grow budget by feature grow rate policy also has its price to pay in that IT might have worked on more projects than it should really have (It has more abandoned or failed projects, uses much more total IT budget, and wastes more development budget).

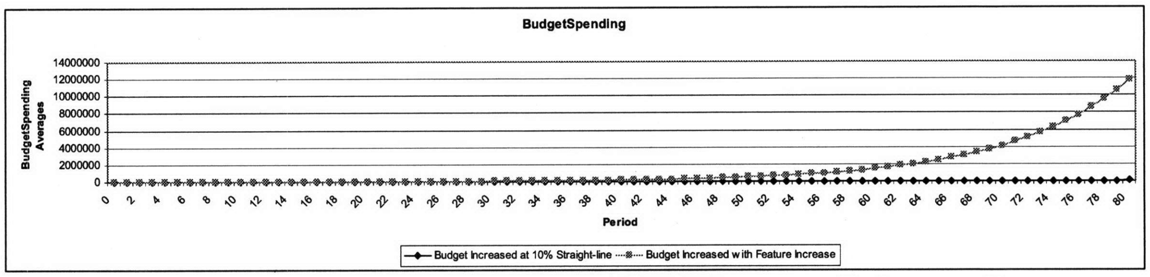


Figure 94: Grow budget by feature vs Grow budget by fixed percentage – Budget spending

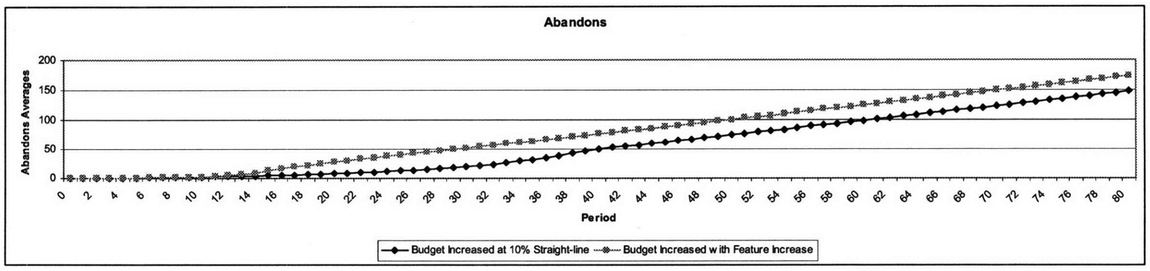


Figure 95: Grow budget by feature vs Grow budget by fixed percentage – Abandoned

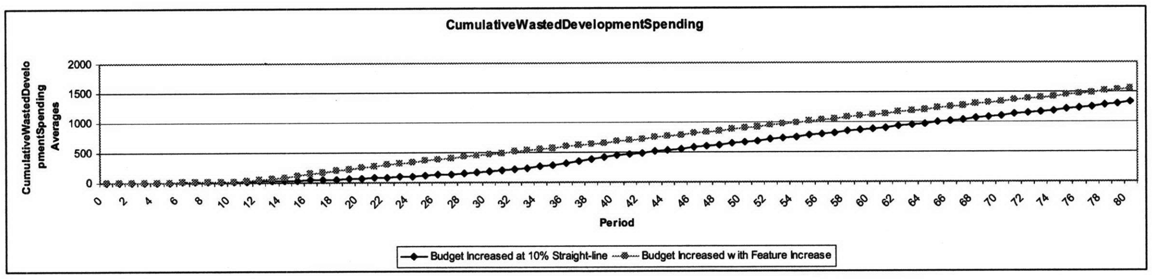


Figure 96: Grow budget by feature vs Grow budget by fixed percentage – Cumulative wasted development spending

In the real world, it is earlier for a firm to use the fixed rate budget increase regime as it is easier to control and more predicable. Although I would argue in some certain situations, increase budget by feature growth rate may be more effective, as the simulation scenario indicated.

We also can see from the charts that the two strategies seemed to generate results that are on the path to converge on some key measures such as Feature Satisfaction Ratio (after period 50) and Number of Perceived Features (after period 40). My understanding is that if we keep increasing the budget allocation even at a small fixed percentage each period, after some periods, the actual budget allocation will become quite big because the increase is in the geometric fashion (compound increase). After some time, both strategies will have extremely abundant resources to spare and would behavior quite the same thereafter.

Another interesting observation is on the Development Set and the Features under Development where they peaked and then became lower before eventually stabilized. This is because in the first several period, budget allocations, even though were increasing, were still relatively small, so some initiatives stayed in the Initiative Set.

Later on, budget became more abundant, those backlog initiatives quickly all became projects such that we saw a peak. Thereafter, all initiatives that had been in the Initiatives Set for the period had all become projects such that Development Set and Features under Development stabilized if the demand (requests to IT form business units) did not increase.

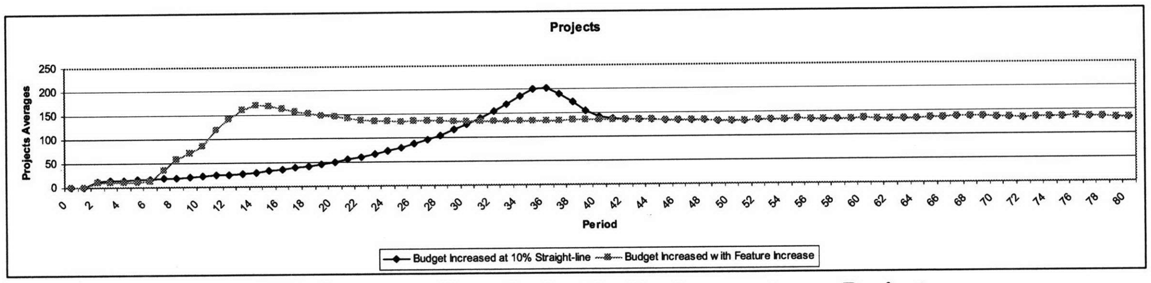


Figure 97: Grow budget by feature vs Grow budget by fixed percentage – Projects

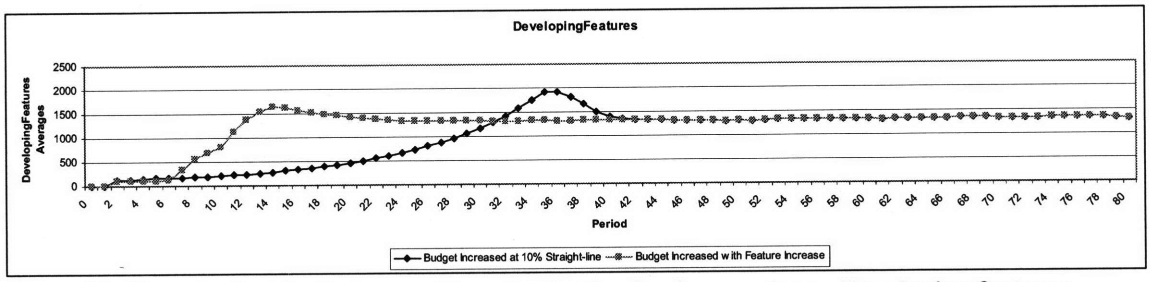


Figure 98: Grow budget by feature vs Grow budget by fixed percentage – Developing features

Simulation Scenario 7: Change Variation Patterns

Westerman and Henry (2006) mentioned that IT may influence the business to change the way how business creates variations. How this would happen is out of the scope for our research at this moment. However, we still can see whether the business practice change may change the IT ecosystem. Fully realizing this impact, an IT organization would reach out to the business, rather than work as a solo, to help shape the business practice in the way that will more likely to achieve better alignment between the two. In this section, we will see impacts on the IT ecosystem when variation process uses different parameter values in generating variations.

Our assumption is that the better quality (higher number of features, or lower costs in development and maintenance, or better conforming to standard technologies) in variations created, the better performance for the IT ecosystem.

Use Different Means for Expected Number of Features for Variations

The base scenario used 10 as the mean of Expected Number of Features to generate variations. I ran a new case that used 12 followed by another that used 8.

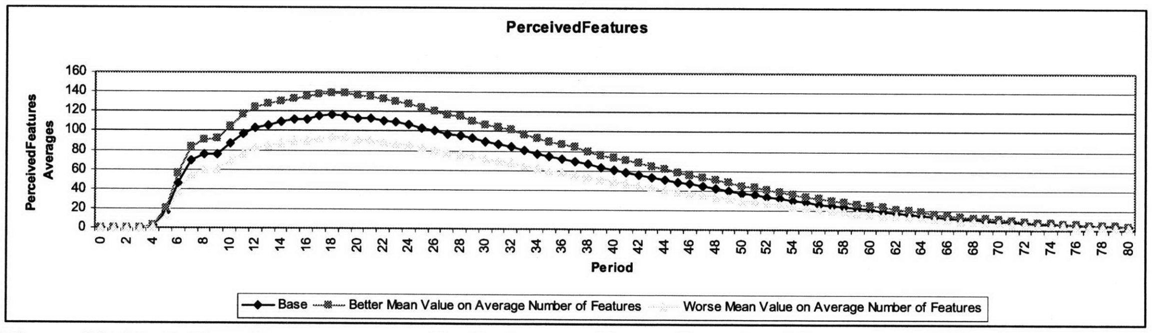


Figure 99: Variation change on mean for features vs Base – Perceived features

As we had expected, higher mean value of Expected Number of Features used in the variation process increases the Number of Features Perceived in the Retained Set.

Use Different Means for Conformance in Variation Creation

When the mean for conformance on variations created is higher, fewer projects will fail and lower budget will be spent on maintaining applications. Therefore, budget is used more efficiently. Our simulation results confirmed my assumption.

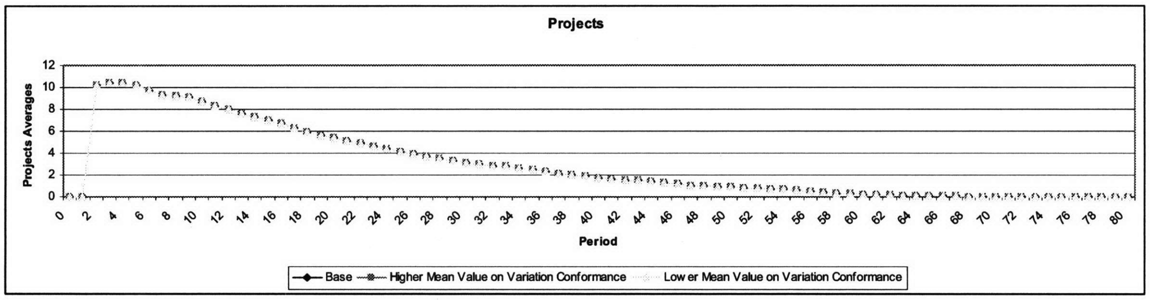


Figure 100: Different mean on conformance – Projects

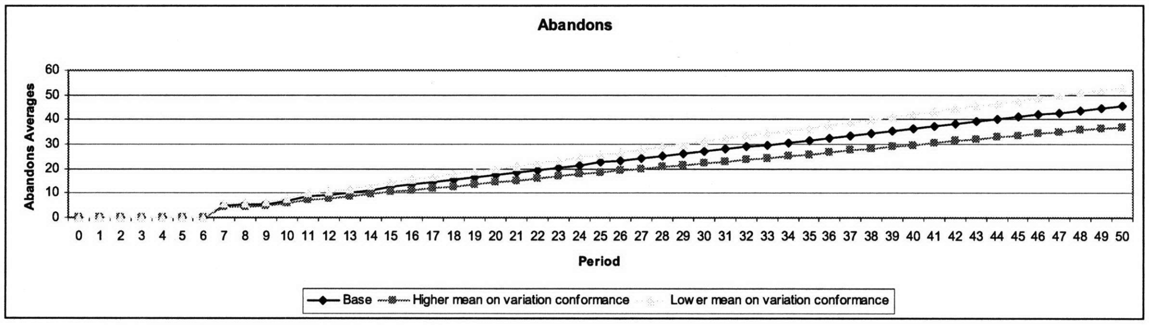


Figure 101: Different mean on conformance – Abandons

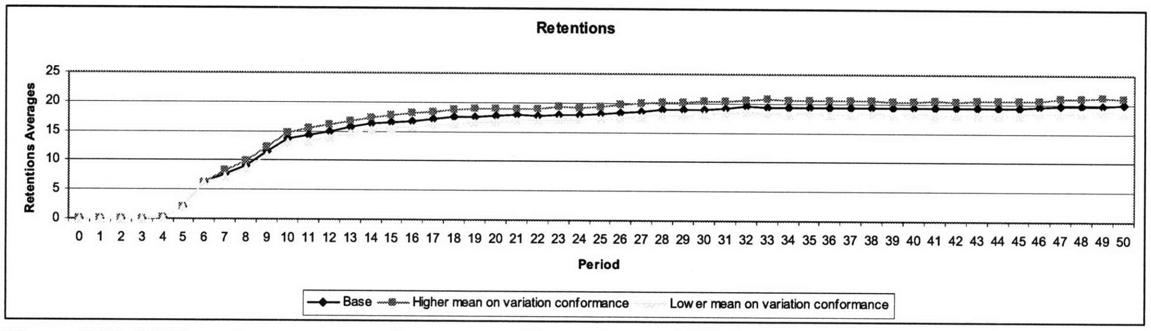


Figure 102: Different mean on conformance – Retentions

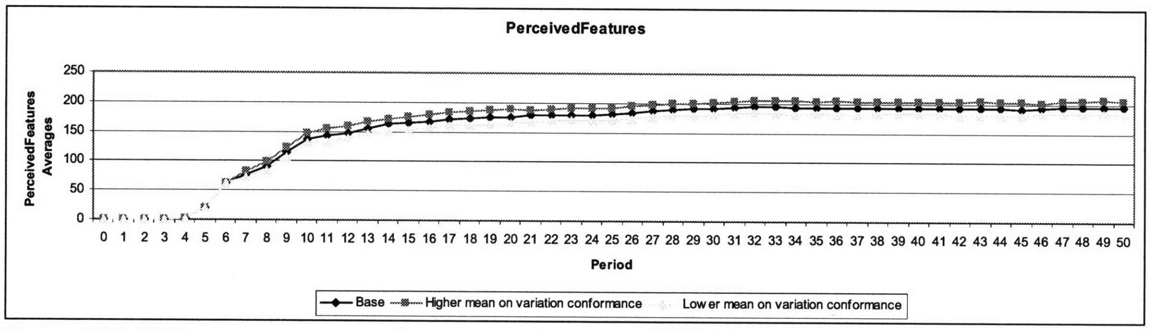


Figure 103: Different mean on conformance – Perceived features

As we can see, all three cases have similar numbers of projects. However, the higher the mean of variation conformance, the lower number in abandoned projects and the ecosystem can sustain more projects and applications, meaning better capability in developing new competitive edge and servicing business with more perceived features.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 7: Discussions

In their proposed IT ecosystem framework, Westerman and Henry (2006) argued that perfect alignment between IT and business or 100% feature satisfaction for IT to serve business is impossible. The main reasons given are limited resources available to IT and the vagueness of business strategies or multiple business strategies. Those are all legitimate reasons. In addition, I found several other factors that may play roles for the misalignment though analysis of our model and the simulation results.

Even if IT can get accurate information about business goals or understand the business strategies very well, IT will still have difficulty achieving a perfect alignment with the business. In our model, we have several delays such as Delay in Reviewing Initiative, Delay in Discarding Initiative and Delay in Abandoning Project that will prevent IT from making immediate changes when business strategies change. For example, the increases in business requests of a period, even with plenty of IT budget allocation, will still NOT make corresponding applications available immediately until those variations are reviewed, developed and deployed, which can be many periods later. This is why we still see, in several of our scenarios, that there are always some initiatives in the Initiative Set even if the IT budget is more than enough to convert them all into projects. Another factor is that individual IT project may have issues described in Project Dynamics by Sterman (1992) and Reichfelt and Lyneis (1999), which we had discussed as Cost Overrun, 90 % syndrome, or effect of rework. In real life, some technological issues designing and implementing IT projects may also lead a project to fail even if the project itself would suit very well to business needs. Some of these scenarios have actually been reflected in our simulation scenarios. For example, even in the case when budget is abundant; we still can not improve the Feature Satisfaction Ratio anywhere close to 100% because of some factors mentioned here.

Although 100% alignment with the business is not an achievable goal for any IT organization, nevertheless it is valuable for IT to understand what drives the alignment so that IT can optimize its work effort to align with the targeted business strategies as much as possible. My research or any other single research effort could not be sufficient to resolve this issue. However, many of my simulation scenarios based on the IT governance framework and System Dynamics theory might help provide some useful findings.

Retirement of out-dated applications is necessary for the health of an IT ecosystem

Just like the metabolism is necessary for any biological entity to maintain or improve its health condition, retirement of out-dated IT applications is critical for sustainable success of an IT ecosystem. Every product has a life cycle, from design, to development, to launch, to maintenance, and eventually to death; IT applications are no exception. If an old application does not provide features in a cost effective way, the organization should phase it out or find better substitute for it. We have pointed out, through analysis of several of our simulation scenarios, that in order to have sustainable resources on developing projects to maintain or expand competitive advantages of a firm, IT needs to reclaim budget from the Retained Set as the funding source for new development. This is especially important when the IT budget will NOT be significantly increased over periods.

Restrictions posted by capacity limits of IT may be a good thing for an organization

Just as a body's reactions (fatigue and pain) to stress from over-use or diseases protect us from further (normally more serious) damage, capacity restrictions of IT' to satisfy business requests can prevent an organization from over-using its valuable resources, thus improving efficiency in delivering values at reasonable costs. Our simulation scenarios indicate this as well. When budget is relatively more abundant (in the cases when budget has significant growth due to the budget growth pattern used), the IT effectiveness measures such as Number of Feature Perceived, Average Waiting Time for a Feature and Feature Satisfaction Ratio are normally better. However, in those cases, the efficiency measures such as Average Development Cost per Feature and Average Maintenance Cost per Feature are normally lower. The organization needs to be aware of this trade-off in order to achieve a right balance between IT efficiency goals and effectiveness goals.

IT managers can possibly help improve the alignment

Our simulation scenarios do indicate some limitations for IT to align with the business strategies. For example, with limited budget allocation while retirement of old applications is not allowed for some reason, the Feature Satisfaction Ratio probably starts low and will get even lower. However, on the other hand, our discovery indeed shows that to some degree, IT managers could possibly help achieve better alignment better by adopting appropriate IT regime policy rules that are better matches to the business strategies. We had discussed several performance measures on IT ecosystem effectiveness and efficiency. We also discussed, in some cases, that impacts of a policy rule may not be the same to the IT ecosystem in different time frames. So we will talk about the policy impacts from both the measure and time horizon stand points. Table 5 shows the definitions for the three time horizons we use.

Time Frame	Value
Short Term	Less than 20 periods
Medium Term	20 to 40 periods
Long Term	40 to 60 periods

Table 5: IT ecosystem time horizons

Table 6 summarizes five measures that can be used to evaluate IT ecosystem performance in terms of efficiency and effectiveness and what simulation variables can be used to help track the measures. The higher are the values of the measures, the better performance the IT ecosystem.

Category	IT Capability Measure	Measured By
Efficiency	Maintenance Efficiency	Reversed value of Average Maintenance Cost per Feature
	Development Efficiency	Reversed value of Average Development Cost per Feature
Effectiveness	New Feature Delivery Speed	Reversed value of Average Waiting Time by Feature
	Development Capability	% of Development spending of IT budget
	Service Level	Value of Number of Features Perceived
	User Satisfaction	Value of Feature Satisfaction Ratio

Table 6: IT ecosystem performance measures

Table 7 shows how we mark the performance impact a particular policy change brings up compared with the base case. Most of the symbols used and interpretations are self explanatory, for example, when we can observe significant improvement on IT ecosystem performance by changing the value of a parameter, we will use a ++ to designate the change as significantly better. A "N/A" is used in some scenarios when the measure can not be interpreted or does not have a definitive conclusion. For example, in the scenario when there is no budget increase and not retirement, the development cost per feature from the base case does not mean much in the later periods because there is no development activity in those periods when maintenance work would consume whole budget allocation. In most of the cases, the policy regime changes may have two directions (enable/disable or increase/decrease). To simplify discussions, I normally take the side that will either enable a regime or increase its value. The effects of disabling the regime or decreasing its value can be deducted easily.

Symbol	Interpretation
--	Significantly worse
-	Somewhat worse
0	Same or no significant difference
+	Somewhat better
++	Significantly better
N/A	Not Applied

Table 7: IT ecosystem performance changes notations

Tables 8, 9 and 10 show impacts on the IT ecosystem relative to the base case (no retirement, selection and retention regime rules using time proposed of variations and no budget increase) for those selected measures in terms of time horizons for short, medium and long terms respectively.

Measure Policy	Development Efficiency	Maintenance Efficiency	Speed to New Features	Service Level	User Satisfaction	Development Capability
Retiring old applications	0	0	+	-	-	+ to ++
Normalization + Conforming	0	+	+	--	--	+
Selection by ROI	0	+ to ++	+ to ++	++	+ to ++	0 to +
Selection by Feature	0	0	0 to ++	0 to ++	0 to ++	0
Improved variation quality	+	+	+	+	+	+

Table 8: Policy impact on IT ecosystem performance -- Short term

Measure Policy	Development Efficiency	Maintenance Efficiency	Speed to New Features	Service Level	User Satisfaction	Development Capability
Retiring old applications	-	0	0	- to ++	- to 0	++
Normalization + Conforming	-	+	+	- to 0	-- to 0	+
Selection by ROI	0	++	++	++	++	+
Selection by Feature	0	0	+	++ to +	++ to +	0
Improved variation quality	+	+	+	+	+	+

Table 9: Policy impact on IT ecosystem performance -- Medium term

Measure Policy	Development Efficiency	Maintenance Efficiency	Speed to New Features	Service Level	User Satisfaction	Development Capability
Retiring old applications	N/A	0	0	++	0 to ++	++
Normalization + Conforming	N/A	+	+	0 to +	0 to ++	+
Selection by ROI	N/A	++	++	++ to +	++	0
Selection by Feature	0	0	++	+ to 0	+ to 0	0
Improved variation quality	N/A	0	0	0	0	0

Table 10: Policy impact on IT ecosystem performance -- Long term

A sample interpretation using the "Retiring old applications" entries from the three tables tell us that in the short term, adopting this policy has significant negative impact on the service level measure (e.g. Number of Features Perceived) in the short term. In the medium terms, it turned things around as the impacts change from significantly worse to significantly better. In the long term, the positive impact to the system performance is very apparent.

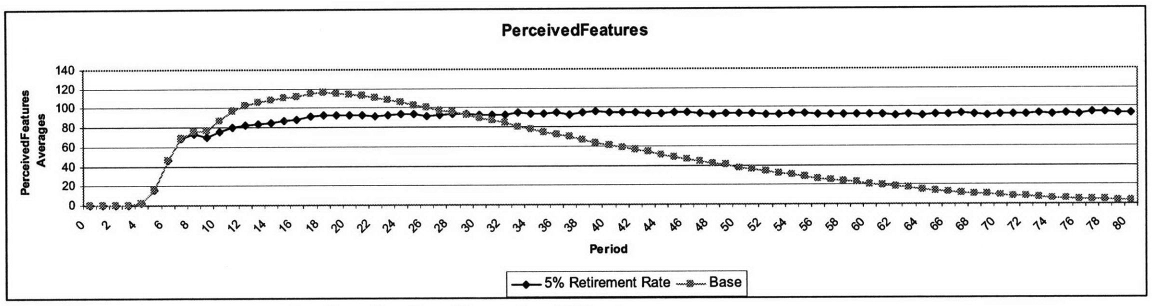


Figure 104: Measure service level using Perceived Features

Please note our comparisons are based on the simulation scenarios from chapter six. The magnitudes of differences in performance measures may change depending on how aggressively we change the values of regime variables as well. For example, if we raise the rate for retirement values even higher, the feature gap and maintainability may change their values too. However, the basic pattern will still be similar.

The above are just general observations based on my simulation analysis, rather than time tested rules. In practice, IT managers should look at them judiciously and be ready to see some exceptions. In addition, IT managers need to understand the dynamic nature of the IT ecosystem behavior, especially things observe that are counter-intuitive. We can see many policies have their impacts to the IT ecosystem the similar way for different time horizons (such as impacts adopting Normalization and Forcing Conforming on the Speed to New Feature). However, we do see some whose policy impacts are not the same in different time phases. For example, Retiring Old Applications, as our simulation results indicated, may have better Service Level (measured by Number of Features Perceived) in the long term, but it could lower the value of this measure in the short term as we pointed out. Managers need to be aware of this possible “worse before better” scenario.

Another key issue is that IT managers need to use the right measure to help them to address the right issues to achieve better alignment with the business. This first seems to be a strange argument. However, in real life, sometimes it is not very clear which measure(s) would be useful to appropriately measure what we want. One example is the use of ROI as the selection regime. It seems that it can help improve both the efficiency (measured by maintenance cost per feature) and the effectiveness (measured by Feature Satisfaction Ratio and Number of Perceived Features). However, if an organization defines the efficiency as the number of projects that can be covered by a specific amount of IT budget, then using ROI selection regime may not help given our ROI definition is the ratio of Number of Features Perceived over Working Maintenance Cost of a project, which is not related to the number of projects in the IT ecosystem. In this case, the IT organization may have to use a different measure to evaluate implications of a policy change.

Better alignment starts from and ends at working with the business

Our discussion, which has been focused on the IT side of the alignment issues, seems to be based on the assumption that IT people understand the business strategies and priorities so that they can use the model to test scenarios and possibly arrive at a desirable result given such understanding. However, this understanding is not a given but needs some efforts from both IT and the business.

More important, how well IT is aligned with business is not determined by how good the number is from running our simulation or even any models, but ultimately by the real quality of service to business. IT managers need to understand not only how to manage IT project portfolios but also how to reach out to business units to get useful inputs on what the business needs and what actually drives the alignment goals.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 8: Limitations and Future Work

Like all other models or frameworks, the IT governance model described in this thesis is a simplification or a view of a much more complex domain that any one single model or framework could NOT possibly describe fully and accurately. We have mentioned that we actually left out a lot of details by making assumptions; however the real world is always more complex. Often some of those details may become very important. For example, we discussed that while generating our variation data, we did not consider correlations among several key variables (such as between number of features and development cost). Another simplification made is assuming all features are equally import and mutually exclusive if they do not belong to the same project where in this case they are 100% correlated (either we do a project for all of its features or none of those features if the project is not chosen). However, as we all know, among IT projects, there are plenty of interdependences, as one can not be built before a specific application has been built. In this case, we just conceptually bundle the two applications as a unit and count them as one variation in our model. This view may not be legitimate in the real world.

One important and useful extension of my work is to do further calibration of the model, perform empirical analysis on assumptions used, and use extensive case studies to help modify and enhance it. Modifying our model design and implementation of the simulation application to allow data feed from real enterprise project portfolio repertories, rather than use random numbers generated internally from the application as the data source for simulation, would make a step closer to the real world.

While designing and implementing the simulation applications, I simplified many of the process scenarios, for example, a user can only choose one from feature, conformance, ROI or proposed time as the selection rule. It would add flexibility if the application could be expanded to allow selection of multiple rules.

THIS PAGE INTENTIONALLY LEFT BLANK

Reference

Aguanno, Kevin (Editor)
2003, Managing Agile Projects
Multi-Media Publications Inc. Ontario CANADA

Ali, A., Kalwani, M. U., and Kovenock, D.
1993, "Selecting product development projects: Pioneering versus incremental innovation strategies"
Management Science, vol. 39, pp. 255–274, March. 1993.

Banker, R. D., S. Slaughter
1997, "A field study of scale economies in software maintenance"
Management Sci. 43(12) 1709–1725.

Bharadwaj, Anandhi
2000 "A Resource-based Perspective on Information Technology Capability and Firm Performance: An Empirical Investigation"
MIS Quarterly, Vol. 24, No. 1, pp 169-196 March 2000

Broadbent, Marianne and Weill Peter
1999, "The Implications of Information Technology Infrastructure for Business Process Redesign"
MIS Quarterly, Vol. 23 No. 2, pp. 159-182/June, 1999

Brown, K. Carol and Magill L. Sharon
1994, "Alignment of the IS Functions With the Enterprise: Toward a Model of Antecedents"
MIS Quarterly, December, 1994

CFO Publishing
2003, CFO Publishing survey

Detlev J. Hoch, Cyriac Roeding, Sandro K. Lindner
2000, Secrets of software success: management insights from 100 software firms around the world
Harvard Business School Press, Boston, Massachusetts, 2000

Eduardo S. Schwartz and Carlos Zozaya-Gorostiza
2003, "Investment under Uncertainty in Information Technology: Acquisition and Development Projects"
Management Science, Vol. 49, No. 1, January 2003 pp. 57–70

Ford, David N., and Sterman, John D.
1998, "Dynamic Modeling of Product Development Processes"

System Dynamics Review, Volume 14, Number 1, spring 1998, pp. 31-86.

Forrester, Jay W.
1961, Industrial Dynamics
Productivity Press, Cambridge, MA, ISBN-10:0-262-56001-1

Guan, Jian Cheng, Yam, Richard C.M. etc.
2004 "A study of the relationship between competitiveness and technological innovation capability based on DEA models"
European Journal of Operational Research, July 2004

Jahnke, Art
2004, "Why Is Business-IT Alignment So Difficult?"
CIO Magazine, June 2004 issue

Jeffery, Mark and Leliveld, Ingmar
2004 "Best Practices in IT Portfolio Management",
MIT Sloan Management Review Vol. 45, No. 3, pp. 41–49 spring 2004

Gurbaxani, Vijay, Melville, Nigel and Kraemer Kenneth
2000, "The Production of Information Services: A Firm-Level Analysis of Information Systems Budgets", Information Systems Research, Vol. 11, No.2, June 2000, pp 159-176

Heidenberger, K.
1996, "Dynamic project selection and funding under risk: A decision tree based MILP approach",
Eurpeanr. Journal. Operational.Research., vol. 95, pp. 284–298, December. 1996.

Henderson, J.C and Venkatraman, N.
1992, "Strategic Alignment: A model for organizational transformation through information technology," in T. Kochan & M. Unseem, eds, Transforming Organisations, Oxford University Press, NY, 1992

Kontio, J.
1999, "Risk management in software development: a technology overview and the riskit method"
Software Engineering, 1999. Proceedings of the 1999 International Conference
1999 Page(s): 679 – 680

Kweku, Ewusl-Mensah and Zblgnlew H. Przasnyski
1991, "On Information Systems Project Abandonment: An Exploratory Study of Organizational Practices"
MIS Quarterly, March 1991

Levine, Harvey A. and Wideman, Max
2005 "Project Portfolio Management: A Practical Guide to Selecting Projects, Managing Portfolios, and Maximizing Benefits"

Jossey Bass, ISBN: 0787977543

Lyytinen, K. and Hirschheim, R.
1987, "Information System Failures – A Survey and Classification of the Empirical Literature"
Oxford Surveys in Information Technology (4), 1987, pp. 257-309

Markowitz, Harry M.
1952, "Portfolio Selection",
Journal of Finance, 7 (1), 77-91, 1952

McFarlan, Warren
1981, Portfolio approach to information systems.
Harvard Business Review (September-October 1981): 142-150

Naomi Oreskes, Kristin Schrader-Frechette, and Kenneth Belitz
1994, "Verification, validation, and confirmation of numerical models in the earth sciences"
Science, 4 Feb 1994, 263, 641-646.

Parker, Marlyn and Robert Benson
1988, Information Economics: Linking Business Performance to Information Technology, Prentice-Hall, Englewood Cliffs, New Jersey

Papageorgiou, Anargyros and Paskov, Spassimir
1998, "Deterministic Simulation for Risk Management"
Portfolio Management, fall, 1995, 113-120

Pinto, J.K. and Slevin, D.P
1987, "Critical Factors in Successful Project Implementation"
IEEE Transactions Engineering Management (EM-34:1), February 1987, pp. 22-28

Reichfelt, Kimberley and Lyneis, James
1999, "The Dynamics of Project Performance: Benchmarking the Drivers of Cost and Schedule Overrun"
European Management Journal Vol. 17 No. 2 April 1999

Rengarajan S. and Jagannathan P.
1997, "Project selection by scoring for a large R&D organization in a developing country",
R&D Manage., vol. 27, pp. 155–164, Apr. 1997.

Schneidewind, N.
1987, "The state of software maintenance".
IEEE Trans. Software Engineering. 13(3) 303–310. 1987

Senge, Peter M.

1990, "The Fifth Discipline: The Art and Science of the Learning Organization"
Doubleday Publishing, Netcong, NJ, ISBN: 0385517254

Society for Information Management (SIM)
2006, Society for Information Management (SIM) annual survey

The Standish Group Report
1995, Standish Group

Staw, B.M. and Ross, J.
1987, "Knowing When to Pull the Plug"
Harvard Business Review, March-April, 1987, pp. 68-74

Sterman, John D.
1992, "System Dynamics Modeling for Project Management"
Online publishing at: http://web.mit.edu/jsterman/www/

Sterman, John
2000, Business Dynamics – Systems Thinking and Modeling for a Complex World
Irwin/McGraw-Hill, ISBN 0-07-238915X

Todd, Datz
2003, "PORTFOLIO MANAGEMENT How to Do It Right"
CIO Magazine, May 1st, 2003

Westerman, George and Henry, Ronald
2006, "IT Governance as Ecology: Toward a general process theory of alignment", 2006

White, K.B.
1984, "A Diverse, Balanced Team Approach Generates Vita Management Strategy"
Data Management (22:9), September, 1984, pp. 31-40

Wikipedia
2007, http://en.wikipedia.org/wiki/IT_portfolio_management

THIS PAGE INTENTIONALLY LEFT BLANK

Appendices

Appendix 1: Software Package Used for Developing and Running the Simulation Application

The following software package are needed for the development of the simulation applications

- Operating System: Microsoft Windows 2000/XP
- Database Management System: Microsoft SQL Server Express: Free software downloaded from http://msdn.microsoft.com/vstudio/express/sql/
- Java Development Kit (JDK 6u 1) and Netbeans Integrated Development Environment: Open source software at http://java.sun.com/javase/downloads/index.jsp
- Microsoft Java Database Connection driver: free software from Microsoft at http://msdn2.microsoft.com/en-us/data/aa937724.aspx
- Java Database Connection Pool DDConnectionBroker: Open source software from http://opensource.devdaily.com/ddConnectionBroker.shtml

To run the simulation application, one does not have to have Netbeans Integrated Development.

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix 2: System Requirements for Developing and Running the Analysis Application

- Operating System: Microsoft Windows 2000/XP
- Microsoft Excel 97 or above
- Microsoft ActiveX Data Object (ADO): Data access component downloaded from http://support.microsoft.com/kb/183606

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix 3: Ecosystem Database Definitions

This section lists major table definitions for used in the simulation and analysis applications.

Simulation Table

Simulation Table			
Field	**Data Type**	**Description**	**Allow Null**
SimulationName	varchar(250)	Name of the simulation	N
SimulationID	int	Identifier of the simulation	N
Iterations	int	Number of iterations of the simulation	N
SimulationDescription	nvarchar(500)	Description of the simulation	N
Periods	int	Number of periods of the simulation	N
StartingBudget	real	Starting budget	N
LogDetails	bit	Whether to log detail level information into database	N
DevelopmentUnitCostSeed	int	Development unit cost random number generator seed	N
DevelopmentDurationSeed	int	Development duration random number generator seed	N
ConformanceOnDevelopmentCostSeed	int	Conformance on development cost seed	N
ConformanceSeed	int	Conformance seed	N
NumberOfFeaturesSeed	int	Number of features seed	N
ConformanceOnFeaturesSeed	int	Conformance on features seed	N
MaintenanceUnitCostSeed	int	Maintenance unit cost seed	N
ConformanceOnMaintenanceUnitCostSeed	int	Conformance on maintenance unit cost seed	N
NumberOfOpportunitiesEachPeriodSeed	int	Number of proposals each period seed	N
FailureProbabilitySeed	int	Failure probability seed	N
StartTime	datetime	Simulation start time	Y
EndTime	datetime	Simulation end time	Y

Table 11: Simulation table definition

Summary Table

Summary Table			
Field	**Data Type**	**Description**	**Allow Null**
SimulationID	int	Identifier for a simulation	N
IterationID	int	Identifier for an iteration	N
PeriodID	int	Identifier for a period	N
Proposals	real	Number of proposals	N
Requests	real	Number of Requests	N
Initiatives	real	Number of Initiatives	N

Discards	real	Number of discarded initiatives	N
Projects	real	Number of projects	N
Abandons	real	Number of abandons	N
Retentions	real	Number of retentions	N
Retired	real	Number of retired	N
DevelopmentCosts	real	Development cost for the period	N
MaintenanceCosts	real	Maintenance cost for the period	N
CumulativeDevelopmentSpending	real	Cumulative development spending	N
CumulativeMaintenanceSpending	real	Cumulative maintenance spending	N
CumulativeWastedDevelopmentSpending	real	Cumulative wasted development spending	N
RetainedSetConformance	real	Retained set conformance	N
ConformanceThreshold	real	Conformance threshold	N
BudgetSpending	real	Budget allocation for the period	N
AverageDevelopmentCostPerFeature	real	Average development cost per features	N
AverageMaintenanceCostPerFeature	real	Average maintenance cost per features	N
AverageWaitingTimePerFeature	real	Average waiting time per features	N
ProposedFeatures	real	Number of features proposed	N
RequestedFeatures	real	Number of features requested	N
InitiativeFeatures	real	Number of features of initiatives	N
DiscardedFeatures	real	Number of features of discarded projects	N
DevelopingFeatures	real	Number of features under development	N
AbandonedFeatures	real	Number of features of abandoned projects	N
PerceivedFeatures	real	Number of features of applications	N
RetiredFeatures	real	Number of features of retired projects	N

Table 12: Summary table definition

Budget Table

Budget Table			
Field	**Data Type**	**Description**	**Allow Null**
SimulationID	int	Identifier of a simulation	N
IterationID	int	Identifier of an iteration	N
PeriodID	int	Identifier of a period	N
AllocatedBudget	real	Allocated budget	N
LowerBudget	real	Lower budget spending bound	N
UpperBudget	real	Upper budget spending bound	N
MaintenanceBudget	real	Maintenance budget spending	N
CommittedDevelopmentBudget	real	Committed development budget spending	N

Table 13: Budget table definition

PeriodOption Table

PeriodOption Table

Field	Data Type	Description	Allow Null
SimulationID	int	Identifier of a simulation	N
PeriodID	int	Identifier of a period	N
ReviewInterval	int	How many period to review settings	N
StraightlineBudgeting	bit	Whether use straight-line budget regime	N
BudgetStraightlineChangePortion	real	Budget change percentage	N
LowerBudgetLimit	real	Lower budget limit	N
UpperBudgetLimit	real	Upper budget limit	N
GrowingBudgetWithFeatures	bit	Grow budget by feature provided	N
NormalizingVariationsByConformance	bit	Normalize variation	N
ConformanceOnFailure	real	Conformance impact on failure probability	N
ConformingThreshold	real	Conformance threshold	N
ConformanceMean	real	Conformance mean	N
ConformanceStd	real	Conformance standard deviation	N
DevelopmentCostMean	real	Development cost mean	N
DevelopmentCostStd	real	Development cost standard deviation	N
DevelopmentDurationMean	real	Development duration mean	N
DevelopmentDurationStd	real	Development duration standard deviation	N
MaintenanceCostMean	real	Maintenance cost mean	N
MaintenanceCostStd	real	Maintenance cost standard deviation	N
NumberOfFeaturesMean	real	Number of features mean	N
NumberOfFeaturesStd	real	Number of features standard deviation	N
NumberOfOpportunitiesMean	real	Number of proposal mean	N
NumberOfOpportunitiesStd	real	Number of features standard deviation	N
FailureProbabilityMean	real	Failure probability mean	N
FailureProbabilityStd	real	Failure probability standard deviation	N
ConformanceImpactOnDevelopmentCostMean	real	ConformanceImpact on development cost mean	N
ConformanceImpactOnDevelopmentCostStd	real	ConformanceImpact on development cost std	N
ConformanceImpactOnFeaturesMean	real	Conformance impact on features mean	N
ConformanceImpactOnFeatureStd	real	Conformance impact on feature std	N
ConformanceImpactOnMaintenanceCostMean	real	Conformance impact on maintenance cost mean	N
ConformanceImpactOnMaintenanceCostStd	real	Conformance impact on maintenance cost std	N
RetiringByDeployedTime	bit	Whether retire by deployed time	N
RetiringMostExpensive	bit	Whether retire the most expensive application	N
RetiringByFeatures	bit	Whether retire by features	N
RetiringByConformance	bit	Whether retire by conformance value	N
RetiringByROI	bit	Whether retire by ROI	N
RetiringPortionOfMaintenanceCost	real	Percentage of maintenance cost to retire each period	N
MinRetentionPeriods	int	Minimum retention periods for applications	N
SelectingByFeatures	bit	Select project by number of variation features	N
SelectingByProposedTime	bit	Select project by time proposed	N

SelectingByROI	bit	Select project by ROI	N
SelectingByConformance	bit	Select project by conformance value	N
DelayInReviewingInitiatives	int	Delay in reviewing initiatives	N
DelayInDiscardingInitiatives	int	Delay in discarding initiatives	N
DelayInAbandoningProjects	int	Delay in abandon projects	N
MaxAllowedFailurePossibility	real	Maximum allowed failure probability for a project	N
ForcingConforming	bit	Whether force conforming	N

Table 14: PeriodOption table definition

Variation Table

Variation Table			
Field	**Data Type**	**Description**	**Allow Null**
SimulationID	int	Identifier of the simulation	N
IterationID	int	Identifier of the iteration	N
PeriodID	int	Identifier of the period	N
SequenceID	int	Variation sequence number	N
DevelopmentUnitCost	real	Base development unit cost	N
MaintenanceUnitCost	real	Base maintenance unit cost	N
NumberOfFeatures	real	Number of features	N
Conformance	real	Conformance value	N
ExpectedDevelopmentDuration	real	Expected development duration	N
ConformingOnDevelopmentUnitCost	real	Conforming on development unit cost	N
ConformingOnMaintenanceUnitCost	real	Conforming on maintenance unit cost	N
ConformingOnNumberOfFeatures	real	Conforming on number of features	N
FailureProbability	real	Failure probability	N
DiscardedPeriod	real	Discarded Period	N
ProjectPeriod	int	Project Period	N
AbandonedPeriod	int	Abandoned Period	N
ApplicationPeriod	int	Application Period	N
RetiredPeriod	int	Retired Period	N

Table 15: Variation table definition

VariationStatus Table

VariationStatus Table			
Field	**Data Type**	**Description**	**Allow Null**
SimulationID	int	Identifier of the simulation	N
IterationID	int	Identifier of the iteration	N
PeriodID	int	Identifier of the period	N
VariationPeriodID	int	Period variation created	N
VariationSequenceID	int	Variation sequence number	N
StatusCode	int	Status code	N
Normalized	bit	Whether the variation is normalized for the period	N

WorkingConformance	real	Working conformance	N
WorkingDevelopmentUnitCost	real	Working development unit cost	N
WorkingMaintenanceUnitCost	real	Working maintenance unit cost	N
WorkingNumberOfFeatures	real	Working number of features	N
WorkingFailureProbability	real	Working failure probability	N

Table 16: VariationStatus table definition

VariationStatusCode Table

StatusCode Table			
Field	**Data Type**	**Description**	**Allow Null**
StatusID	int	Status code ID	N
StatusDescription	varchar(50)	Short description	N
LongDescription	varchar(500)	Long description	N

Table 17: VariationStatusCode table definition

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix 4: Table Relations

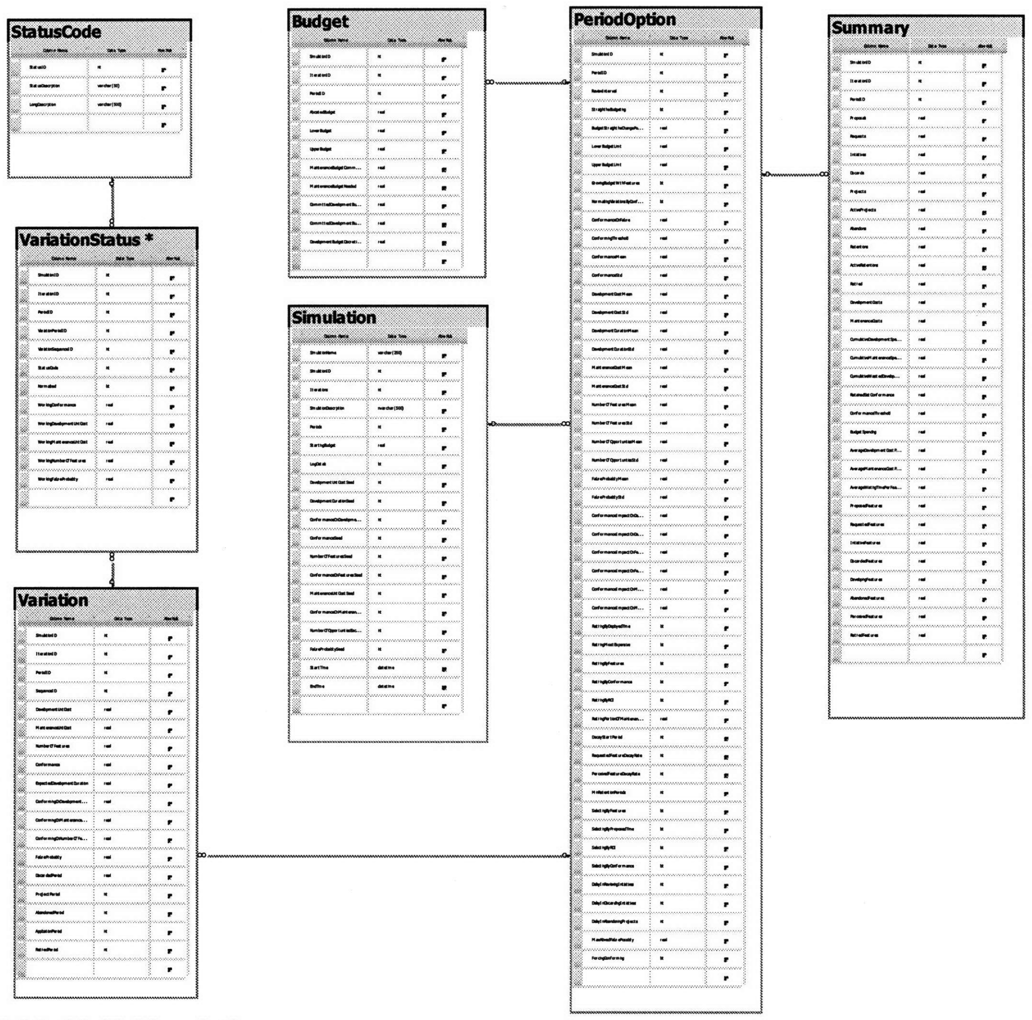


Table 18: Table relations

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix 5: Simulation Application Class Design Diagram using Unified Modeling Language (UML)

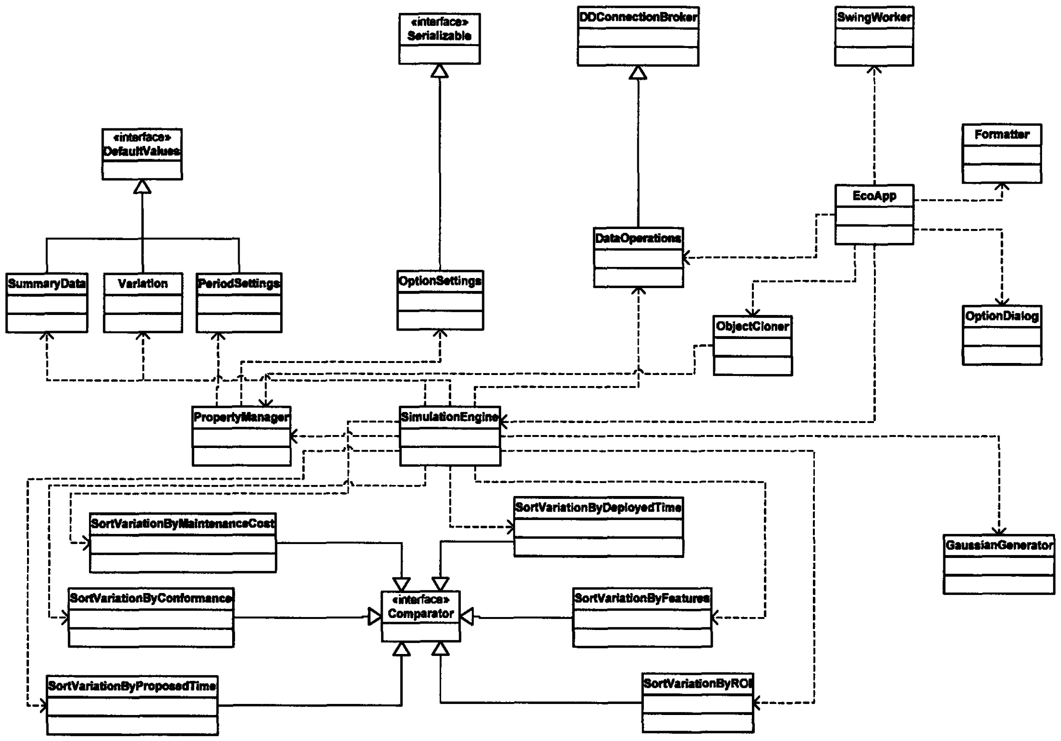


Table 19: Simulation application Unified Molding Language (UML) class diagram

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix 6: Sample code

Because of the space constraint, only Java source codes related to the key functions for the simulation engine component are shown below.

```
/*
 * SimulationEngine.java
 *
 * Created on July 30, 2006, 11:34 PM
 *
 * @author Bin Zhou
 * @version 1.0
 *
 * IT Ecosystem System Dynamics Simulation Application SimulationEngine Class
 */

package edu.mit.ecosystem.process;

import java.util.*;

import edu.mit.ecosystem.container.*;
import edu.mit.ecosystem.generator.*;
import edu.mit.ecosystem.tool.*;
import edu.mit.ecosystem.dataaccess.*;
import edu.mit.ecosystem.constants.*;

/**
 * This class implements the simulation engine that performs the simulation scenarios using
 * parameters passed in from the GUI
 *
 */
public class SimulationEngine
{
   // Current simulation ID
   private long m_Simulation = -1;

   // Current iteration ID
   private int m_Iteration = -1;

   // current period ID
   private int m_Period = -1;

   // configuration options for the simulation
   private OptionSettings m_OptionSettings = null;

   // period configurations
   private PeriodSettings m_PeriodSettings = null;

   // summary data
   private SummaryData m_SummaryData = null;

   // Proposal Set data
```

```
private List <Variation> proposalSet    = new ArrayList <Variation> (150);

// Request Set data
private List <Variation> requestSet     = new ArrayList <Variation> (150);

// Initiative Set data
private List <Variation> initiativeSet  = new ArrayList <Variation> (150);

// Discarded Set data
private List <Variation> discardedSet   = new ArrayList <Variation> (150);

// Development Set data
private List <Variation> developmentSet = new ArrayList <Variation> (150);

// Abandoned Set data
private List <Variation> abandonedSet   = new ArrayList <Variation> (150);

// Retained Set data
private List <Variation> retainedSet    = new ArrayList <Variation> (150);

// RetiredSet Set data
private List <Variation> retiredSet     = new ArrayList <Variation> (150);

// List contains simulation Setting data for every period
private List <OptionSettings> optionsSet   = new ArrayList <OptionSettings> (150);

// List contains setting data for every period
private List <PeriodSettings>  periodSet   = new ArrayList <PeriodSettings> (150);

// List of summary data -- use LinkedList to ensure the order of the data populated
private List <SummaryData> summarySet   = new ArrayList <SummaryData> (150);

// file used to inspect the sorted order on selection and retirement regime data
private ResultOutput sortedVariationCandidateOutput = null;        // sorted variations
private ResultOutput sortedRetiredCandidateOutput = null;          // sorted retirement

// random number generators
private GaussianGenerator numberOfOpportunitiesGenerator = null;
private GaussianGenerator developmentCostGenerator = null;
private GaussianGenerator conformingOnDevelopmentCostGenerator = null;
private GaussianGenerator conformanceGenerator = null;
private GaussianGenerator featuresGenerator = null;
private GaussianGenerator comformingOnFeaturesGenerator = null;
private GaussianGenerator maintenanceCostGenerator = null;
private GaussianGenerator conformingOnMaintenanceCostGenerator = null;
private GaussianGenerator durationGenerator = null;
private GaussianGenerator failurePossibilityGenerator = null;

// budgets
private double lowerBudgetAmt = 0.0D;
private double upperBudgetAmt = 0.0D;

private double lastPeriodBudget = 0.0D;
private double currentPeriodBudget = 0.0D;

private double maintenanceBudgetNeeded = 0.0D;
private double maintenanceBudgetCommitted = 0.0D;

private double committedDevelopmentBudgetNeeded = 0.0D;
private double committedDevelopmentBudgetCommitted = 0.0D;

private double developmentBudgetDiscretionary  = 0.0D;

// features
private double numberOfApplicationFeatures = 0.0D;
private double numberOfApplicationFeaturesMaintained = 0.0D;

private double numberOfDevelopmentFeatures = 0.0D;
private double numberOfDevelopmentFeaturesDeveloping = 0.0D;


/** Creates a new instance of SimulationEngine */
public SimulationEngine()
{
   System.out.println("Loading SimulationEngine ...");
}


/**
 * Cleans up the data stroes for the next iteration. This method will be run after each period
 * has been run
 */
public void clean()
{
   System.out.println("Clean up SimulationEngine data ...");

   m_Period = -1;

   m_OptionSettings = null;

   m_PeriodSettings = null;

   m_SummaryData = new SummaryData();

   // Proposal Set data
   proposalSet.clear();

   // Request Set data
   requestSet.clear();

   // Initiative Set data
   initiativeSet.clear();

   // Discarded Set data
   discardedSet.clear();

   // Development Set data
   developmentSet.clear();

   // Abandoned Set data
   abandonedSet.clear();
```

```
        // Retained Set data
        retainedSet.clear();

        // RetiredSet Set data
        retiredSet.clear();

        // setting data
        optionsSet.clear();

        // values derived from the options
        periodSet.clear();

        // Summary data -- use LinkedList to ensure the order of the data populated
        summarySet.clear();
    }


    /** Loads initial simulation settings from property file, this will only be executed
     *  once at the start of the simulation engine.
     *
     * @param OptionSettngs
     */
    public void start(OptionSettings optionsSettings)
    {
        System.out.println("Starting simulation engine...");

        this.m_OptionSettings = optionsSettings;

        // open inspection files for later use
        sortedVariationCandidateOutput = new ResultOutput("SortedInitiativesCandidateData.txt");
        sortedRetiredCandidateOutput = new ResultOutput("SortedRetiredCandidateData.txt");

        /* initialize the random number generators with the seed specified from the property file */
        // number of opportunities for a period
        numberOfOpportunitiesGenerator
              = new GaussianGenerator(m_OptionSettings.getNumberOfOpportunitiesSeed());

        // development cost
        developmentCostGenerator
              = new GaussianGenerator(m_OptionSettings.getDevelopmentCostSeed());

        // conforming impact on development cost
        conformingOnDevelopmentCostGenerator
              = new
GaussianGenerator(m_OptionSettings.getConformanceImpactOnMaintenanceCostSeed());

        // conformance
        conformanceGenerator
              = new GaussianGenerator(m_OptionSettings.getConformanceSeed());

        // features
        featuresGenerator
              = new GaussianGenerator(m_OptionSettings.getNumberOfFeaturesSeed());

        // comforming impact on number of features
        comformingOnFeaturesGenerator
```

```
            = new
GaussianGenerator(m_OptionSettings.getConformanceImpactOnFeaturesSeed());

        // maintenance cost
        maintenanceCostGenerator
            = new GaussianGenerator(m_OptionSettings.getMaintenanceCostSeed());

        // comforming on maintenance cost
        conformingOnMaintenanceCostGenerator
            = new
GaussianGenerator(m_OptionSettings.getConformanceImpactOnMaintenanceCostSeed());

        // development durations
        durationGenerator
            = new GaussianGenerator(m_OptionSettings.getDevelopmentDurationSeed());

        // failure threshold
        failurePossibilityGenerator
            = new GaussianGenerator(m_OptionSettings.getFailurePossibilitySeed());
    }


    /**
     * Work flow method that executes once for every period
     */
    public void run(long simulation, int iteration, int period, OptionSettings optionsSettings)
    {
        // settings
        this.m_Simulation = simulation;
        this.m_Iteration = iteration;
        this.m_Period = period;
        this.m_OptionSettings = optionsSettings;

        // budgets
        this.currentPeriodBudget = 0.0D;

        this.maintenanceBudgetCommitted = 0.0D;
        this.maintenanceBudgetNeeded = 0.0D;

        this.committedDevelopmentBudgetCommitted = 0.0D;
        this.committedDevelopmentBudgetNeeded = 0.0D;

        this.developmentBudgetDiscretionary = 0.0D;

        // features
        this.numberOfApplicationFeatures = 0.0D;
        this.numberOfApplicationFeaturesMaintained = 0.0D;

        this.numberOfDevelopmentFeatures = 0.0D;
        this.numberOfDevelopmentFeaturesDeveloping = 0.0D;

        // keep the options data
        optionsSet.add(m_OptionSettings);

        // find the total IT budget for the prriod
        getPeriodBudget();
        // create variations for the period
        generateVariations();

        // normalize variations
        normalize();

        // move finished projects into the ApplicationSet
        processFinishedProjects();

        // process retention, including retiring old applications and recalculate budget available
        processRetentions();

        // select which projects to work on for the period and abandon failed projects
        processProjects();

        // select new projects and discard initiatives
        processInitiatives();

        // decay old features in the RequestSet and the ApplicationSet
        decayFeatures();

        // generate summary data for reporting and analysis
        processSummary();

    }


    // Last method to run before the simulation is finished
    public void stop()
    {
        System.out.println("Finished running simulation engine!");
    }


    /**
     * Method to calculate budget allocation for the up-coming period according to the budget
regime
     */
    private void getPeriodBudget()
    {
        // fill in with base budget for the first period
        if (this.m_Period == 0)
        {
            this.lastPeriodBudget = this.m_OptionSettings.getBaselineBudget();
        }

        // assume fixed budget first if no budget growth regime applied
        this.currentPeriodBudget = this.lastPeriodBudget;

        // if we do straightline budgeting
        if (m_OptionSettings.isStraightlineBudgeting())
        {
            if (this.m_Period > 0)      // at least we have 2 periods
            {
                double incremental = m_OptionSettings.getBudgetStraightlineChangePortion();
                currentPeriodBudget = lastPeriodBudget * (1 + incremental / 100.0);
            }
        }

        // if growing budget by increase rate of features
        if(m_OptionSettings.isGrowingBudgetWithFeatures())
        {
            if (this.m_Period >= 2)
            {
                double featuresCurrentMinus2 = 0.0D;
                double featuresCurrentMinus1 = 0.0D;

                for (int i = 0; i < this.retainedSet.size(); i++)
                {
                    Variation retained = this.retainedSet.get(i);
                    int deployedTime = retained.getTimeDeployed();

                    if (deployedTime <= this.m_Period - 1 && deployedTime > 0)
                    {
                        featuresCurrentMinus1 += retained.getWorkingNumberOfFeatures();

                        if (deployedTime <= this.m_Period - 2)
                        {
                            featuresCurrentMinus2 += retained.getWorkingNumberOfFeatures();
                        }
                    }
                }

                if (featuresCurrentMinus2 > 0)
                {
                    this.currentPeriodBudget = this.lastPeriodBudget * ((double) featuresCurrentMinus1 /
featuresCurrentMinus2);
                }
            }
        }

        // get the lower and upper bound of budget allocations
        this.lowerBudgetAmt = m_OptionSettings.getLowerBudgetLimit() * this.currentPeriodBudget
/ 100.0;
        this.upperBudgetAmt = m_OptionSettings.getUpperBudgetLimit() * this.currentPeriodBudget
/ 100.0;

        // prepare for the next period budget allocation
        this.lastPeriodBudget = this.currentPeriodBudget;
    }


    /**
     * Method to create the raw variation data for one period
     */
    private void generateVariations()
    {
        System.out.println("generateVariations() current period: " + m_Period);

        // find out how many opportunities to propose for the period
    double [] numberOfOpportunitiesSet = numberOfOpportunitiesGenerator.getGaussianData(
            1,
            m_OptionSettings.getNumberOfOpportunitiesMean(),
            m_OptionSettings.getNumberOfOpportunitiesStd());

    // at least one initiative for each period
    int numberOfOpportunities = (int) (numberOfOpportunitiesSet[0] + 1);
    if (numberOfOpportunities < 1)
    {
        numberOfOpportunities = 1;
    }

    // now we know how many opportunities for the period, so we can create data fields for each
    // of the opportunity

    // get random data here and put them into array structure for later convenient use
    double [] developmentCosts = developmentCostGenerator.getGaussianData(
            numberOfOpportunities,
            m_OptionSettings.getDevelopmentCostMean(),
            m_OptionSettings.getDevelopmentCostStd());

    double [] developmentDurations = durationGenerator.getGaussianData(
            numberOfOpportunities,
            m_OptionSettings.getDevelopmentDurationMean(),
            m_OptionSettings.getDevelopmentDurationStd());

    double [] conformingOnDevelopmentCost =
conformingOnDevelopmentCostGenerator.getGaussianData(
            numberOfOpportunities,
            m_OptionSettings.getConformanceImpactOnDevelopmentCostMean(),
            m_OptionSettings.getConformanceImpactOnDevelopmentCostStd());

    double [] conformances = conformanceGenerator.getGaussianData(
            numberOfOpportunities,
            m_OptionSettings.getConformanceMean(),
            m_OptionSettings.getConformanceStd());

    double [] features = featuresGenerator.getGaussianData(
            numberOfOpportunities,
            m_OptionSettings.getNumberOfFeaturesMean(),
            m_OptionSettings.getNumberOfFeaturesStd());

    double [] conformingOnFeatures = comformingOnFeaturesGenerator.getGaussianData(
            numberOfOpportunities,
            m_OptionSettings.getConformanceImpactOnFeaturesMean(),
            m_OptionSettings.getConformanceImpactOnFeaturesStd());

    double [] maintenanceCosts = maintenanceCostGenerator.getGaussianData(
            numberOfOpportunities,
            m_OptionSettings.getMaintenanceCostMean(),
            m_OptionSettings.getMaintenanceCostStd());

    double [] conformingOnMaintenanceCosts =
conformingOnMaintenanceCostGenerator.getGaussianData(
            numberOfOpportunities,
            m_OptionSettings.getConformanceImpactOnMaintenanceCostMean(),
            m_OptionSettings.getConformanceImpactOnMaintenanceCostStd());

        double [] failurePossibilities = failurePossibilityGenerator.getGaussianData(
            numberOfOpportunities,
            m_OptionSettings.getFailurePossibilityMean(),
            m_OptionSettings.getFailurePossibilityStd());

        // set the raw period data
        this.m_PeriodSettings = new PeriodSettings();
        this.m_PeriodSettings.setCurrentPeriodNumber(this.m_Period);
        this.m_PeriodSettings.setNumberOfOpportunities(numberOfOpportunities);
        this.m_PeriodSettings.setDevelopmentCostSet(developmentCosts);
        this.m_PeriodSettings.setDevelopmentDurationSet(developmentDurations);

this.m_PeriodSettings.setConformanceOnDevelopmentCostSet(conformingOnDevelopmentCost);
        this.m_PeriodSettings.setConformanceSet(conformances);
        this.m_PeriodSettings.setConformanceOnFeaturesSet(conformingOnFeatures);
        this.m_PeriodSettings.setMaintenanceCostSet(maintenanceCosts);

this.m_PeriodSettings.setConformanceOnMaintenanceCostSet(conformingOnMaintenanceCosts);
        this.m_PeriodSettings.setNumberOfFeatureSet(features);
        this.m_PeriodSettings.setFailurePossibilitySet(failurePossibilities);

        // populate variation data into varation data containter
        for (int i = 0; i < numberOfOpportunities; i++)
        {
          Variation request = new Variation();

          request.setPeriod(this.m_Period);
          request.setSequence(i);
          request.setConformance(Math.max(0, Math.min(1, conformances[i])));
          request.setDevelopmentUnitCost(Math.max(0, developmentCosts[i]));
          request.setMaintenanceUnitCost(Math.max(0, maintenanceCosts[i]));

          request.setConformingOnDevelopmentUnitCost(Math.max(0,
conformingOnDevelopmentCost[i]));
          request.setConformingOnMaintenanceUnitCost(Math.max(0,
conformingOnMaintenanceCosts[i]));
          request.setConformingOnNumberOfFeatures(Math.max(0, conformingOnFeatures[i]));

          // a project will have at least one period to develop,
          request.setExpectedDevelopmentDuration(Math.max(1, ((int)developmentDurations[i])));

          // At least one exptected number of features
          request.setNumberOfFeatures(Math.max(1, features[i]));

          // base failure probability
          request.setFailureProbability(Math.max(0, failurePossibilities[i]));

          request.setTimeProposed(this.m_Period);
          request.setTimeDiscarded(-1);
          request.setTimeSelected(-1);
          request.setTimeAbandoned(-1);
          request.setTimeDeployed(-1);
          request.setTimeRetired(-1);
          request.setPeriodsInDevelopment(0);
        request.setDevelopmentSpending(0);
        request.setMaintenanceSpending(0);
        request.setMaintained(false);
        request.setDeveloping(false);

        // proposalSet contains all proposals business has ever asked for
        proposalSet.add(request);

        // reqeustSet is the set with all requests (=propsoalSet - retiredSet),
        // When an Application retires, it will be out of the reqeustSet, but still in the proposalSet
        // This one will be used to calculation the business satisfaction Features in Application
        // over Features in this requestSet
        requestSet.add(request);

        // initialSet is the one for working in progress variation
        initiativeSet.add(request);

        //  DataOperations dbOperations = DataOperations.getInstance();
        String variationSql = createVariationSql(request);

        try
        {
           DataOperations.getInstance().executeSQL(variationSql, false);
        }
        catch (Exception e)
        {
           System.out.println("Error in Variation sql: " + e.getMessage());
           System.exit(-5);
        }
     }
  }


  /**
   * Insert variaton data into the database table for later analysis
   */
  private String createVariationSql(Variation variation)
  {
     String sql = " INSERT INTO Variation (SimulationID, IterationID, "
           + " PeriodID, SequenceID, "
           + " DevelopmentUnitCost, MaintenanceUnitCost, "
           + " NumberOfFeatures, Conformance, "
           + " ExpectedDevelopmentDuration, ConformingOnDevelopmentUnitCost, "
           + " ConformingOnMaintenanceUnitCost, ConformingOnNumberOfFeatures, "
           + " FailureProbability, DiscardedPeriod,  "
           + " ProjectPeriod, AbandonedPeriod, "
           + " ApplicationPeriod, RetiredPeriod) "
           + " VALUES ("
           + m_Simulation + "," +  m_Iteration + ", "
           + variation.getPeriod() + ", " + variation.getSequence() + ", "
           + variation.getDevelopmentUnitCost() + ", " + variation.getMaintenanceUnitCost() + ", "
           + variation.getNumberOfFeatures() + ", " + variation.getConformance() + ", "
           + variation.getExpectedDevelopmentDuration() + ", "  +
variation.getConformingOnDevelopmentUnitCost() + ", "
           + variation.getConformingOnMaintenanceUnitCost() + ", " +
variation.getConformingOnNumberOfFeatures() + ", "
```

```
            + variation.getFailureProbability() + ", " + DefaultValues.STATUS_UNKNOWN + ", "
            + DefaultValues.STATUS_UNKNOWN + ", " + DefaultValues.STATUS_UNKNOWN
            + ", " + DefaultValues.STATUS_UNKNOWN + ", " +
DefaultValues.STATUS_UNKNOWN + ") ";
    //   System.out.print("Variation: " + sql);
    return sql;
  }


  /**
   * Normalization process on the sets that are involved
   */
  private void normalize()
  {
    normalizeDataSet(this.proposalSet);
    normalizeDataSet(requestSet);
    normalizeDataSet(initiativeSet);
    normalizeDataSet(discardedSet);
    normalizeDataSet(developmentSet);
    normalizeDataSet(abandonedSet);
    normalizeDataSet(retainedSet);
    normalizeDataSet(retiredSet);
  }


  /**
   *  Normalization on a list of variation data
   */
  private void normalizeDataSet(List <Variation> dataSet)
  {
    for (int dataCount = 0; dataCount < dataSet.size(); dataCount++)
    {
      Variation data = dataSet.get(dataCount);

      // determine which set is the source to set up the working data
      if (m_OptionSettings.isNormalizingProjectsWithConformance())        // use normalized
data
      {
        data.setNormalized(true);

        data.setWorkingDevelopmentUnitCost(Math.max(0,
              data.getDevelopmentUnitCost() * data.getConformingOnDevelopmentUnitCost()));

        // maintenance cost will decrease for more conforming variation after the normalization
process
        data.setWorkingMaintenanceUnitCost(data.getMaintenanceUnitCost()
        * (1 - Math.max(0, m_OptionSettings.getConformingThreshold() -
data.getConformance())));

        // the higher the data conformance, the more features
        data.setWorkingNumberOfFeatures(Math.max(0, data.getNumberOfFeatures()
        * Math.max(0, data.getConformance() / m_OptionSettings.getConformingThreshold())
        * Math.max(0, data.getConformingOnNumberOfFeatures())));

        // failure probability
        data.setWorkingFailureProbability(Math.max(0,
                    data.getFailureProbability() + m_OptionSettings.getConformanceOnFailure()
                    * (1 - data.getConformance())));

                data.setWorkingConformance(m_OptionSettings.getConformingThreshold());
            }
            else        // use raw data if there is no need for normalization
            {
                data.setNormalized(false);
                data.setWorkingConformance(Math.max(0, data.getConformance()));
                data.setWorkingDevelopmentUnitCost(Math.max(0, data.getDevelopmentUnitCost()));
                data.setWorkingMaintenanceUnitCost(Math.max(0, data.getMaintenanceUnitCost()));
                data.setWorkingNumberOfFeatures(Math.max(0, data.getNumberOfFeatures()));
                data.setWorkingFailureProbability(Math.max(0, data.getFailureProbability()));
            }
        }
    }


    /**
     * Process finished projects and move them into the Application Set
     */
    private void processFinishedProjects()
    {
        for (int i = this.developmentSet.size() - 1; i >= 0; i--)
        {
            Variation project = this.developmentSet.get(i);

            // this is a finished project and needs to be moved into RetainedSet
            if (project.getPeriodsInDevelopment() >= project.getExpectedDevelopmentDuration())
            {
                project.setTimeDeployed(this.m_Period);
                project.setMaintained(false);
                project.setDeveloping(false);

                retainedSet.add(project);
                developmentSet.remove(i);

                // update the database on the variation status
                DataOperations.getInstance().updateVariationStatus("ApplicationPeriod", project,
this.m_Period,
                    this.m_Iteration, this.m_Simulation);
            }
        }
    }


    /**
     * Retention process: retire applications from the retainedSet and also
     * remove the retired variation from the requestSet so that we have accurate number
     * of requested features, add the retired vairation into the retired data set
     */
    private void processRetentions()
    {
        // System.out.println("processRetentions() current period: " + m_Period);
        this.adjustPeriodMaintenanceBudget();
    double retireMaintenanceCostPct =
m_OptionSettings.getRetiringPortionOfMaintenanceCost();

    // need to retire something
    if (retireMaintenanceCostPct > 0.0)
    {
      // using the retirement regime, retire at least to the retireMaintenanceCostPct
      // if we have over run the budget
      double minReducedCostAmt = this.maintenanceBudgetNeeded *
retireMaintenanceCostPct / 100.0;

      LinkedList <Variation> sortedSet = new LinkedList <Variation> (this.retainedSet);
      getRetirementSortedList(sortedSet);

      // print out the sorted variation set for debug purpose
      sortedRetiredCandidateOutput.printRequest("\n" + Variation.showHeader() + "\n");
      for (int i = 0; i < sortedSet.size(); i++)
      {
        Variation sortedVariation = sortedSet.get(i);
        sortedRetiredCandidateOutput.printRequest(sortedVariation.toString() + "\n");
      }

      // now remove an retired application from the Application Set and send it into
      // the Retained Set. We also remove it from the Request Set
      for (int i = sortedSet.size() - 1; i >= 0 && minReducedCostAmt > 0; i--)
      {
        Variation sorted = sortedSet.get(i);

        if (m_Period - sorted.getTimeDeployed() > m_OptionSettings.getMinRetentionPeriods())
        {
          for (int j = this.retainedSet.size() - 1; j >= 0 ; j--)
          {
            Variation application = this.retainedSet.get(j);

            if (sorted.getPeriod() == application.getPeriod() && sorted.getSequence() ==
application.getSequence())
            {
              application.setTimeRetired(this.m_Period);
              application.setMaintained(false);
              this.retainedSet.remove(j);

              // add this into the retired set
              this.retiredSet.add(sorted);

              DataOperations.getInstance().updateVariationStatus("RetiredPeriod", sorted,
this.m_Period,
                  this.m_Iteration, this.m_Simulation);

              // remove from the reqeustSet
              for (int k = requestSet.size() - 1; k >= 0; k--)
              {
                Variation request = this.requestSet.get(k);

                if (request.getPeriod() == sorted.getPeriod() && request.getSequence() ==
sorted.getSequence())
                {
                        this.requestSet.remove(k);
                        break;
                    }
                }

                minReducedCostAmt -= sorted.getWorkingMaintenanceUnitCost();
                break;
            }
        }
    }
}
}

// now set maintained flags all to false
for (int i = 0; i < this.retainedSet.size() - 1; i++)
{
    this.retainedSet.get(i).setMaintained(false);
}

// set flag to true for those that we can afford to keep maintaining
this.adjustPeriodMaintenanceBudget();

LinkedList <Variation> sortedSet = new LinkedList <Variation> (this.retainedSet);
getRetirementSortedList(sortedSet);

// adjust the maintenance budget
for (int i = 0; i < sortedSet.size() && this.maintenanceBudgetCommitted <=
this.lowerBudgetAmt; i++)
{
    Variation sorted = sortedSet.get(i);

    if (this.maintenanceBudgetCommitted + sorted.getWorkingMaintenanceUnitCost() <=
this.upperBudgetAmt)
    {
        for (int j = this.retainedSet.size() - 1; j >= 0 ; j--)
        {
            Variation application = this.retainedSet.get(j);

            if (application.getPeriod() == sorted.getPeriod() && application.getSequence() ==
sorted.getSequence())
            {
                application.setMaintained(true);

                // cost will increase by Maintenance Unit Cost for the application
                application.setMaintenanceSpending(application.getMaintenanceSpending() +
application.getWorkingMaintenanceUnitCost());
                break;
            }
        }

        this.maintenanceBudgetCommitted += sorted.getWorkingMaintenanceUnitCost();
    }
}

// we recalculate the committed budget needed and allocated to reflected the retention
regime impact
```

```
    this.adjustPeriodMaintenanceBudget();
}


/**
 * Sort the Retained Set according to the Retention Regime
 */
private void getRetirementSortedList(List <Variation> sortedSet)
{
    // by deployed time, earlier will be earlier
    if (this.m_OptionSettings.isRetiringByDeployedTime())
    {
        Collections.sort(sortedSet, new SortVariationByDeployedTime());
        Collections.reverse(sortedSet);
    }

    // by conformance, lowest will be retired first
    if (this.m_OptionSettings.isRetiringByConformance())
    {
        Collections.sort(sortedSet, new SortVariationsByConformance());
        Collections.reverse(sortedSet);
    }

    // by maintenance cost, highest will be retired first
    if (this.m_OptionSettings.isRetiringMostExpensive())
    {
        Collections.sort(sortedSet, new SortVariationsByMaintenanceCost());
    }

    // by features, highest will be retired last
    if (this.m_OptionSettings.isRetiringByFeatures())
    {
        Collections.sort(sortedSet, new SortVariationsByFeatures());
        Collections.reverse(sortedSet);
    }

    // by ROI, lowest will be retired first
    if (this.m_OptionSettings.isRetiringByROI())
    {
        Collections.sort(sortedSet, new SortVariationsByROI());
        Collections.reverse(sortedSet);
    }

}


/**
 * Process existing committed projects for a period
 */
private void processProjects()
{
    //    System.out.println("processProjects() current period: " + m_Period);
    this.committedDevelopmentBudgetCommitted = 0.0D;
    this.committedDevelopmentBudgetNeeded = 0.0D;

    // abandon failed projects
```

```
      for (int i = this.developmentSet.size() - 1; i >= 0; i--)
      {
         Variation project = this.developmentSet.get(i);

         // fail some project
         if (this.m_Period - project.getTimeSelected() >
this.m_OptionSettings.getDelayInAbandoningProjects()
         && project.getWorkingFailureProbability() >
this.m_OptionSettings.getMaxAllowedFailurePossibility())
         {
            project.setTimeAbandoned(this.m_Period);
            project.setDeveloping(false);
            abandonedSet.add(project);
            developmentSet.remove(i);

            // since project was never delivered, we think all spending on it was wasted
            //          System.out.println("project wasted: " + project.getDevelopmentSpending());

this.m_SummaryData.setWastedDevelopmentSpending(this.m_SummaryData.getWastedDevelo
pmentSpending()
            + project.getDevelopmentSpending());

            //          System.out.println("getWastedDevelopmentSpending: " +
m_SummaryData.getWastedDevelopmentSpending());

            DataOperations.getInstance().updateVariationStatus("AbandonedPeriod", project,
this.m_Period,
                  this.m_Iteration, this.m_Simulation);
         }
      }

      // set all project to be non-active
      for (int i = 0; i < this.developmentSet.size() - 1; i++)
      {
         this.developmentSet.get(i).setDeveloping(false);
      }

      LinkedList <Variation> sortedSet = new LinkedList <Variation> (this.developmentSet);
      getInitiativeSortedList(sortedSet);

      // now enable the projects that we can afford
      for (int i = sortedSet.size() - 1; i >= 0; i--)
      {
         Variation sorted = sortedSet.get(i);

         if (this.maintenanceBudgetNeeded + this.committedDevelopmentBudgetCommitted
               + sorted.getWorkingDevelopmentUnitCost() <= this.upperBudgetAmt)
         {
            // if we can afford developing this project

            for (int j = 0; j < this.developmentSet.size(); j++)
            {
               Variation project = this.developmentSet.get(j);

               if (sorted.getPeriod() == project.getPeriod() && sorted.getSequence() ==
project.getSequence())
```

```
                {
                   project.setPeriodsInDevelopment(project.getPeriodsInDevelopment() + 1);
                   project.setDeveloping(true);

                   // development cost will increase by Working Development Unit Cost for each of
the active project
                   project.setDevelopmentSpending(project.getDevelopmentSpending() +
project.getWorkingDevelopmentUnitCost());

                   this.committedDevelopmentBudgetCommitted +=
project.getWorkingDevelopmentUnitCost();
                }
              }
           }

           this.committedDevelopmentBudgetNeeded += sorted.getWorkingDevelopmentUnitCost();
        }
     }


     /**
      * this is the selection process to select some initiatives into the Development Set
      * as new projects or discard initiatives into discardedSet for a period
      */
     private void processInitiatives()
     {
//       System.out.println("processInitiatives() current period: " + m_Period);

        this.developmentBudgetDiscretionary = 0;

        // now we first find out the discarded initiatives
        for (int i = this.initiativeSet.size() - 1; i >= 0; i--)
        {
           Variation initiative = this.initiativeSet.get(i);

           // If it has been in the InitiativeSet for too long, move it into the DiscardedSet
           if (this.m_Period - initiative.getTimeProposed() >
this.m_OptionSettings.getDelayInDiscardingInitiatives())
           {
              //         System.out.println("Initiative " + initiative.getPeriod() + "_" +
initiative.getSequence()
              //         + " will be moved into DiscardedSet");

              // populate the Time Discarded
              initiative.setTimeDiscarded(this.m_Period);

              // movve the Initiative into the DiscardedSet
              discardedSet.add(initiative);

              // remove this one from the Initiative Set
              initiativeSet.remove(i);

              DataOperations.getInstance().updateVariationStatus("DiscardedPeriod", initiative,
this.m_Period,
                   this.m_Iteration, this.m_Simulation);
           }
```

```
    }

    double committedBudget = this.maintenanceBudgetNeeded +
this.committedDevelopmentBudgetNeeded;

    // if there is no discretionary budget existing, then stop here
    if (committedBudget > this.lowerBudgetAmt)
      return;

    // if we do not spend at least up to the lowerBudgetLimit
    // then we have some for discretionary selection on new development for the period
    double budgetAmt = this.currentPeriodBudget;

    // find out how we do the selection process
    List <Variation> sortedSet = new ArrayList <Variation> (this.initiativeSet);
    getInitiativeSortedList(sortedSet);

    // print out the sorted variation set
    sortedVariationCandidateOutput.printRequest("\n" + Variation.showHeader() + "\n");
    for (int j = 0; j < sortedSet.size(); j++)
    {
      Variation sortedVariation = sortedSet.get(j);
      sortedVariationCandidateOutput.printRequest(sortedVariation.toString() + "\n");
    }

    for (int j = sortedSet.size() - 1; j >= 0 && committedBudget < this.lowerBudgetAmt; j--)
    {
      Variation sorted = sortedSet.get(j);

      // only do the selection process on the Initiatives that have been proposed
      // longer than the DELAY_IN_REVIEW_THRESHOLD
      if (this.m_Period - sorted.getTimeProposed() <
this.m_OptionSettings.getDelayInReviewingInitiatives())
      {
        // System.out.println("Initiative #" + sortedVariation.getPeriod() + "_" +
sortedVariation.getSequence() + " will be reviewed");
        // System.out.println("Its proposed time is: " + sortedVariation.getTimeProposed());
        // System.out.println("Its conformance value is: " +
sortedVariation.getWorkingConformance());

        continue;
      }

      // if does not pass the conformanceThreshold, then pass this one
      if (this.m_OptionSettings.isForcingConforming()
      && sorted.getWorkingConformance() < this.m_OptionSettings.getConformingThreshold())
      {
        continue;
      }

      // now we try to check if the first variation in the line can become a project
      if (committedBudget + sorted.getWorkingDevelopmentUnitCost() <= upperBudgetAmt)
      {
        // add this initiative into the development set
        sorted.setTimeSelected(this.m_Period);
        sorted.setPeriodsInDevelopment(0);
```

```
            sorted.setDeveloping(true);
            developmentSet.add(sorted);

            DataOperations.getInstance().updateVariationStatus("ProjectPeriod", sorted,
this.m_Period,
                    this.m_Iteration, this.m_Simulation);

            committedBudget += sorted.getWorkingDevelopmentUnitCost();

            this.developmentBudgetDiscretionary += sorted.getWorkingDevelopmentUnitCost();

            // delete it from the initiative set
            for (int k = this.initiativeSet.size() - 1; k >= 0 ; k--)
            {
                Variation initiativeInSet = this.initiativeSet.get(k);

                if (initiativeInSet.getPeriod() == sorted.getPeriod()
                && initiativeInSet.getSequence() == sorted.getSequence())
                {
                    // remove the initiative from the initiative set
                    this.initiativeSet.remove(k);

                    break;
                }
            }
        }
    }

    String sql = " INSERT INTO Budget(SimulationID, IterationID, PeriodID, AllocatedBudget, "
            + " LowerBudget, UpperBudget, MaintenanceBudgetCommitted,
MaintenanceBudgetNeeded, "
            + " CommittedDevelopmentBudgetCommitted, CommittedDevelopmentBudgetNeeded,
"
            + "DevelopmentBudgetDiscretionary) VALUES ("
            + this.m_Simulation + ", " + this.m_Iteration + ", " + this.m_Period + ", "
            + this.currentPeriodBudget + ", " + lowerBudgetAmt + ", " + upperBudgetAmt + ", "
            + this.maintenanceBudgetCommitted + ", " + this.maintenanceBudgetNeeded + ", "
            + this.committedDevelopmentBudgetCommitted + ", " +
this.committedDevelopmentBudgetNeeded + ", "
            + this.developmentBudgetDiscretionary
            + ") ";

    try
    {
        DataOperations.getInstance().executeSQL(sql, false);
    }
    catch (Exception e)
    {
        System.exit(-9);
    }
  }


  /**
   * Sort initiatives according to Selection Regime
   */
   private void getInitiativeSortedList(List <Variation> sortedSet)
   {
      // if selecting by proposed time, the earliest will become project first
      if (m_OptionSettings.isSelectingByProposedTime())
      {
         Collections.sort(sortedSet, new SortVariationsByProposedTime());
         Collections.reverse(sortedSet);
      }

      // filtering by conformance factor, the highest will become project first
      if (m_OptionSettings.isSelectingByConformance())
      {
         Collections.sort(sortedSet, new SortVariationsByConformance());
         //      Collections.reverse(sortedSet);
      }

      // if selecting by number of features, the one with most number of features will become
project first
      if (m_OptionSettings.isSelectingByFeatures())
      {
         Collections.sort(sortedSet, new SortVariationsByFeatures());
      }

      // if selecting by ROI (features/maintenance cost), the highest one will become project first
      if (m_OptionSettings.isSelectingByROI())
      {
         Collections.sort(sortedSet, new SortVariationsByROI());
      }

   }


   /**
    * Decay Featrues sets
    */
   private void decayFeatures()
   {
      decaySet(this.requestSet, m_OptionSettings.getRequestedFeatureDecayRate());
      decaySet(this.retainedSet, Math.max(0,
            m_OptionSettings.getPerceivedFeatureDecayRate() -
m_OptionSettings.getRequestedFeatureDecayRate()));
   }


   /*
    * Decay of number of features after a certain period for a variation set
    */
   private void decaySet(List <Variation> dataSet, double discount)
   {
      for (int dataCount = 0; dataCount < dataSet.size(); dataCount++)
      {
         Variation data = dataSet.get(dataCount);

         if (this.m_Period - data.getPeriod() >= this.m_OptionSettings.getDecayStartPeriod())
         {
            data.setNumberOfFeatures(data.getNumberOfFeatures() * (1 - ((double) discount) /
100.0 ));
          }
        }
    }


    /**
     * Method to populate summary data
     */
    private void processSummary()
    {
        this.periodSet.add(this.m_PeriodSettings);

        SummaryData summary = new SummaryData();

        summary.setPeriod(this.m_Period);
        summary.setProposals(this.proposalSet.size());
        summary.setRequests(this.requestSet.size());
        summary.setInitiatives(this.initiativeSet.size());
        summary.setDiscarded(this.discardedSet.size());
        summary.setProjects(this.developmentSet.size());
        summary.setActiveProjects(this.getNumberOfActiveProjects());
        summary.setAbandoned(this.abandonedSet.size());
        summary.setRetained(this.retainedSet.size());
        summary.setActiveApplications(this.getNumberOfActiveApplications());
        summary.setRetired(this.retiredSet.size());

        // period costs
        summary.setDevelopmentCost(this.committedDevelopmentBudgetCommitted +
this.developmentBudgetDiscretionary);
        summary.setMaintenanceCost(this.maintenanceBudgetCommitted);
        summary.setConformanceThreshold(this.m_OptionSettings.getConformingThreshold());

        // set cumulative development and maintenance costs, features
        if (m_Period > 0)
        {
            summary.setDevelopmentSpending((summarySet.get(m_Period -
1)).getDevelopmentSpending()
            + this.committedDevelopmentBudgetCommitted + this.developmentBudgetDiscretionary);

            summary.setMaintenanceSpending((summarySet.get(m_Period -
1)).getMaintenanceSpending()
            + this.maintenanceBudgetCommitted);
        }
        else
        {
            summary.setDevelopmentSpending(this.committedDevelopmentBudgetCommitted +
this.developmentBudgetDiscretionary);
            summary.setMaintenanceSpending(this.maintenanceBudgetCommitted);
        }


summary.setWastedDevelopmentSpending(this.m_SummaryData.getWastedDevelopmentSpend
ing());
```

```
      summary.setRequestedFeatures(this.getSetNumberOfFeatures(requestSet));

      // calculate total featrues and weighted averages for cost and waiting time on feature
      double numberOfRetainedFeatures = 0.0D;
      double featureTimesConformance = 0.0D;
      double featureTimesWaitingTime = 0.0D;
      double featureTimesMaintenanceCost = 0.0D;

      for (int i = 0; i < retainedSet.size(); i++)
      {
        Variation retained = retainedSet.get(i);
        if (retained.isMaintained())
        {
          //    numberOfRetainedFeatures += retained.getWorkingNumberOfFeatures();
          featureTimesConformance += retained.getWorkingNumberOfFeatures() *
retained.getWorkingConformance();
          featureTimesWaitingTime += retained.getWorkingNumberOfFeatures() *
(retained.getTimeDeployed() - retained.getPeriod());
          featureTimesMaintenanceCost += retained.getWorkingNumberOfFeatures() *
retained.getWorkingMaintenanceUnitCost();
          numberOfRetainedFeatures += retained.getWorkingNumberOfFeatures();
        }
      }

      summary.setPerceivedFeatures(numberOfRetainedFeatures);

      if (numberOfRetainedFeatures > 0)
      {
        // weighted average conformance in the retained set
        summary.setRetainedSetConformance(((double) featureTimesConformance) /
numberOfRetainedFeatures);

        // average waiting time from proposal to the delivery per feature
        summary.setAvgFeatureWaitingTime((double) featureTimesWaitingTime /
numberOfRetainedFeatures);

        // find out cost per feature for the period
        summary.setAvgCostPerMaintainedFeature((double) featureTimesMaintenanceCost /
numberOfRetainedFeatures);
      }

      double numberOfDevelopingFeatures = 0.0D;
      double featureTimesDevelopmentCost = 0.0D;

      for (int i = 0; i < developmentSet.size(); i++)
      {
        Variation development = developmentSet.get(i);

        if (development.isDeveloping())
        {
          featureTimesDevelopmentCost += development.getWorkingNumberOfFeatures() *
development.getWorkingDevelopmentUnitCost();
          numberOfDevelopingFeatures += development.getWorkingNumberOfFeatures();
        }
      }
```

```
        if (numberOfDevelopingFeatures > 0)
        {
    summary.setAvgCostPerDevelopmentFeature(featureTimesDevelopmentCost/numberOfDevelopi
    ngFeatures);
        }
        else
        {
          summary.setAvgCostPerDevelopmentFeature(0.0);
        }

        summary.setProposedFeatures(this.getSetNumberOfFeatures(proposalSet));
        summary.setDevelopingFeatures(numberOfDevelopingFeatures);
        summary.setAbandonedFeatures(this.getSetNumberOfFeatures(abandonedSet));
        summary.setDiscardedFeatures(this.getSetNumberOfFeatures(discardedSet));
        summary.setRetiredFeatures(this.getSetNumberOfFeatures(retiredSet));
        summary.setInitiativeFeatures(this.getSetNumberOfFeatures(initiativeSet));

        summary.setBudgetSpending(this.currentPeriodBudget);

        this.m_SummaryData = summary;
        this.summarySet.add(summary);

        // save date updates into database table
        DataOperations dbOperations = DataOperations.getInstance();
        String summarySql = insertSummarySql();

        try
        {
          dbOperations.executeSQL(summarySql, false);
        }
        catch (Exception e)
        {
          System.out.println("Error updating summary data" + e.getMessage());
          System.exit(-4);
        }

        if (this.m_OptionSettings.isLogDetails())
        {
          // Initiative set
          for (int i = 0; i < initiativeSet.size(); i++)
          {
            Variation initiativeData = initiativeSet.get(i);
            try
            {
              dbOperations.executeSQL(logVariationStatusSql(initiativeData,
    DefaultValues.STATUS_REQUEST), false);
            }
            catch (Exception e)
            {
              System.out.println("Error adding VariationStatus data");
              System.exit(-5);
            }
          }

          // discarded set
        for (int i = 0; i < discardedSet.size(); i++)
        {
           Variation discardedData = discardedSet.get(i);
           try
           {
              dbOperations.executeSQL(logVariationStatusSql(discardedData,
DefaultValues.STATUS_DISCARD), false);
           }
           catch (Exception e)
           {
              System.out.println("Error adding Discarded data");
              System.exit(-7);
           }
        }

        // development set
        for (int i = 0; i < developmentSet.size(); i++)
        {
           Variation developmentData = developmentSet.get(i);
           try
           {
              dbOperations.executeSQL(logVariationStatusSql(developmentData,
DefaultValues.STATUS_DEVELOPMENT), false);
           }
           catch (Exception e)
           {
              System.out.println("Error adding Development data");
              System.exit(-6);
           }
        }

        // abandon set
        for (int i = 0; i < abandonedSet.size(); i++)
        {
           Variation abandonedData = abandonedSet.get(i);
           try
           {
              dbOperations.executeSQL(logVariationStatusSql(abandonedData,
DefaultValues.STATUS_ABANDONED), false);
           }
           catch (Exception e)
           {
              System.out.println("Error adding Abandoned data");
              System.exit(-8);
           }
        }

        // application/retained set
        for (int i = 0; i < retainedSet.size(); i++)
        {
           Variation retainedData = retainedSet.get(i);
           try
           {
              dbOperations.executeSQL(logVariationStatusSql(retainedData,
DefaultValues.STATUS_APPLICATOON), false);
           }
```

```
            catch (Exception e)
            {
               System.out.println("Error adding Retained data");
               System.exit(-9);
            }
         }

         // retired set
         for (int i = 0; i < retiredSet.size(); i++)
         {
            Variation retiredData = retiredSet.get(i);
            try
            {
               dbOperations.executeSQL(logVariationStatusSql(retiredData,
DefaultValues.STATUS_RETIRED), false);
            }
            catch (Exception e)
            {
               System.out.println("Error adding Retired data");
               System.exit(-10);
            }
         }
      }
   }


   /**
    * Find number of active project
    */
   private int getNumberOfActiveProjects()
   {
      int numberOfActiveProjects = 0;

      for (int i = 0; i < this.developmentSet.size(); i++)
      {
         if (this.developmentSet.get(i).isDeveloping())
         {
            numberOfActiveProjects++;
         }
      }

      return numberOfActiveProjects;
   }


   /**
    * Find active applications
    */
   private int getNumberOfActiveApplications()
   {
      int numberOfActiveApplications = 0;

      for (int i = 0; i < this.retainedSet.size(); i++)
      {
         if (this.retainedSet.get(i).isMaintained())
         {
            numberOfActiveApplications++;
         }
      }

      return numberOfActiveApplications;
   }


   /**
    * Find active featrues from a set
    */
   private double getSetNumberOfFeatures(List <Variation> data )
   {
      double numberOfFeatures = 0.0d;

      for (int i = 0; i < data.size() - 1; i++)
      {
         numberOfFeatures += data.get(i).getWorkingNumberOfFeatures();
      }

      return numberOfFeatures;
   }


   /**
    * Log detailed level of processing data
    */
   private String logVariationStatusSql(Variation data, int status)
   {
      int isNormalizedValue = 0;
      if (data.isNormalized())
      {
         isNormalizedValue = 1;
      }

      String sql = "INSERT INTO VariationStatus(SimulationID, IterationID, "
            + " PeriodID, VariationPeriodID, "
            + " VariationSequenceID, StatusCode, "
            + " Normalized, WorkingConformance, "
            + " WorkingDevelopmentUnitCost, WorkingMaintenanceUnitCost, "
            + " WorkingNumberOfFeatures, WorkingFailureProbability) "
            + " VALUES ("
            + this.m_Simulation + ", " + this.m_Iteration + ", "
            + this.m_Period + ", " + data.getPeriod() + ", "
            + data.getSequence() + ", " + status + ", "
            + isNormalizedValue + ", " + data.getWorkingConformance() + ", "
            + data.getWorkingDevelopmentUnitCost() + ", " +
data.getWorkingMaintenanceUnitCost() + ", "
            + data.getWorkingNumberOfFeatures() + "," + data.getWorkingFailureProbability() + ")
";

      //      System.out.println("VariationStatus sql: " + sql);

      return sql;
   }
```

```
  /**
   * Insert summary data into the database table
   */
  private String insertSummarySql()
  {
     String sql = "INSERT INTO Summary(SimulationID, IterationID, "
            + " PeriodID, Proposals, "
            + " Requests, Initiatives, "
            + " Discards, Projects, ActiveProjects, "
            + " Abandons, Retentions, ActiveRetentions, "
            + " Retired, DevelopmentCosts, "
            + " MaintenanceCosts, CumulativeDevelopmentSpending, "
            + " CumulativeMaintenanceSpending, CumulativeWastedDevelopmentSpending, "
            + " RequestedFeatures, "
            + " PerceivedFeatures, RetainedSetConformance, "
            + " ConformanceThreshold, BudgetSpending, AverageDevelopmentCostPerFeature, "
            + " AverageMaintenanceCostPerFeature, AverageWaitingTimePerFeature, "
            + " ProposedFeatures, InitiativeFeatures, DiscardedFeatures, "
            + " DevelopingFeatures, AbandonedFeatures, RetiredFeatures "
            + ") "
            + " VALUES ("
            + m_Simulation + ", " + m_Iteration + ", "
            + m_Period + ", " + m_SummaryData.getProposals() + ", "
            + m_SummaryData.getRequests() + ", " + m_SummaryData.getInitiatives() + ", "
            + m_SummaryData.getDiscarded() + ", " + m_SummaryData.getProjects() + ", "
            + m_SummaryData.getActiveProjects() + ", "
            + m_SummaryData.getAbandoned() + ", " + m_SummaryData.getRetained() + ", "
            + m_SummaryData.getActiveApplications() + ", "
            + m_SummaryData.getRetired() + ", " + m_SummaryData.getDevelopmentCost() + ", "
            + m_SummaryData.getMaintenanceCost() + ", " +
m_SummaryData.getDevelopmentSpending() + ", "
            + m_SummaryData.getMaintenanceSpending() + ", " +
m_SummaryData.getWastedDevelopmentSpending() + ", "
            + m_SummaryData.getRequestedFeatures() + ", "
            + m_SummaryData.getPerceivedFeatures() + ", " +
m_SummaryData.getRetainedSetConformance() + ", "
            + m_SummaryData.getConformanceThreshold() + ", " +
m_SummaryData.getBudgetSpending() + ", "
            + m_SummaryData.getAvgCostPerDevelopingFeature() + ", "
            + m_SummaryData.getAvgCostPerMaintainedFeature() + ", " +
m_SummaryData.getAvgFeatureWaitingTime() + ", "
            + m_SummaryData.getProposedFeatures() + ", " +
m_SummaryData.getInitiativeFeatures() + ", "
            + m_SummaryData.getDiscardedFeatures() + ", " +
m_SummaryData.getDevelopingFeatures() + ", "
            + m_SummaryData.getAbandonedFeatures() + ", " +
m_SummaryData.getRetiredFeatures()
            + ")";

     //    System.out.println("Summary sql: " + sql);

     return sql;
  }
```

```
    /**
     * Getter for summary data
     */
    public SummaryData getSummaryData()
    {
        return this.m_SummaryData;
    }


    /**
     * Find period maintenance budget need
     */
    private void adjustPeriodMaintenanceBudget()
    {
        this.maintenanceBudgetCommitted = 0.0D;
        this.maintenanceBudgetNeeded = 0.0D;

        for (int i = 0; i < this.retainedSet.size(); i++)
        {
            Variation retained = retainedSet.get(i);

            // committed will only be there if the flag is true
            if (retained.isMaintained())
            {
                this.maintenanceBudgetCommitted += retained.getWorkingMaintenanceUnitCost();
            }

            // needed will always increase
            this.maintenanceBudgetNeeded += retained.getWorkingMaintenanceUnitCost();
        }
    }


    // ------------- Tester ---------------
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args)
    {
        OptionSettings optionSettings = new OptionSettings();

        SimulationEngine engine = new SimulationEngine();
        engine.start(optionSettings);

        for (int i = 0; i < optionSettings.getNumberOfPeriods(); i++)
        {
            engine.run(1, 1, i, optionSettings);
        }

        engine.stop();
    }
}
```