AMERICAN UNIVERSITY OF BEIRUT

OPTIMAL INVESTMENT DECISIONS IN PRODUCT DEVELOPMENT

by

REMY CHARBEL MAKHLOUF

A thesis
submitted in partial fulfillment of the requirements
for the degree of Master of Engineering Management
to the Engineering Management Program
of the Faculty of Engineering and Architecture
at the American University of Beirut

Beirut, Lebanon
October 2010

AMERICAN UNIVERSITY OF BEIRUT

OPTIMAL INVESTMENT DECISIONS IN PRODUCT DEVELOPMENT

by

REMY CHARBEL MAKHLOUF

Approved by:

Dr. Ali Yassine, Associate Professor Engineering Management Program	Advisor
Dr. Bacel Maddah, Assistant Professor Engineering Management Program	Member of Committee
Dr. Walid Nasr, Assistant Professor Suliman S. Olayan School of Business	Member of Committee

Date of thesis defense: October 28th, 2010

AMERICAN UNIVERSITY OF BEIRUT

THESIS RELEASE FORM

I, Remy Charbel Makhlouf

□ authorize the American University of Beirut to supply copies of my thesis to libraries or individuals upon request.

□ do not authorize the American University of Beirut to supply copies of my thesis to libraries or individuals for a period of two years starting with the date of the thesis defense.

____
Signature

____
Date

ACKNOWLEDGMENTS

First and foremost, I am most grateful to God Almighty who has bestowed on me his grace and kind blessings.

I am heartily thankful as well to my supervisor, Dr. Ali Yassine. With his enthusiasm, his inspiration, his sound advice and his great efforts to explain things clearly and simply, he helped me in making this thesis possible.

I am also grateful to the other members of my thesis committee *Dr. Bacel Maddah*, and *Dr. Walid Nasr* for their ongoing help and guidance through the development of my thesis.

Furthermore, I owe my deepest gratitude to my parents, *Charbel* and *Samia Makhlouf* and my two adorable sisters *Rita* and *Rosemary* who have stood by my side, motivated me, supported me, and encouraged me in all the stressed moments.

Last but not least, I am indebted to my many student colleagues at AUB who supported me in all my work. AUB has not only given me higher level of knowledge and quality education, but also provided me with rich skills that might not have been acquired elsewhere.

I thank you all for being such modest supporters of mine. I could not have done this without you.

AN ABSTRACT OF THE THESIS OF

Remy Charbel Makhlouf for Master of Engineering Management
Major: Engineering Management

Title: Optimal Investment Decisions In Product Development

Product development is an essential activity in most organizations as it reflects its long-term health and profitability. Furthermore, the importance of innovation is paramount in today's technologically driven world. Consequently, this study suggests a systematic methodology to optimize product development investments.

The objective of my work is to develop a mathematical model to maximize the performance of a product under development based on investment constraint. This thesis introduces two product development models: one is deterministic and the other is stochastic. The outcome would be a set of managerial guidelines for optimally investing in various modules of a product and in design rules while taking into consideration the interdependencies between modules. Different scenarios will be explored based on two important problem dimensions: module performance uncertainty and investment frequency. While performance uncertainty reflects the amount of risk (in terms of achieving higher levels of module performance) involved in the investment in product modules, the investment frequency describes whether these investment decisions (in product modules) are made one shot or periodically. The architecture of the product played an essential role in affecting the optimal results and leading to a conclusion that local optimal investments may not necessary lead to global optimal system/product performance.

CONTENTS

ILLUSTRATIONS

TABLES

Table Page

DEDICATION

To the most precious individual

who strongly believed in me, my

beloved: Paul Youssef

To you I dedicate this study

Thank you for your support…

CHAPTER 1

INTRODUCTION

1.1. Background: Designing Complex Products

Throughout history, scholars have been attempting to explore the notion of modularity when trying to understand and develop complex systems in different application fields such as biology (Khastan, et al., 2009), management (Huberman and Hogg, 1995), engineering (Mihm, et al., 2003), psychology (Samuels, 1998), aerospace (Button and Soeder, 2004), software development (Sullivan, et al., 2001) and many others. Designing such a complex system is based upon designing individual components (subsystems or modules) that are parts of larger systems and which can be examined, substituted, modified, augmented and excluded based on their economic value. This is what Baldwin and Clark defined as option values and modular operators (Baldwin and Clark, 2004). They have explained a modularized process by a set of "designed elements that are split up and assigned to modules according to a formal architecture or plan (design rules)" (Baldwin and Clark, 2004). Baldwin and Clark gave an "option-like" property to each module in the system where evaluating the value or performance of any product goes from the option of evaluating the system as a whole to the option of evaluating each module independently. Accordingly, modularity implies that changes in one part of a system should not lead to unexpected behavior in other parts. Then why are we still witnessing transplant rejections even when the donor highly matches the receipt?

Is it due to coordination among many interdependent organs in the complex human body system?

As a result, many researchers believe that in designing complex products, we can individually design or improve each component's performance separately, but this may affect the behavior or performance of other components. This is due to some known or unknown common function or feature in the product which is implemented by more than one component. As opposed to perfect modularity, where each component has its unique functions, integral systems involve a strong dependency between individual modules where changes made to any component (to improve its performance) may deteriorate or improve the performance of others. Consequently, in an integral architecture a local optimal performance for each individual component may not necessarily lead to a global optimal performance of the whole product and this is due to complex interactions between the various components. Integrality supporters argue that any best reachable component's performance is affected by other decision makers (i.e. components) and thus communication is needed among engineers at any decision point to coordinate the mutual development of these components and eliminate this mutual dependency. Mihm, et al. (2003) highlighted on the issue of system's performance "arising from designers making successive local component decisions over time, taking into account the current status of surrounding components".

1.2. Problem Statement

A Review of the literature shows that there have been few studies which address any kind of product whether been modular in design, integral or hybrid. The literature revealed that improving the performance of any product required formulating a mathematical model showing the total product performance which depends on the topology of the product. Either the product was considered modular in design thus all sub-components are independent or the product was considered integral in design where the status of other components must be taken into consideration. Based on the architecture of the product the model was constructed accordingly. What if we consider a product where some of its components do depend on each other while the rest are totally independent? What if we have the option to develop some design rules which eliminate or reduce the interdependency, thus getting a perfectly modular architecture? What if we chose not to invest in design rules but take into account the dependency between modules every time we invest in a certain component?

This thesis addresses these identified gaps in the literature by trying to answering all the above mentioned questions. A methodology for optimally investing in a complex engineering product will be provided by taking into consideration a limited budget and resource constraint.

1.3. Scope of Work and Research Objectives

Between perfect modular designs (where all modules are completely independent of each other) and perfect integral designs (where every module affects others in the system), my study aims to develop a theory of product development performance where the typology (whether modular, integral or a hybrid of both) of the product architecture will be taken into account while improving the product performance by optimally investing in modules and/or design rules.

In reality most engineered systems are neither perfectly modular nor completely integral but somewhere in between and thus a theory to understand the investment policies of such performance evolution systems is necessary. Several techniques will be used to divide each product into module groups aiming to improve the performance of each group separately, thus improving the global product performance (Allada and Lan, 2002). We will suggest two kinds of models: one is deterministic and the other is stochastic. While the first optimizes total product performance for certain modules the latter targets uncertainty where the return on investments for risky modules is no more certain but depend on some uniform distribution function assumed. In each model, two types of investments will be provided: one shot investment versus periodic investments. Finally some managerial guidelines will be provided which will give quick hints about investments strategies. Those insights will be based upon results, analysis work and sensitivity studies done for each model and investment type.

1.4. Significance of the Study

Since product development is a key for any business success and innovation is essential in capturing market demand in our technological driven world, a systematic methodology to optimize the modules' performances of an evolving architectural product is extremely necessary. Accordingly, this study would suggest a set of best practices or guidelines for optimally investing in any product taking into consideration the topology of the product whether modular, integral or hybrid especially that previous developed models targeted only a specific architectural type.

The proposed model should maximize the total product performance taking into account interdependencies of the modules, a limited budget, design rules effect, performance function of each module, time horizon, and the difference in return on investments between risky and certain module.

Finally, this study is beneficial for most business, engineering or any kind of companies where their main objective is to design or re-design a complex product whether been financial, medical, electrical, technological, etc… and bringing it up to the market.

CHAPTER 2

LITERATURE REVIEW

2.1. Introduction

Product development is a term used to define the process of designing a product and bringing it to market. Many researchers have targeted such a topic and described a strategy for improving the performance or value of a certain product by taking into consideration the inter-dependency that exists between the different modules of the product. The literature summarized below will give the reader a diversified idea about product evolution and how such existing models will shape our model to generate a new technique for optimal investment decisions in product development.

2.2. Modularity in the Design of Complex Engineering Systems

Baldwin and Clark (2004) have demonstrated the power of modularity by discussing how a complex engineering system can be modular-in-design by splitting it up in the design process into separate modules. Modularization has three purposes: reducing complexity, allowing parallel work and capturing future uncertainty. Reducing complexity is done by transforming one whole system to many independent modules as shown in Fig.2.1. Some of these modules do not affect other modules and thus called “hidden”. Other modules are called “visible” since design decision for these modules do affect others; thus design rules are needed and must be obeyed by those “visible” modules to eliminate interdependency between them. After establishing the design rules, modules are designed independent of one another and parallel work can be enabled.

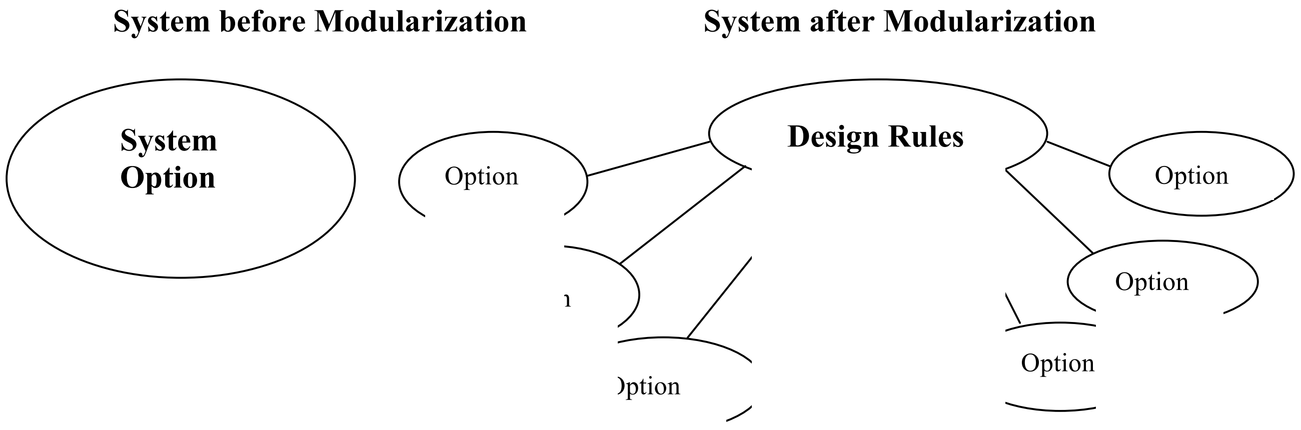


Fig. 2.1: Modularity Creatε Baldwin and Clark, 2004)

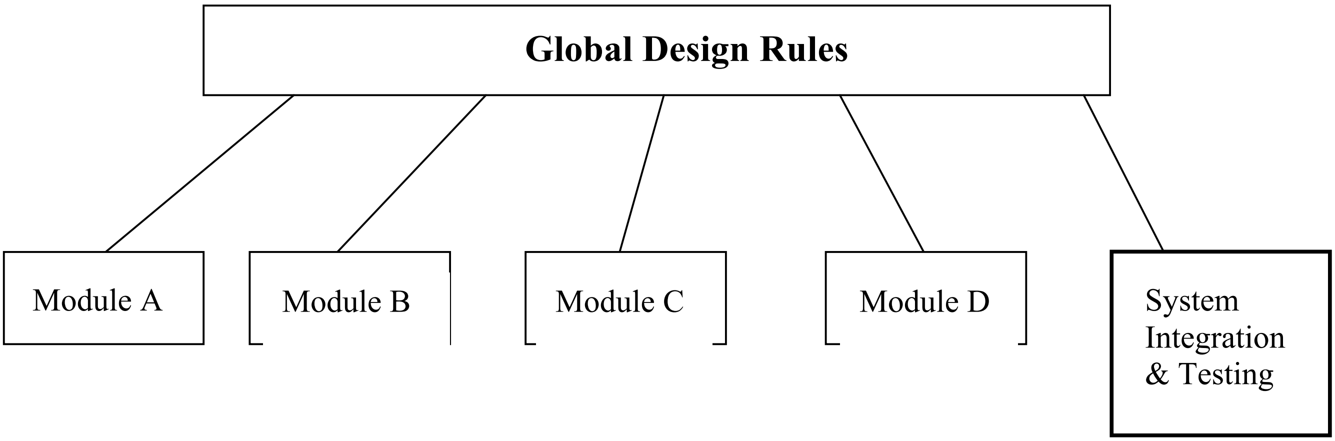
Modularity captures uncertainty becaus ; of such modularized system can be altered and improved over time as long as the design rules are respected. Hence modularity has created design options in each separate module where the engineer has “the right but not the obligation” to choose a certain design over its alternatives when that design shows a better performance. An “option-like” property was given to each module in the system where evaluating the value or performance of any product goes from the option of evaluating the system as a whole to the option of evaluating each module independently.

Baldwin and Clark (2004) explained that there is no perfect design rules and unforeseen insignificant compatibility problems may occur in advanced stages, thus “System Integration and Testing” (SIT) is needed to resolve such minor incompatibility problems. The design rules and the hidden modules affect the System Integration and

Testing but SIT's decisions should not affect the modular architecture of the product or else the system would no longer be modular.

Representing a modular system can be done in several ways. Baldwin and Clark (2004) used the Design Structured Matrix (DSM) map. The DSM map contains several blocks where the first and the last are the design rules and SIT respectively and in between are the components of the product. Such map shows the dependencies between component blocks.

Another representation of a modular system is the Design Hierarchy Representation where the Hierarchy starts with the design rules as a first level and on the second level we have the hidden modules and the SIT as shown in Fig. 2.2.



A Two- r Design H

There is no dden n ow what is ne SIT stage as long as they totally obey the design rules but the System Integration and Testing unit should have knowledge about Modules A, B, C and D in order to resolve any unforeseen incompatibility problem.

In opposed to design rules which once developed are no more altered and are considered to be long lasting, modules accommodate uncertainty, thus lodging experimentation. Baldwin and Clark (2004) introduced six modular operators where the designer can: split, substitute, exclude, augment, collect and organize, and create shells for any module. Such operators affect the structure of the modular system and transform the two-level hierarchy into a more complex one.

The main objective of all Baldwin's and Clark's work was to establish the economic value of a complex engineering system by splitting it up into modules having "option" values. Accordingly, they have assumed that the system's minimal value (i.e. base line) exists and has a value of S_0 while the modules of the system are not yet realized and thus have an uncertain payoff of X_j^1. Then the economic value of the system would be the sum of S_0 and all the X_j^1 as shown in equation 2.2.1:

$$\text{Economic Value of the System: } S_0+\sum_{j=1}^{J} X_j^1 \qquad (2.2.1)$$

X_j^1 denotes "the economic value of a single realization of the random variable X_j" (Baldwin and Clark, 2004) where each "j" denotes a distribution of random variables. Then the total economic value of the system is a sum of J realizations with different distributions. The development efforts realized in each module design define the realization. The realization can take a positive or a negative value. If the value was greater than zero (positive realization) then the total system value would increase by that amount. If the realization was less than or equal to zero (negative or zero realization), the engineer can disregard such module and develop another one. The six modular operators discussed previously can be used in such a case. Accordingly the economic value of the system can be expressed as follows:

□□= □0+ □□□□ □11 , 0+□□□□ □21 , 0+…+ □□□□ □□1 , 0

= □0+ □□1++□□2++ …+ □□□+ 2.2.2

The expected value of any module design is the maximum between its realization and zero. Since equation 2.2.2 is too general, Baldwin and Clark developed further work. They considered the system to be composed of N design parameters and X_α to be the value of a module of size αN where summation of all αs equal 1:□=1□□□ = 1 X_α is assumed to be normally distributed with a mean zero and a variance □^{2}αN :

X_α ~ N(0, □^{2}αN)

X_α = z_α □(αN)$^{1/2}$

Where z_α is a standard normal variant with mean zero and variance one: z_α ~ N(0,1) Substituting X_α in equation 2.2.2, holding S_0 and factorizing, we get:

□□=□ □1, □2, …, □□ ; □;□ = □□12 □112+ □212+ …+ □□12□□□□□, 0 (2.2.3)

As mentioned before, modularization enables parallel work. To express parallel experimentation, Baldwin and Clark supposed that each designer produces k_j independent design efforts in each of the J modules. When all these designs are accomplished, the engineer chooses the best of these k_j designs in each module. Q (k) is defined to be the expected value of the highest realization of k independent designs and the distribution of k is the distribution of the “maximum order statistic of a sample of size k”:

Q (k) = k0∞□(□□□−1□□□□

where N(z) is a standard normal distribution and n(z) is the density function.

Equation 2.2.3 can be updated to accommodate the k designs as below:

□□,□ ;□, □= □□12□=1□□□12□□□□=1□□□□□□ (2.2.4)

After establishing the economic value of a complex engineering system as a function of α, k, □, and N, Baldwin and Clark concluded that modularity in design is neither good nor bad but it is an extremely dangerous concept that need not to be ignored by reminding us with the internet bubble crash where the world fall into extreme losses after being in a highly innovated period.

Design rules' power in eliminating interdependency is used in our model where investing in design rules is a decision variable and depends on some modularity/integrality factor k_{ij} which will be explained later in the model. Baldwin and Clark assumed that Design rules should always exist during the design stage of any product while investing in the modules was assumed to be optional. On the contrary, our model assumes that investing in design rules and in modules are optional decisions depending on the performance of each module and its effect on total product performance. So the designer can choose to invest only in modules and disregard design rules thus taking into account the interdependencies between modules. To reduce such dependencies, one can choose to invest in design rules and make the product modular in design. Consequently investing in the modules in such a case does not demand an attention to the interdependencies that existed originally (before investing in Design rules) in the product. A mathematical model will be developed to explain the performance of each module and design rules will be used in the process of maximizing total product performance.

2.3. Problem – Solving Oscillations in Complex Engineering Projects

This paper targets complex product development projects which require frequent and prosperous communications among project members to ensure the best performance of each project. Complex products are composed of many inter-related sub-components where each engineer is responsible of designing a certain component by taking into consideration the status of other components present in the system as well. Mihm, et al. (2003) have characterized the dynamic behavior of a complex system and have used simulation to derive some managerial actions to improve performance dynamics.

Since complex products are composed of many sub-components, each engineer was responsible of optimizing a local performance measure specific to his component. Accordingly an aggregate system performance was defined as the sum of all local performances of individual components where equal weights were assumed between components. The notation P_i was used to denote the performance function of engineer *i* for the component he is responsible for and *P* as total system performance. As a result, we can write *P* as a function of all P_i :

□=□□□ (2.3.1)

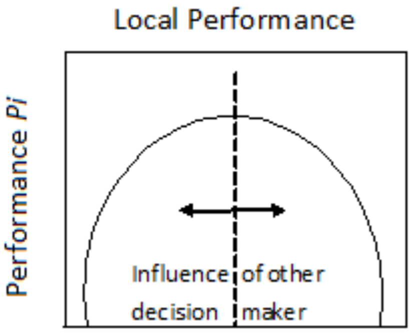
The performance of any component *i* depends on a weighted average of all components in *i* denoted by the decision variable h_i and other components *j* not present in *i* denoted by the decision variable h_j. Thus, $\overline{P_i} = f(h_i, \overline{\{h_j\}})$ (h_i and h_j are assumed to be continuous). The other components effect $\{h_j\}$ were assumed to be constant and P_i became a function of h_i alone as shown in Fig. 2.3 and is represented by $P_i = f(h_i, \{h_j\})$.

To simplify things, $\overline{P_i}$ was assumed to have only one optimum and a quadratic function which includes the effect of other components as shown below:

□□=□ℎ□, ℎ□ =−□□(ℎ□− □≠□□□□,□ℎ□)2+ 1 (2.3.2)

$\overline{hj}$ represents the most recent decision on component *j* which engineer *i* takes it as given. The other decision makers *j* affect the performance of component *i* in two ways. First they can influence the optimal choice of engineer *i* and can shift the optimal position of h_i. So the influence that $\overline{h_j}$ has on $\overline{P_i}$ is represented by $b_{i,j}$ and the summation term □≠□□□□,□ℎ□ captures the shift of the optimal *hi*. The second effect is the influence of the decision makers' *j* status on the optimal performance reached by designer *i*. The best case scenario is when the decisions of components *j* do not affect at all the decision of component *i* and allow it to reach its best performance and the worst case scenario is when components *j*'s designs tighten component *i*'s performance and bring it to its minimum value. In these extreme scenarios, a small change in h_j will create a small difference in P_i but in an intermediate scenario, a change in *hj* matters as shown in Fig. 2.4. To represent such twofold interactions, a performance constraint I_{ij} was introduced and it takes constant values at the extremes $c_{i,j}$ and a slope $a_{i,j}$ in the intermediate region. The total performance $\overline{P_i}$ of component *i* was then obtained by multiplying the potential performance P_i by performance constraint I_{ij} :

$$\overline{\square\square} = \square\square h\square ,\ \overline{h}\square\ \square \neq \square\square\square ,\square \qquad (2.3.3)$$



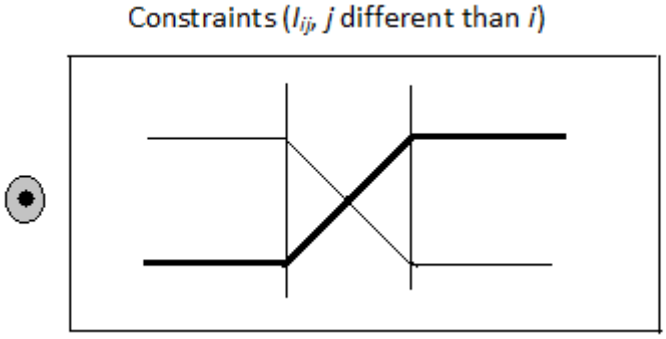


Fig. 2.3: Own decision variable h_i

Fig. 2.4: Decision variable of others, h_j

Fig. 2.3 shows how the position of the optimum shifts linearly with a change in other decision maker's status. It shows that the decision variable h_j highly influence the performance function of component *i* and a local optimum does not necessary lead to a global optimum when the interactions of other components are taken into account.

Fig.2.4 shows how the decision variable h_j which is represented by the gray circle places a linear multiplicative constraint function on performance P_i. As discussed earlier, as the influence of h_j is relatively high on P_i it tightens the performance of component *i* and this is represented by the dotted line. And as component j puts no restrictions on i, it loosen the constraint $I_{i.j}$ and allow component *i* to achieve higher performance. This is represented by the solid line. The constraint $I_{i,j}$ is mostly sensitive to middle values of h_j rather than the extremes and this is shown in the middle region of the figure where the solid and the dotted lines intersect.

After defining the structure of the model, Mihm, Loch, and Huchzermeier have used simulation to characterize the dynamic problem-solving behavior. They have simulated a base- case scenario which was considered as a bench mark for comparing other scenarios. Managerial actions were provided based on the results of the simulation.

The notion of communication between engineers and its effect on the engineer's performance have helped us in our model in defining an integral product where all components are dependent on each other. In the absence of design rules, the designer must allocate his budget between modules while taking into consideration the status of related modules. That is any amount spent on any module will force the designer to spend money on updating or re-designing related modules to accommodate for the changes and remain compatible with the revised or improved module. We will consider in our model the importance of taking other components' status into consideration while trying to improve the performance of any specific module but the influence of one components on the others will not have equal weights as assumed in this model, rather a matrix showing the fraction of re-work at each module will be developed.

2.4. Communities of Practice: Performance and Evolution

Huberman and Hogg (1995) started their paper by presenting a brief definition of community of practice. They explained that once informal networks exist within an organization where communication between people becomes feasible, it creates unified goals, norms and interaction activities, thus constituting a community of practice. A key feature in this paper is about the dependency between the performance of any single individual and the other members of the community. Information should always flow across individuals and each member can choose to work on his own or exploits others' help when it is useful. Huberman and Hogg (1995) tried to characterize the performance of each member by being dependent on its own skills or on some interactions with other members in the organization. Accordingly, they related the total performance of a community of practice to the skills of individuals within the organization.

To quantify things, the overall performance of a community would be the sum of all individual performances:

□= □□□ *(2.4.1)*

where P_i is the performance of individual *i*

It is assumed that for each individual to finalize his work, he must pass by series of steps. So the task is divided into several stages. As mentioned previously, at each stage the individual can choose to do "self-work", i.e. to work on his own and not use others' help in the community or he can decide to make use of others' information which is called "hints". As a result the notation p_{ij} was developed to express the probability of individual *i* choosing to use a hint from individual *j*. When $i = j$, p_{ii} would then denotes the probability of performing self-work. At each stage, the summation of all p_{ij} across j should be equal to 1, □□□□*=1*.

It is assumed that all steps are completed asynchronously (all workers progress and move together in time) at a rate *r*. Then the rate of individual *i* utilizing a hint form *j* is rp_{ij} and the performance of any individual working on his own and does not use hints is: □□ =□□

Each task accomplished whether being self-work or through the use of others' help in the community should have a value. Huberman and Hogg assumed that all self-work activities produce the same benefit denoted by *s* where as the quality of a new hint sent from *j* to *i* is h_{ij} where individual *j* is assumed to be doing self-work in this case. If the hint was useful, then h_{ij} will be greater than *s*. At each step, the member of the community can do self-work and earn *s* or with probability p_{ij} he will use a hint and produces h_{ij}. The hint may be useful thus h_{ij} will be greater than *s* or it may be useless hence losing the opportunity of making *s*.

Hints are assumed to be produced at a rate *w* which is less than the rate *r* which means that not at each step the member can develop a new hint. If *r* was too high, then using hints repeatedly will carry no innovation and this will lead to a decrease in the hint quality. For this reason, a new measure for the hint quality h_{ij}^{eff} is developed which reflects the decline in h_{ij} in case of reusing old hints: *h*□□□□□= *h*□□*1*− □□□□□

Making use of all information presented earlier, Huberman and Hogg defined individual performance P_i while taking into account all the interactions with the community as:

□□=□□□□□*h*□□□□□ *(2.4.2)*

After defining the performance of the community as being the sum of all individual performances and after relating each member’s performance to the interactions that existed in the community, Huberman and Hogg continued their work by examining the changes in the community of practice upon varying the interaction structure. They considered different cases as:

1- All members of the community act independently. There is no flow of information between individuals, thus they cannot use hints.

2- All members have same probability of using hints thus having equal links to all individuals in the community. This is known as flat community.
3- All members can choose hints from one single neighbor; usually the one with the best hint quality.
4- All members can accept hints from several high quality hint sources.

Considering each case separately, Huberman and Hogg tried to define the individual performance P_i starting with extreme cases (1 and 2) and reaching a more generalized individual performance in case number 4. The highest P_i was that of case number 4 which showed that increased size of the community and diversity in receiving hints would lead to the optimal performance.

Similarly, in our model the optimal product performance is attained through diversification. It is always optimal to invest your budget in different modules and not limit yourself to the module with the highest performance as case number 3 since it will not lead to a global optimum. This result was revealed as well in the previous paper where Mihm, Loch, and Huchzermeier (2003) showed that a local optimum does not necessarily lead to a global optimum.

In addition to that, our model measures the total performance of an architectural product rather than a community of practice and the relation that exists between members of the community exists in our work between modules. A similar notation to $\mathrm{p_{ij}}$ is used in our representation where we defined f_{ij} to be the fraction of re-work to be applied to Module j when changes are done to Module i. In contrary to p_{ii}, f_{ii} cannot exist.

2.5. New Modules Launch Planning For Evolving Modular Product Families

Product family (PF) is a group of products manufactured by a firm and which share a common platform. These products have similar characteristics, functions, uses and even marketing requirements. They are also known as product line or product group. Allada and Lan (2002) in their paper tried to develop a methodology to optimize an evolving product family. They have developed a sequential decision process where they aimed to maximize the total profit subject to a time horizon and interdependencies between modules.

Allada and Lan (2002) used Dynamic programming (DP) for representing such "stage-wise sequential decision process". Similar to Baldwin and Clark (2004) who transferred the evaluation of the performance of any product from the option of evaluating the system as a whole to the option of evaluating each module independently, Allada and Lan (2002) will proceed from optimizing the whole product family to optimizing module groups within that product line. Accordingly, smaller DP optimization problems will be developed. As in all dynamic programming problems, stages, states, decision variables and objective function should be clearly defined. The stages in this model are the "time points with equal intervals during the planning horizon" while the states are the possible modules' combination within one module group. The control variable is the decision of whether adopting a certain module design and the objective function is to maximize the profit change of a certain module group by taking into consideration the interdependency assumption.

To formulate several smaller DP problems, "module groups" were defined to group together all modules that depend and affect each other. In this way, any product will be divided into several sub-module groups and the optimization problem would target those characterized groups. Furthermore, Allada and Lan (2002) decreased the state space by introducing the concept of "module cluster". "A module cluster is defined as a set of modules within a module group that are strictly inter-dependent on each other in replacement actions" i.e. module *i* affects module *j* and module *j* in its turn affects module *i* then these two module can be set together as one module *i*.*j*, since any changes done to *i* affects *j* and vice versa. Consider the below four products:

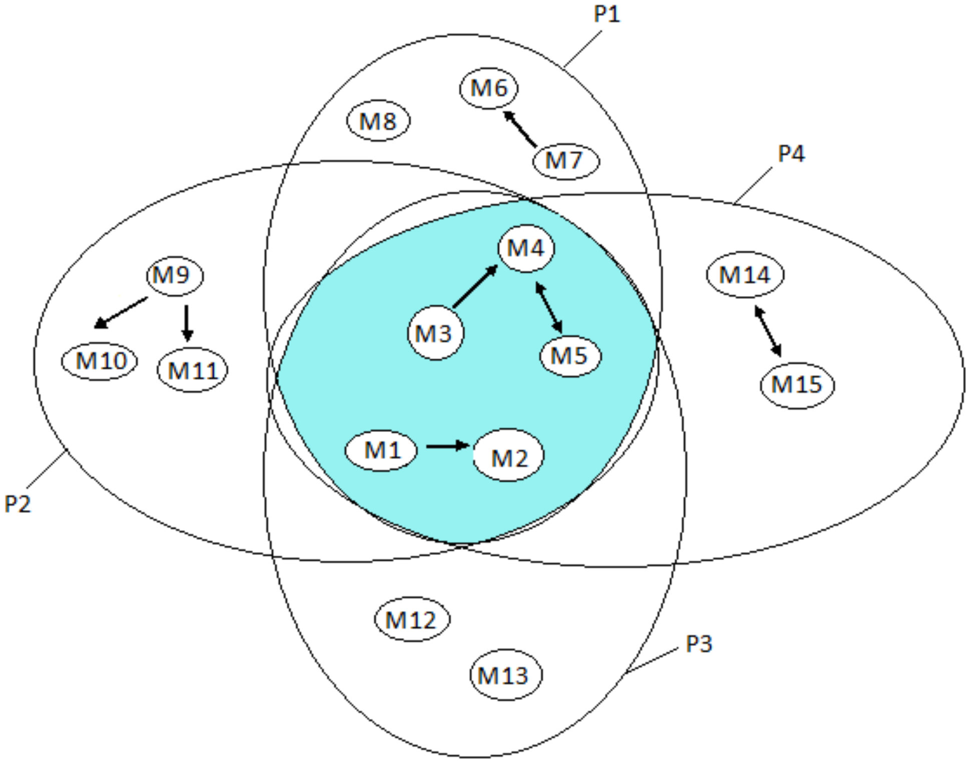


Fig. 2.5: Product family architecture and module interdependency

Fig. 2.5 shows four products P1, P2, P3 and P4 that share a common platform (the blue section) constituting of modules M1, M2, M3, M4 and M5. The arrow (→) shows replacement dependence. To explain what is meant by replacement dependence, different generations of the modules should be considered. Consider the notation *M (A, j)* which represents module A at instance j. If module A was redesigned and been replaced by its higher instance *M (A, j+1)*, then the modules that depend on *M (A, j)* must be redesigned and replaced to attain compatibility with module's *A* changes: *M (A, j+1)*. For example, the decision of replacing module 7 depends on replacement action of module 6. Since a module group is defined as "a group of interacting modules linked by the replacement dependence relationships in a PF" then Fig. 2.5 is divided accordingly into eight module groups:

Table 2.1: Module Group

Group 1	Module 1 and Module 2
Group 2	Module 3, Module 4, and Module 5
Group 3	Module 6 and Module 7
Group 4	Module 8
Group 5	Module 9, Module 10, and Module 11
Group 6	Module 12
Group 7	Module 13
Group 8	Module 14 and 15

As you notice some groups constitute only of one module (as groups 4, 6 and 7) since they do not interact with other modules within the product line. Then redesigning any module in the PF will not affect such groups and no updates are required.

Further work can be done within a group by using the concept of module cluster. As defined earlier, a module cluster is a set of modules that should be replaced together once replacement actions take place. They strictly inter-depend on each other. For example modules 14 and 15 constitute a module cluster and modules 4 and 5 as well. Once redesigning module 14, one should accumulate for module's 15 replacement action and vice versa. Then modules 14 and 15 can be seen as one member in the DP represented by MC14•15.

After defining the states, stages, control variables and objective function, Allada and Lan (2002) indentified their transition probabilities and dynamic programming map aiming to reach an optimal module replacement strategies. This was clearly shown through a deterministic illustrative example after which they generalized their work by developing a stochastic dynamic programming model.

Many ideas from this paper were used in structuring our model. For example, the module group and module cluster scheme is used in our model where each product is divided into groups and the size of the group is decreased by using the concept of module cluster. Accordingly, instead of maximizing the total product performance as a whole, we seek maximizing groups' performances. Dynamic programming cannot be applied to our model, since it bought up some complications especially that our system includes continuous-stochastic formulation.

In addition to that, the notion of updating module *j* to accommodate the changes of module *i* given that dependency exists between the two modules is used as well in our work but expressed in different manner. As said earlier a data matrix which will show the fraction of the rework that should be done once changes are applied to a certain module and other relative data will be developed. In contrary to this paper, our model suggests that the use of design rules decreases the interdependency between modules which is considered in this paper to be fixed and previously known.

2.6. Dell Case-Study

New product development aspects offer various advancement opportunities for the product's performance but it may also add some challenges and riskiness to the product development process. The notion of uncertainty was well described by Krishnan and Bhattacharya (2002) through a Dell case example. Before illustrating their model with the Dell portable computer example, Krishnan and Bhattacharya (2002) defined two technological choices which the development team faces. A choice of *proven technology* which provide limited but certain product improvement or a *prospective technology* choice which is not yet fully proven but offers higher improvement level than a proven technology. So a certain technology would yield a low but guaranteed development where as the uncertain technology would yield high but not guaranteed development.

Thomke and Nigmade (1999) have prepared a detailed case example about "Product Development at Dell Computer Corporation". Dell in 1993 was considering issuing a new portable product to be launched in a 12 month period. By that time, Dell was losing some market share since it lacked portable product. So a high pressure was set on the development team for choosing which feature to be considered as a differentiating characteristic. Researches indicated that price, microprocessor choice, battery life, screen resolution, reliability, weight and size are the respective high rated features in the minds of laptop consumers. Since the company did not want to struggle on price nor on processor speed, the battery life was then considered as the differentiating feature where emphasis should take place while developing the laptop. By that time, NiHi was the used battery which had memory problems and lasts for less than three hours. In contrast, LiOn a new battery technology developed at Sony promised longer recharge lives but was still under development thus considered to be a risky choice. So the team is now faced with three options:

- Use a safe choice battery but which captures less market demand: NiHi
- Use a riskier battery which is still under development but is expected to have a larger profit than the proven technology: LiOn
- Defer commitment to either technologies to a later stage and adopt one of these Two approaches: parallel path approach or overdesigning approach

The defer commitment choice would mean that Dell will not engage neither to the proven NiHi nor to the prospective LiOn but would wait for more information before taking a choice of action. Waiting for more information might cause some delay in the product lunching time. To reduce such delays, the team may consider to *overdesign* the product so that it can accommodate either battery choices. Or the team may choose a *parallel path* approach where two different products are pursued simultaneously one using NiHi and the other using LiOn. The below table summarizes the advantages and disadvantages of considering a certain option. NiHi was referred to as the safest choice but lowest potential. LiOn was considered to give Dell unique product position but with a high uncertainty level and the defer commitment choice was expected to give the largest net profit but requires immediate money outlays and by that time Dell was considered to be severely cash-constrained.

Table 2.2: Advantages and drawback of the three battery choices facing Dell

	Advantages	Drawbacks
1- NiHi	- Safe choice - Dell cannot afford another failure - Would validate new structured process (which seeks to protect firm from further setbacks)	- Lowest upside potential - Does not allow significant market differentiation with respect to battery life.
2- LiOn	- Provides longest battery life and would give Dell unique product position	- Highest risk: technology is still under development - Supply is uncertain if product is very successful; Sony would be single supplier
3- Defer commitment	- Highest expected net margin - Limits downside technology risk if LiOn does not work by qualification	- Violates new process and may becomes precedent for many other decisions involving uncertain outcomes - Not consistent with Dell culture of commitment - Requires additional resources - May demoralize the team involved with the option that is dropped

Krishnan and Bhattacharya (2002) continued their paper by formulating specific equations for each considered choice reflecting the respective expected profit or expected net margins. A decision analysis situation was created for each of the three options and results showed close profits amounts which made the decision hard and thus concluded that no ultimate decision can be based solely on the quantitative calculations; instead some non-monetary factors as morale, process and product strategy should be taken into consideration.

Similar to Dell example, uncertain modules will exist in our model and are generally expected to have higher performance than the certain modules and are considered to be the top features of the product that can provide a higher market share. Three kinds of decisions are available for the designer in our case as well:

1- Since the modules are uncertain and improvements are not definite, a risk averter designer can choose to invest in the parts of the modules that are independent of the uncertain module; such decision will certainly improve the product performance but not with a significant volume. High product performance values cannot be attained in such an investment.

2- Since the uncertain modules are more profitable for the company and extremely important in the eye of the customer, a risk taker designer may choose to solely invest in risky modules aiming to add value to the company and capture most of market demand. Once investing in such risky modules, the designer must allocate his time in a way to update the dependent modules to attain compatibility with the uncertain module's changes if those changes were successful. In such a case, higher product performances can be attained but not with certainty.

3- A designer, who is neither extremely risk taker nor enormously risk averter, may choose to hedge against risk and eliminate the dependency between modules through establishing design rules. Accordingly, once changes are done to any module they will not affect others, as if modules were over designed to accommodate all changes. So in such a case a designer may chose to invest in the risky and certain module at the same time and then decide which one to drop depending on the product performance.

CHAPTER 3

MODEL FORMULATION

3.1. Overview

The methodology adopted in this thesis consists of developing a mathematical model that reflects the performance of a given product. As discussed earlier some ideas from other papers will be utilized in the process of defining an aggregate system performance. Our work will be divided into two types of models: deterministic and stochastic models which will be discussed in chapters 4 and 5 respectively. In this chapter, the performance function of each module and its corresponding parameters will be defined. Below is a table which summarizes all the parameters needed to formulate the model.

Table 3.1: Parameters of the model

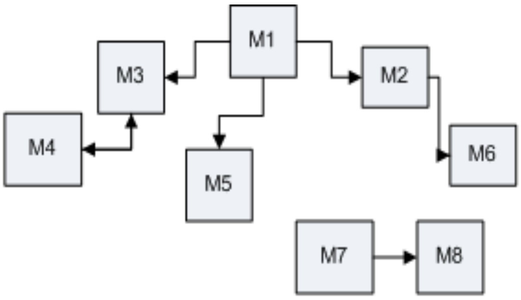
Parameter	Definition
S_T	Total System Performance
S_g	Performance of group g $1 \leq g \leq m$
P_i^g	Performance of module i in group g $1 \leq i \leq n_g$
f_{ij}	Fraction of update applied to module j to become compatible with the changes applied to module i in the absence of design rules
$f_{ij}^{`}$	Fraction of update applied to module j to become compatible with the changes applied to module i in the presence of design rules
k_{ij}	Modularity score which defines the knowledge of the relationship that exists between modules i and j
θ_{ij}	Amount invested in design rules between modules i and j
α_i	Percentage invested in module i
U_i	Upper limit value for the performance function of module i
C_i	Proxy for module i design's complexity
n_g	Number of modules in group g
m	Total number of groups

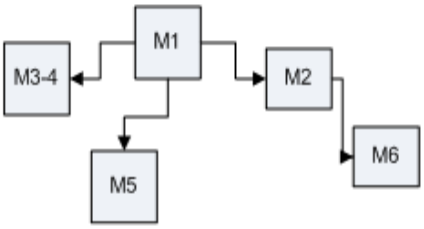
3.2. Defining Parameters

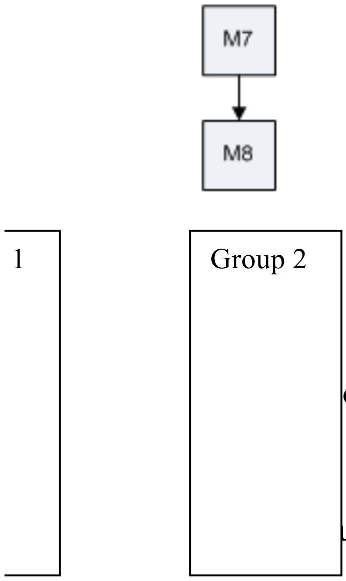
The main objective of this model is to achieve a global optimum performance for a given product taking into consideration modularity and integrality effects between the sub-components of the product. Any product is made of different modules where some modules are dependent on others while some are totally independent. In this model, we will group the modules that affect each other and we will be optimizing groups' performances aiming to reach optimum system performance. Equation 3.2.1 denotes the total system performance (S_T) which can be expressed as the sum of m optimal independent groups' systems performances (S_g). Our main work will be on optimizing S_g:

⍰ ⍰=⍰=⍰⍰⍰⍰ (⍰.⍰.⍰)

The option-like property defined by Baldwin and Clark (2004) helped us in moving from one total system decision to many sub-system group decision. As Allada and Lan (2002), we will proceed from optimizing the whole system product to optimizing module groups present in that product. Within a group we can check for *module cluster* which is defined as "set of modules within a module group that are strictly inter-dependent on each other" (Allada and Lan, 2002). A module cluster will be treated as one module, since any changes done to any module in the cluster will affect all the remaining in the cluster. Fig. 3.1 shows a product composed of eight modules which can be split up into two groups: Group 1 containing modules 1, 2, 3, 4, 5, and 6 and Group 2 containing modules 7 and 8. Groups 1 and 2 are totally independent and we will seek maximizing groups' 1 and 2 performances to reach an optimal total system performance. The double headed arrow (◀▶) between M3 and M4 indicates a module cluster in Group 1 composed of modules 3 and 4. Any changes to module 3 will affect module 4 and vice versa. Then, modules 3 and 4 can be seen as one module M3-4 as shown in Fig. 3.2. The one headed arrow (→) indicates dependence, i.e. any changes done to M7 will affect M8. If M1 changes then M2, M5 and M3-4 are affected but not vice versa and once M2 changes, M6 will change as well.







odules' Interdependencies

Fig. 3.2: Group 1 (n_1=5) and Group 2 (n_2 =2)

ıpdate a certain product, the designer seeks improvements to be done to the modules whether in shape, size, quality etc… Improvements differ among modules. Higher investments amounts will be usually allocated to those modules that are considered to be the top features of the product causing a high market capturing rate. Similar to Dell Case, uncertain modules are expected to have higher performance than any other certain modules. Uncertain modules will be targeted in Chapter 5 of this thesis.

To improve performances of the modules present in the product, money should be spent then in an optimized way. For that reason, companies specify budgets to be spent on their products for the exerted efforts and invested resources. Each company specifies a budget B_g for each group present in its product depending on the size of the group and types of existing modules. Some groups will demand higher budget than others since they will be composed of more complex modules or even more important or essential modules. The budget will be used for improving the groups' performances present in a particular product and for establishing some design rules to reduce interdependencies between modules. This model gives the option of investing in design rules, unlike

Baldwin’s and Clark’s (2004) model which assumes that design rules always exist. As more effort is spent on design rules as much the modules tend to be independent. So if two modules are dependent, then spending money on improving the first oblige us as well to spend money on the second and redesign it to remain compatible with the first. But by the excessive use of design rules, the two modules will tend to be totally independent. Thus spending money on improving the first does not force the designer to spend money on the second to attain compatibility.

Consider a product composed of two modules M_i and M_j only (i.e. one group only) and a budget *B* which is assigned to improve the performance of this group whether in investing in design rules or in modules. The dependency that exists between modules is explained through f_{ij}; $0 \leq f_{ij} \leq 1$. We will define for each group *g* a data matrix “D^g” (as in Fig. 3.3) that shows the fraction or percentage of effect between modules *i* and *j* in a certain group *g* $(1 \leq g \leq m)$ and other parameters related to that group which will be explained later in this chapter.

$$D^g = \begin{array}{c} \\ M_i \\ M_j \end{array} \begin{array}{c} \begin{array}{cc} M_i & M_j \end{array} \\ \begin{pmatrix} C_i^g ; U_i^g & f_{ji}^g ; k_{ji}^g \\ f_{ij}^g ; k_{ij}^g & C_j^g ; U_j^g \end{pmatrix} \end{array}$$

Fig. 3.3: Data matrix

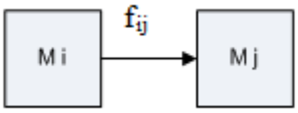


Fig. 3.4: A product composed of one group

The data matrix “D^g” contains several notations. The diagonal reflects parameters that are related to the unique structure of each module whereas other elements as M_iM_j and M_jM_i reflect parameters that has to do with the relationship that exists between modules. We will first start by explaining f_{ij} which indicates that (f_{ij}*100) % of Module *j* must be redesigned when changes are done to Module *i* and (f_{ji}*100) % of Module *i* must be redesigned to become compatible with Module's *j* changes. A zero f_{ij} or f_{ji} in the matrix indicates no impact between modules, in our case f_{ji} should be equal to zero indicating that M_j does not affect M_i and this is shown in Fig. 3.4 where the arrow indicates only effect from M_i to M_j. As a result, and prior to spreading the budget between the modules, a good understanding of the architecture of the product and the relationship between modules is necessary.

So the notation f_{ij} will be used to specify the fraction of change that should be applied on module *j* to become compatible with the changes applied to module *i* in absence of design rules, and we will introduce the notation f_{ij}` to specify the fraction of change that should be applied on module *j* to become compatible with the changes applied to module *i* in the presence of design rules. Note that $f_{ij}` \leq f_{ij}$ since design rules have the potential to decrease the interdependencies between modules. Any two modules will be first related by f_{ij}, and after establishing some design rules and reducing interdependency between them, they will be related by f_{ij}`. Note that each group has its unique data matrix and unique parameters that is why the superscript g is used to differentiate between parameters of different groups.

Let $\theta_{ij}B$ be the amount invested in design rules between Modules *i* and *j*. f_{ij}` is a function of $\theta_{ij}$ and λ (λ is an improvement rate parameter, $\lambda \geq 0$) where $f_{ij}$` decreases as θ_{ij} increases $0 \leq \theta_{ij} \leq 1$. Accordingly, we will assign a decreasing function for f_{ij}` and $f_{ij}$` should be in general a function of λ, θ_{ij} and k_{ij}: $f_{ij}` = \mathrm{f}\,(\lambda, \theta_{ij}, k_{ij})$.

We have chosen an exponential function for illustrating f_{ij}` but this is not the only function that can be used to model $f_{ij}$`. If an exponential decreasing function is chosen, then f_{ij}` will be expressed in the below equation:

□ □□ `= λ□−□□□□□□□; ;λ ≥ 0

(3.2.2)

When we choose not to invest in design rules, θ_{ij} must be equal to zero and f_{ij}` should be exactly equal to $f_{ij}$. In this case $\theta_{ij} = 0 \longrightarrow f_{ij}` = \lambda = f_{ij} \longrightarrow$ therefore:

□ □□`= □□□□−□□□□□□□□□ $k_{ij}^{g} > 0$; $0 \leq \theta_{ij}^{g} \leq 1$; $0 \leq f_{ij}^{g} \leq 1$

(3.2.3)

k_{ij} is a parameter that reflects the amount of decrease in f_{ij} with respect to an increase in θ_{ij} and is shown next to f_{ij} in the data matrix. Similarly for θ_{ij}^{g} and k_{ij}^{g}, the superscript g is used just for differentiating parameters among groups.

Fig. 3.5 below shows f_{ij}` versus changes in θ_{ij} for k_{ij} = 10, 20, 30 and 40.

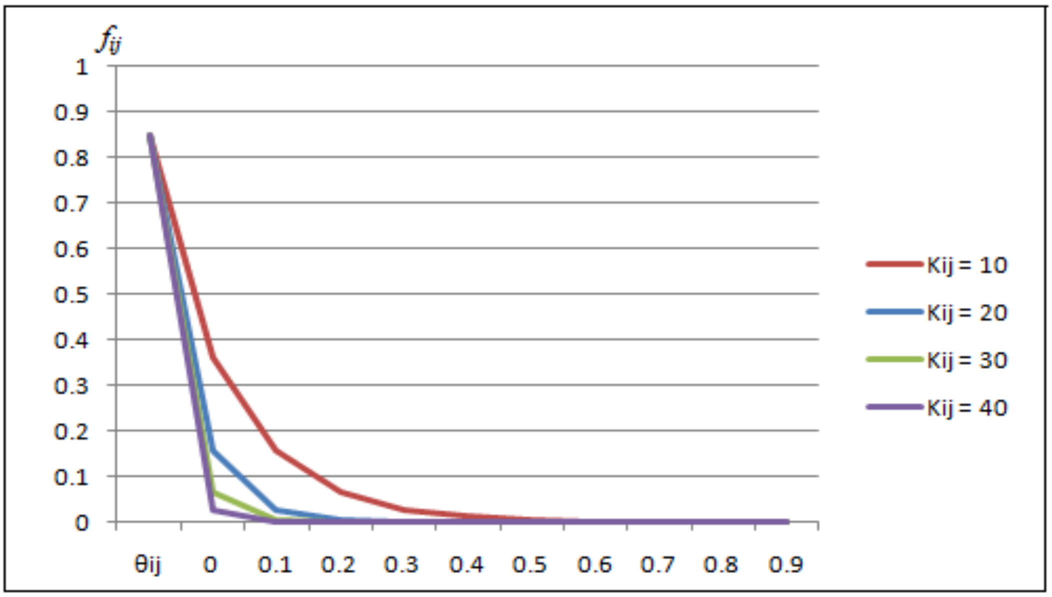


Fig. 3.5: f_{ij}` for different k_{ij} , θ_{ij} and $f_{ij} = 0.85$

As you notice from Fig. 3.5, for a low k_{ij}, the drop in the curve is less than that for a higher k_{ij}. Accordingly we will define k_{ij} as a score of Modularity i.e. k_{ij} will define a measure of modularity between any two dependent modules and can take a score greater than zero. In other words, k_{ij} defines then the knowledge of the relationship that exists between two modules *i* and *j* and it increases with the increase of the designer's knowledge making the relationship more modular. This means that in a complex group where designer doesn't know much about the dependency between the modules, investing in design rules has a lower impact on f_{ij}. By this we mean, that it will slightly decrease the interdependencies between modules. Such kinds of complex groups are assumed to be highly integral and demand a low score of Modularity as the red curve in Fig. 3.5 where we notice that as k_{ij} gets closer to zero as the group's integrality increases.

Concerning trivial or simple relationships between modules, where the designer knows much about the architecture of the group, investing in design rules will make f_{ij}` highly less than f_{ij} implying more decrease in the interdependencies between modules. Such kind of uncomplicated relationships between modules are assumed to be highly modular and demand a high score of modularity. Once the group is composed of more than two modules, k_{ij} must be an indicator for the kind of relationship between any module i and module j whether modular, integral or somewhere in between. We will have in this case several k_{ij}. Referring back to Fig. 3.2, the designer should have knowledge of four modular scores: k_{12}, k_{15}, $k_{13\text{-}4}$, and k_{26}.

As the product is more integral (k_{ij} is low), investing in design rules will not lead to an optimal product performance since reducing interdependencies will demand a large part of the budget. In such cases we are better off not investing in design rules. As the product is less integral and more modular, as investing in design rules is necessary to achieve total optimal product performance. As k_{ij} increases as amount invested in design rules increases as well up to certain k_{ij} where beyond it θ_{ij} attains approximately constant level. This happens when k_{ij} become very large and the drops in f_{ij} to f_{ij}` become too similar as shown in Figure 3.6.

As you notice from Figure 3.6, the three curves collapses approximately for values of k_{ij} = 40, 50 and 60, thus the amount invested in design rules for such three modularity/integrality relationships will be roughly equal even though k_{ij} are different. This proves what have been said earlier about f_{ij}` becoming too similar when k_{ij} attains large modular values. Accordingly, an upper limit for k_{ij} could be defined and it is equal to M. Note that even when f_{ij} is different than 0.85 assumed in the Fig. 3.5 and Fig. 3.6, the upper boundary of k_{ij} still holds.

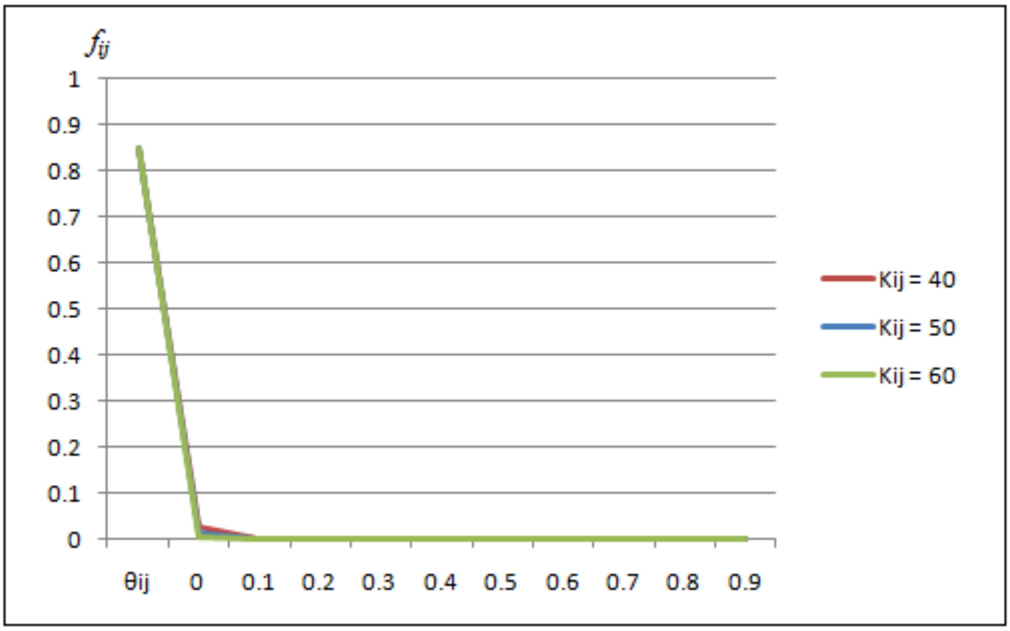


Figure 3.6: f_{ij} in function of θ_{ij} for high values for k_{ij}

3.3. Performance Function

Each module's performance should be measured by a certain function and differentiated from other modules by some parameters. Performance functions are dependent on amount invested in each module. Some modules' performances are highly sensitive to dollar amount invested in improving the module whereas others are less sensitive. We will define different types of modules' performances all based on the same performance function with different parameters and we will assume that the designer is extremely knowledgeable about the module he is designing that he can specify in advance the type of performance the module will attain by specifying the parameters U_i and C_i discussed below. Accordingly the performance of any module should be a function of U_i, C_i, and α_i. Since our objective in the coming chapters is to introduce time component and formulate a periodic investment model, then $P_i = f(U_i, C_i, \alpha_i)$ should increase at a decreasing rate by time. For illustration, we have chosen the following function to express the performance of modules aiming to introduce time component in the coming chapters:

□□=Ui1+e−CiαiBg−Ui2 (3.3.1)

Where P_i denotes the performance of module i (in units if U_i)

U_i = value of curve at upper limit for performance value (units of performance)

C_i = a proxy for module *i* design complexity, $0 \leq C_i \leq 1$ (unit less)

α_i : denotes percentage invested in module *i*, $0 \leq \alpha_i \leq 100$ (unit less)

B_g : denotes budget of group g where *i* belongs to group g (\$)

Note that in the model formulation all notations will include the superscript g as shown in the data matrix above (Fig. 3.3) to differentiate between modules from one group to another.

C_i can be a proxy for modules designs' complexity, where a simple design module, which we will assign for it a large C_i, will directly react upon investing in it a small amount of the budget while a complex design module (small C_i) will demand a higher investment amount than a simple module for attaining a similar performance. Complex modules are assumed to be the most essential modules in the product and are expected to have a higher upper limit value U_i than any other simple module and are considered to be the top features of the product causing a high market capturing rate. The parameter C_i is independent of k_{ij} the modularity factor where the first explains the type of each module whether complex or simple, and the latter explains the relationship between two dependent modules. Since each module has its specific $\mathrm{U_i}$ and $\mathrm{C_i}$, then the values of these two parameters are known in advance and shown on the diagonal of the data marix.

$\alpha_i B_g$ is the fraction invested in module i (from a budget B_g). Then α_i represent a percentage of the budget B_g ; $0 \leq \alpha_i \leq 100$. For every \$ $\alpha_i B_g$ invested in module i, we need to invest "\$ $f_{ij}` \alpha_i (C_j / C_i) B_g$" in module j for module j to remain compatible with module i where $(\mathrm{C_j} / \mathrm{C_i})$ is a scaling factor used to demonstrate the complexity of module j with respect to module i. We should differentiate between investing α_j% in module j where our aim would be improving performance of module j and investing "$(C_j / C_i) f_{ij}` \alpha_i$ %" in module j as a result of investing α_i% in module i and our aim in this case is to update module j to accommodate module's i changes. α_j% is optional, i.e. a designer can choose to invest in module *j* or not, but if the designer chose to invest α_i% in module *i*, then he is obliged to spend “$(C_j / C_i) f_{ij}$` α_i %” in module *j* taking into consideration that module *j* depends on module *i*.

We are assuming in our model that “$(C_j / C_i) f_{ij}$` α_i %” invested in updating module *j* is totally independent from α_j% spent in improving module’s *j* performance. But this is not always the case. Consider the below four scenarios shown in Fig. 3.7, 3.8, 3.9 and 3.10:

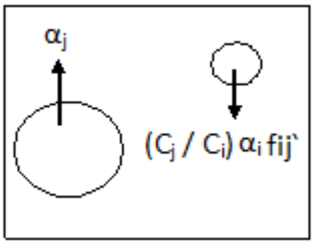


Fig. 3.7: updating module *j* and improving *j* are totally independent

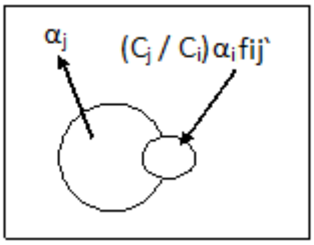


Fig. 3.8 updating module *j* intersects improving module *j*

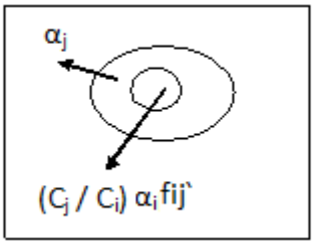


Fig. 3.9: investing in module *j* includes updating *j* with module’s *i*

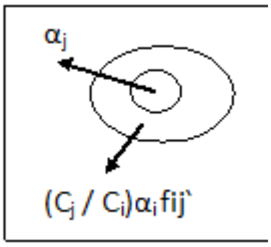


Fig. 3.10: updating module *j* with module’s *i* changes includes improving *j*’s

Fig. 3.7 shows that updating module *j* to attain compatibility with module *i* does not influence the performance of module *j* where as Fig. 3.8 shows an intersection between these two acts where updating module *j* affects part of module's *j* performance. Fig. 3.9 shows that investing in module *j* and improving its performance will cover achieving compatibility with module *i* as well while Fig. 3.10 shows the opposite; updating module *j* will improve its performance also.

Similar to equation (3.3.1), the performance of each group S_g is the sum of all modules' performances present in that group:

□□=□=□ □□□□□ ; $1 \leq g \leq m$

(3.3.2)

where n_g is the number of modules present in group g

Our objective is to maximize S_g subject to a budget constraint. We will develop an optimization problem for maximizing each group's performance and the total product performance will be the sum of all the maximized S_g. We will target in our model those groups with large number of modules where the optimal amount invested in modules or in design rules is not quite simple or direct.

After defining all the parameters needed and before moving to Chapter 4 to formulate the optimization problem under deterministic conditions, we will list the main assumptions presumed in this chapter:

- k_{ij} has an upper limit of M
- C_i and k_{ij} are totally independent
- Investing module *j* and updating it are two independent acts
- Complex modules will have higher performance i.e. low C_i demands high U_i

CHAPTER 4

DETERMINISTIC MODEL

4.1. Overview

After defining all the parameters needed to formulate our optimization model, we will target in this chapter certain modules where there upper limit value U_i is guaranteed. In the next chapter, the notion of uncertainty will be introduced and we will see how the optimal investments decisions would be affected once risky modules exist in a product. This chapter will be divided into two main sections: the first will show one shot investment and the second will show periodic investments decisions. In the multiple shots decisions model, the time component will be introduced. Illustrative examples followed by analysis will be provided to ensure full understanding of the deterministic model.

4.2. One-Time Investment Model

Consider a product composed of m groups and each group contains n_g modules. Our objective is to maximize total product performance using a single investment decision (i.e. how much dollars to allocate to each module or design rule) at the beginning of the development process; i.e., no time component is present. We are assuming that the company makes a one-time decision about how much should each module acquire from the budget for the re-design or improvement stage. We consider a normalized budget of 1 for each group: B_g = 1 for 1 ≤ g ≤ m.

Max ST=Max□=□□□□

= Max□=□ □□=□ □□□□□ = Max□=□ □□=□ □□Uig 1+e−Cigαig – Uig2

Subject to: □=1□□=1□□□□□ + □,□(□≠□)□□□□□□ + □,□ (□≠□)   □□□□□`□

□□□□□□□□□□] = 1

□=1□□=1□□□□□ + □,□(□≠□)□□□□□□ + □,□ (□≠□)   □□□□□□(

□−□ □□□□□□□□□□□□) □□□□□□□□□ =1

$0 \leq \alpha_i \leq 100, \quad 0 \leq \theta_{ij} \leq 1$

Since we are maximizing the objective function, it is necessary to check the optimality conditions and test for the concavity of the maximized function and convexity of the constraint. Let us first consider the objective function:

- Max S_T= Max □=□ □□=□ □□Uig 1+e−Cigαig – Uig2

Since any weighted sum of a concave function is concave, then it is enough to show that Uig 1+e−Cigαig – Uig2 is concave for any α_i^g by showing that second order derivative with respect to α_i^g is nonpositive. Taking the second derivative we get the following:

∂2□(□□□)2Uig 1+e−Cigαig – Uig2 =UigCig2e−Cigαige−Cigαig−11+e−Cigαig3 <0   for all α_i^g

Therefore, we can conclude that the objective function is concave for all α_i^g.

Second we must consider the constraint and check as well its convexity, since any concave function on a convex set should have a unique global optimal.

- □=1□□=1□□□□□ + □,□(□≠□)□□□□□□ + □,□ (□≠□) □□□□□□(□−□ □□□□□□□□□□□□) □□□□□□□□□□

Taking each term by itself: □=1□□□□□ and □,□(□≠□)□□□□□□ are convex and concave for all α_i^g and θ_{ij}^g. Concerning the last term of the summation we have to prove that (□−□ □□□□□□□□□□□□) □□□ is convex where constants were omitted. To prove convexity, we have to prove that the Eigen values for the Hessian matrix are always positive or $x^THx > 0$ for all x where H denotes the hessian matrix.

Since the Taylor series expansions for exponential functions starts always with a linear function (□□=1+□+ x22!+...) then the assumed exponential function for $f_{ij}^{`g}$ can be converted to a linear version where □−□ □□□□□□□□□□□□□=1−□□□□ □□□□ □□□□ and the proof will target f_{ij}^g (1−□□□□ □□□□ □□□□ α_i^g (C_j^g/C_i^g) instead of □□□□(□−□ □□□□□□□□□□□□) □□□□□□□□□□ . Since constants can be omitted we will work with (1- $k_{ij}^g f_{ij}^g \theta_{ij}^g$)α_i^g only. The Hessian matrix associated with this function is the below:

H=0−□□□□□□□□□−□□□□□□□□□0 and

xTHx=□1□20−□□□□□□□□□−□□□□□□□□□0□1□2

= −2□1□2□□□□□□□□<0 for all x

≫□□□□□□□

Therefore, we can conclude that the constraint is concave. But we know that if the constraint g(x) was set to be < B and g(x) is concave, then –g(x) > -B is convex. That is, taking the negative of a concave function will give us a convex function. Concave function over a convex set allows for a global optimum.

After defining the model and testing the optimality conditions, we move next to an illustrative example.

4.2.1. Illustrative Example

Consider a product composed of only one group (m=1). Within this group, we have six modules (n_1 =6) related together based on the below diagram:

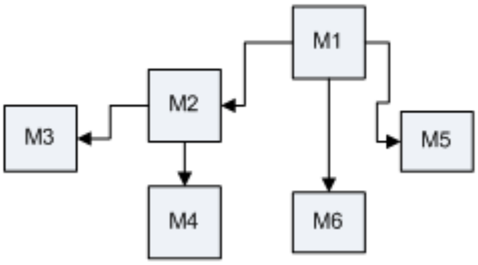


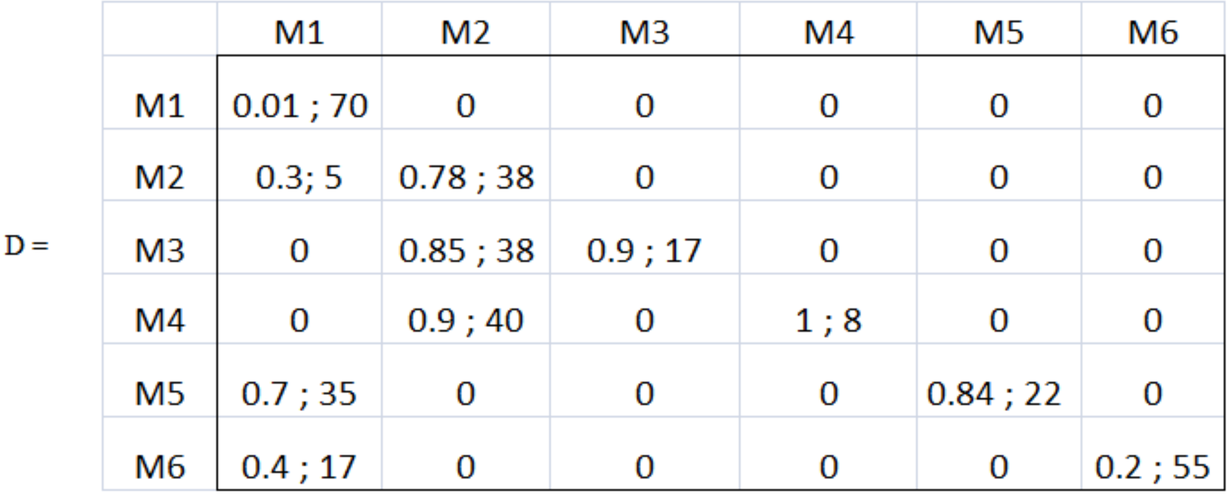
Fig. 4.1: A product composed of
one group having six modules

From Fig. 4.1 we notice that modules six and five depends on module one and modules four and three depends on module two which in its turn depends on module one. Therefore, the designers of such a product should be well knowledgeable of the relationships between modules and about the types of modules whether being complex or simple so that he can specify the parameters accordingly.

Assume that module one is the most important feature of the product and module four the least important. Based on that, module one should have the highest upper limit value U_1 and the lowest simplicity factor C_1. One ultimate question would be: will the highest investments amounts be allocated to the most complex modules with the highest upper limit U_i ? A quick answer to this question would be: it depends.

It depends on how much C_i is low and on how much U_i is high. This example will show us that even though module one has the highest upper limit between all the modules but it will take the smallest alpha and this is due to many reasons that will be discussed in the analysis section. So we are better off investing a small amount in module one.

Consider the data matrix "D" which summarizes all the information about the architectural design of the modules and their interdependencies.



D =

	M1	M2	M3	M4	M5	M6
M1	0.01 ; 70	0	0	0	0	0
M2	0.3; 5	0.78 ; 38	0	0	0	0
M3	0	0.85 ; 38	0.9 ; 17	0	0	0
M4	0	0.9 ; 40	0	1 ; 8	0	0
M5	0.7 ; 35	0	0	0	0.84 ; 22	0
M6	0.4 ; 17	0	0	0	0	0.2 ; 55

Fig. 4.2: Data matrix for a product composed of six modules

The diagonal of the data matrix "D" shows the complexity factor C_i and the upper limit value U_i for the performance of each module. As you notice module one is the most important module in the product which has the highest performance reflected by the upper limit U_1 = 70 and it is the most complicated module in the product having the lowest simplicity factor C_1 = 0.01. Module six is the second important module in the product having a U_6 = 55 which is less than U_1 but the design of module six is less complex than that of module one and it is reflected by a higher C_6 = 0.2. The least complex module is module four. Its complexity factor C_4 = 1 implying no complexity at all. From the other side, being too simple will not result in a high performance where U_4 = 8.

On the other hand, the diagonal entries, M1M2 for example shows an f_{12} = 0.3 and k_{12} = 5 which means that 30 % of module two must be re-designed to attain compatibility when changes are applied to module one and the modularity factor k_{12} shows an integral relationship between the two modules. This implies that the designer does not know much about the interdependency between module one and two. Remember that once k_{ij} is small then we will not see a sharp drop from f_{12} to f_{12}\`. We expect in such a highly integral case no investments in design rules to take place between module one and two. M2M4 shows the highest modularity/integrality factor where k_{24} = 40 which shows full designer's knowledge of the architectural link between modules two and four. If the amount invested in module two was among the highest alphas then we are better of investing in design rules to reduce interdependency especially that 90% of module four must be re-designed once updating module two. In addition to that, the designer is extremely knowledgeable about the relation of these two modules, so investing in design rules can directly decrease f_{12} to a relatively small f_{12}\`. In such a case, we expect that the designer to make use of this high $k_{24}$ to reduce amount needed to update module four: $\alpha_2 f_{24}$\`(C_4/C_2). The data matrix contains five non-diagonal entries:

M1M5, M1M6, M1M2, M2M3 and M2M4 which clearly explain all the links present in Fig. 4.1.

4.2.1.1. Excel Solver Results

Using Excel-Solver, we optimized the above mentioned example which is composed of one group only made of six modules and we got an optimal total product performance of 69.78 and the optimal investments amounts in design rules and modules are shown in the below table:

Table 4.1: Decision variables values after optimization

α_1	0.36%	θ_{12}	0%
α_2	12%	θ_{15}	5.27%
α_3	8.09%	θ_{16}	0%
α_4	6.56%	θ_{23}	4.12%
α_5	9.21%	θ_{24}	4.66%
α_6	27%		

From Table 4.1 we notice that around 63.23% (=16) of the budget went to re-designing modules, 14.06% (,=1 ≠ 6) for developing design rules and the 22.71% (,=1 ≠ 6`( )) left is for updating modules to attain compatibility with the changes done. We notice as well that re-designing module six demanded 27% from the budget followed by module two where 12% of the budget was allocated for its development work. Modules five, three and four took from the budget 9.21%, 8.09% and 6.56% respectively. The lowest alpha is that of module one where only 0.36% of the budget should be spent in re-designing it. Concerning design rules, no work should be done to reduce the interdependency between modules one and two and one and six. 5.27% of the budget must be spent on developing design rules between modules one and five, 4.66% and 4.12% must be spent respectively to decrease the dependency between modules two and four and between modules two and three.

Other important results which are of a high benefit to us in the analysis are the reduction in the fraction of updates from f_{ij} to f_{ij}` and the amounts that should be spent to update module j once changes are applied to module i: $\alpha_i f_{ij}$` (C_j / C_i).

Table 4.2: f_{ij} *vs* f_{ij}` *and* $\alpha i f_{ij}$` (C_j / C_i)

M_i M_j	M_1 M_2	M_1 M_5	M_1 M_6	M_2 M_3	M_2 M_4
f_{ij}	0.3	0.7	0.4	0.85	0.9
f_{ij}`	0.3	0.19	0.4	0.22	0.17
$\alpha_i f_{ij}$` (C_j / C_i)	8.37%	5.78%	2.86%	3.10%	2.59%

Table 4.2 shows that f_{ij}`$=f_{ij}$ in the absence of design rules where θ_{15} and $\theta_{16} = 0$. We notice a huge decrease in the fraction of update between modules two and four, where prior of investing in design rules, 90% (=f_{24}) of module four must be re-designed to attain compatibility with module's two changes while only 17% (=f_{24}`) now must be re-designed after spending part of the budget on developing design rules which clearly

decreased the dependency between these two modules. The fraction of re-work between modules one and five decreased from 70% to 19% and that of modules two and three decreased from 85% to 22%. The last row in Table 4.2 shows the percentages of the budget that should be assigned to update the dependent modules after investing the optimal alphas amounts in improving the modules' performances.

4.2.1.2. Analysis

In the analysis section we will start first by sorting the optimal alphas in a descending order where the first listed alpha refers to the highest amount invested in a certain module and the last listed alpha refers to the least amount of investment between the modules. The sorted alphas are as follows: α_6, α_2, α_5, α_3, α_4, and α_1. The highest investment amount went to module six which has the second highest upper limit $U_6 = 55$ and the lowest investment amount went to module one which has the highest upper limit $U_1 = 70$. One usually expects that the highest α_i goes to the module with the highest U_i. This can be the case in our example if module one was disregarded and the rest of the modules were sorted by their upper limit in a descending order. By this we mean, if we sort modules two, three, four, five and six by their U_is from the largest to the smallest we get: U_6, U_2, U_5, U_3, and U_4 which is a clear indication for the optimal investments amounts (α_i) without α_1 sorted from highest to lowest. In contrary if we don't disregard U_1, then the correct sorting of the modules by their U_is from the largest to the smallest is: U_1, U_6, U_2, U_5, U_3 and U_4 which cannot be used as an indication for the highest alphas since it assumes that module one having the largest upper limit should have the largest alpha which totally contradicts our optimal findings.

Accordingly, one could conclude that the complexity factor C_i is affecting the optimal results. Remember that a high C_i implies that the module is too simple in design and reaching its optimal performance does not demand a large part of the budget. A low C_i refers to a very complex module where improving such a module demands a huge amount of the budget. Such modules are usually the most important modules in the product.

A good comparison here would be between optimal alpha and maximum alpha. By maximum alpha we mean the investment amount which helps the module reaches its highest performance. Consider the below table:

Table 4.3: Maximum α_i vs optimal α_i

Module	Maximum α_i	Optimal α_i
M_1	100%	0.36%
M_2	12%	12%
M_3	10%	8.09%
M_4	8%	6.56%
M_5	10%	9.21%
M_6	47%	27%

Inspecting Table 4.3 we notice that module one needs all the budget to attain its highest performance that's why we are better off not investing the largest amount in module one. This clearly proves that module one is too complex and this is due to the high complexity factor C_1 assigned to it. The next complex module is M_6 where $C_6 = 0.2$ and the maximum alpha which will make module six reaches its highest performance is 47% of the budget which is less than the optimal amount assigned to it where $\alpha_6 = 27\%$. Note that if maximum alphas were assigned as optimal alphas then easily we will violate our budget constraint. That is why most of the optimal alphas should be much less than maximum alphas. In addition to that, remember that the budget also should be spread among updating modules to attain compatibility rather than just improving modules. We do not benefit if we spend higher amounts on improving each module's performance separately but once combining all the modules of the product they will not fit to each other. In this case, money would be spent and modules' performances are improved but the total product performance would remain equal to the same value before those investments.

Going back to Table 4.3, we notice that only module two has same value for maximum and optimal α_2, whereas all other modules have a maximum α_i less than optimal α_i.

Let us consider some sensitivity analysis on the value of C_1 to understand the relationship between the module's complexity and its optimal investment amount α_1. The below table shows: different values for the complexity of module one, its correspondent optimal investment amount and its rank between the modules.

Table 4.4: Sensitivity on C_1

C_1	0.01	0.05	0.06	0.07	0.15	0.2	0.3

α_1	0.36%	4.06%	5.3%	6.53%	13.59%	15%	15.03%
Rank	6	6	4	2	2	2	2

Since module one has the highest upper limit $U_1 = 70$ then it is assumed to be the most complex module having the lowest C_i. From Table 4.4 we notice that for $C_1 = 0.05$, α_1 is still the smallest between all the modules and ranked the last. When C_1 was increased to 0.06, α_1's rank increased to four and when C_1 was assigned a value larger or equal to 0.07, α_1 was ranked the second highest between the six available modules. Note that module one cannot take a value of complexity larger or equal to 0.2 since module six has the second largest upper limit value of 55 and a C_6 of 0.2. So module one cannot have a larger upper limit and a larger simplicity factor at the same time. The two last columns were introduced to say that even though module one became less complex then module six and even though module's one upper limit is larger than that of module six, still M_6 is taking the largest optimal invested amount. In all the above cases, α_6 was the largest. We can conclude that α_1 increased with the increase in C_1 but would stabilize for high values of C_1 (simple designs). This explains that assigning optimal alphas depends on both the complexity factor and the upper limit and foreshadows for some other elements interfering in making the optimal solution as such.

If we reconsider Fig. 4.1, we can notice that modules two, five and six depends on module one. So any amount α_1 spent on module one, $\alpha_1 f_{12}`(C_2/C_1)\%$ will be spent on module two, $\alpha_1 f_{15}`(C_5/C_1)\%$ will be spent on module five, and $\alpha_1 f_{16}`(C_6/C_1)\%$ will be spent on module six. And since module one is the most complex module having the smallest C_i and since its maximum alpha is 100%, then one could expect that module one should not take the largest investment amount. If α_1 was the largest then a huge amount of the budget will go to update modules two, five and six. In addition to that, even though C_1 was close to C_6 (as shown in Table 4.4), the largest alpha still goes to module six and this is due to the fact that re-designing module six will not demand any updates to other modules. In Fig. 4.1 we notice that none of the module is related to module six so we can say that module six is not visible to any other modules as opposed to module one which is visible to modules two, five and six. These results build upon the concept of "visibility" discussed by Baldwin and Clark (1999). Accordingly, one could expect that a module with a high upper limit and no other modules depending on it should take the largest percent of the budget for its performance improvement.

Rather than the optimal alphas, one should also pay attention to the analysis of the optimal θ_{ij}. From Table 4.1 we knew that it is optimally to spend money on developing design rules between modules one and five, two and three, and two and four. No advice on spending money on the design rules between module one and six and one and two since their respective f_{ij}s are initially small, thus spending money on decreasing them further is of no use. In addition to that, k_{12} and k_{16} are the smallest between all the k_{ij}, which implies the highest integrality factor and the lowest drop from f_{ij} to $f_{ij}`$. If we didn't allocate money for developing the design rules between M1M5, M2M3 and M2M4 then higher amounts would go for updating the modules due to the interdependencies between them. Consider Table 4.5 which shows the amount of money to update the modules in the absence of design rules i.e. when $f_{ij}` = f_{ij}$:

Table 4.5: $\alpha_i f_{ij}` (C_j / C_i)$ % in absence of design rules

M_i M_j	M_1 M_2	M_1 M_5	M_1 M_6	M_2 M_3	M2 M4

$f_{ij} = f_{ij}`$	0.3	0.7	0.4	0.85	0.9
$\alpha_i f_{ij}` (C_j / C_i)$	8.37%	21.04%	2.86%	11.77%	13.85%

As you notice from Table 4.5, larger amounts are needed to update the modules in the absence of design rules where 57.89% of the budget (more than half of the budget) will go to updating the modules and the rest will go for improving the performance of the modules. On the other hand, in the presence of design rules, 22.7% (refer to Table 4.2) of the budget is needed for updating the modules and 14.05% (refer to table 4.1) of the budget for developing design rules which makes a total of 36.75% (22.7% + 14.05%) which is less than the 57.89% in the absence of design rules. We can conclude that the amount of updates in the absence of design rules is always greater than the amount of updates plus the amount spent on developing design rules.

4.3. Periodic Investments Model

In the previous section, we assumed that the time component does not exist. The implicit assumption here is that the assigned budget for a specific module will be spent within the development time line at the same rate. If the company wants to investigate further its allocation decision as to how much dollars must be invested on a module at different intervals of time during the development time line (Burn rate), then a time component must be added to the earlier model.

In this section we will assume that the development process is divided into T periods and the PD (product development) managers make these periodic allocation decisions at the beginning of the process. By this we mean that, at $t = 0$ the designer makes in advance $\alpha_i(1)$, $\alpha_i(2)$, …, $\alpha_i(T)$ and $\theta_{ij}(1)$, $\theta_{ij}(2)$, …, $\theta_{ij}(T)$ where $\alpha_i(t)$ and $\theta_{ij}(t)$ denote optimal amount invested in the modules and design rules respectively for module *i* at time *t* where $0 \leq t \leq T$. Accordingly, the performance function of module *i* at time *t* will be denoted by $P_i\ (t)$ i.e. performance per period and not total performance of the module. The performance function of module *i* at time *t*, $P_i\ (t)$, will be have some additional terms reflecting the time component that was not present in the function used previously. Many researchers believe that product development performance follow an S-shaped curve (Foster, 1986) where performance build up starts slow then picks up rather quickly in the middle and then finally slows down and stabilizes for large t. Accordingly, we will assume that the performance of any product increases with time but at a decreasing rate. That is, higher levels of performance can be attained in the early periods of development than in the later periods. So $\alpha_i(t)$ will decrease with time for all the modules starting with a high $\alpha_i(1)$ and reaching zero for $\alpha_i(\infty)$. This implies that the total product performance increases at a decreasing rate in time and then stabilizes for large values of t. Accordingly P_i (t) which denotes the increment in the performance at time t will have the below functional form:

□ □(t)=Ui1+e−Ci(αi(t)Bg)− Ui2 e−t (4.3.1)

Two main objectives are achieved by this performance exponential smoothing. The first desired behavior achieved is the fact that spending a dollar on development earlier is better than later as more design freedom is still available and re-work costs are low. Second, spending a dollar over a larger development time is better than spending it over a shorter period. This reflects the crunch of time (with more time, designers can perform more experiments and tasks) and less likelihood of making errors (with less time, designers are prone to make more design errors).

To understand more equation 4.3.1, consider an available budget of $10, a duration of three periods (T=3), upper limit = 20, complexity factor = 0.3 and only one module to invest in. The below table shows different combinations for spreading the budget along the three periods (α_i (t) is shown in dollar and not in percentage), the performance $P_i(t)$ per period and the overall performance ($P_i(1) + P_i(2) + P_i(3)$) for each arrangement.

Table 4.6: Spreading $10 among one module along three periods

α_i (1)	α_i (2)	α_i (3)	$P_i(1)$	$P_i(2)$	$P_i(3)$	Overall Performance
3.33	3.33	3.33	1.7	0.6254	0.6254	2.9508
10	0	0	3.3299	0.0000	0.0000	3.3299

0	10	0	0.0000	1.2250	0.0000	1.2250
0	0	10	0.0000	0.0000	0.4506	0.4506
1	9	0	0.5477	1.1829	0.000	1.7306
9	1	0	3.2155	0.2015	0.0000	3.4170
0	9	1	0.0000	1.1829	0.0741	1.2570
4	6	0	1.9757	0.9694	0.0000	2.9451
6	4	0	2.6351	1.9757	0.0000	4.6108
6	0	4	2.6531	0.0000	0.2674	2.6040
0	6	4	0.0000	2.6351	0.3162	2.9513
5	4	1	2.3366	1.9757	0.0741	4.3864

As you notice from Table 4.6 the highest total performance goes for those combinations which have alphas that decrease by time. The lowest overall performance goes for the combinations that start with α_i (1) = 0. For example (10, 0, 0) gave a total performance of 3.33 whereas (0, 10, 0) gave a total performance of 1.22 and this is due to time pressure where investing a $10 in the first period will give a higher performance (due to the availability of more time) then a $10 spent in the second or in the last period. Ulrich and Eppinger (2008) explained that behavior by the below graph which indicates that design freedom for any product decreases with time where as the cost of change increases with time.

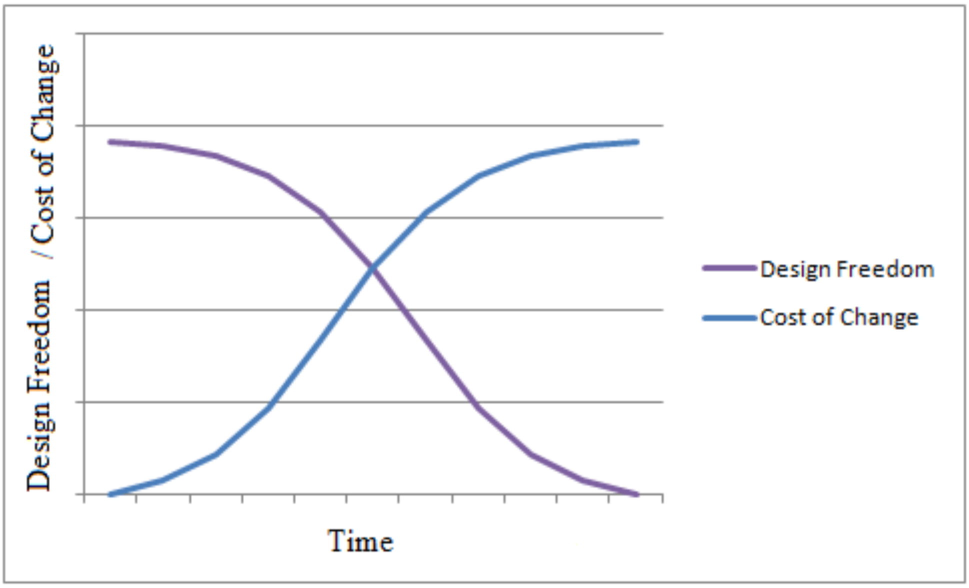


Fig. 4.3: Time pressure affecting Cost of Change and Design Freedom

Fig. 4.3 shows that the freedom of re-designing a certain module decreases with time while its respective cost increases with time. Accordingly we have chosen to multiply our performance function by e^{-t} to reflect time pressure. That is, investing a certain amount during the first few periods will have a higher performance than investing the same amount during the last periods where we have a significant cost of change and a minimal freedom of designing.

Consequently, even though the same amount of alpha (= 10) is spent in both arrangements but the factor e^{-t} in the performance function is shifting the result where $e^{-1} = 0.37$ and $e^{-2} = 0.14$ when $Ui1+e-Ci(\alpha i(t)- Ui2 = 9.0515$ in both cases. Accordingly, it is not optimal to miss the opportunity of investing in the first period, thus missing 37% (e^{-1}) from the performance. Moreover, since e^{-t} is always greater than $e^{-(t+1)}$ then in order to maximize the performance of a certain module, $\alpha_i(t)$ should be greater than $\alpha_i(t+1)$. Accordingly one can conclude that once $\alpha_i(t)$ becomes zero, then $\alpha_i(t+1)$, $\alpha_i(t+2)$, etc…

are equal to zero as well. In addition to that, Table 4.6 shows that dividing your budget among periods is better than investing the whole amount in a one period. For example, spending \$5 in the first period, \$4 in the second period and \$1 in the last period is much better than spending the whole \$10 in the first period. You will be able to spend your whole budget in a one period but you will not guarantee an optimal total performance.

In the periodic investment model, the fraction of update between modules is a recursive formula expressed by:

fij`t= fij`t−1e−θijtfij`t−1kij (4.3.2)

where the decrease in the fraction of update in period t, $f_{ij}`(t)$, depends on amount of update reached in the previous period $f_{ij}`(t-1)$.

To simplify representations, we will introduce the superscript “^{Ov}” denoting “overall” sum whether for performance, amount invested, design rules or fraction of updates as follows:

P_i^{Ov} = =*1*(*)* (4.3.3)

α_i^{Ov} = =*1*(*)* (4.3.4)

θ_{ij}^{Ov} = =*1*(*)* (4.3.5)

fij`αiCjCiOv=t=1Tfij`tαi(t)CjCi (4.3.6)

where $1 \leq i \leq n_g$ and $1 \leq g \leq m$

The optimization problem for the periodic investments model where we consider a product composed of m groups, each group contains n_g modules, T periods exist, normalized budget of 1 for each group, and an objective of maximizing total product performance in a multiple shot investments all known in advance will be as follows:

Max S_T = Max g=1mSg = Max g=1mi=1ngPigOv = Max □=□ □□=□ □□t=1TPig(t)

= Max □=□ □□=□ □□t=1T(Uig1+e−Cigαig(t) − Uig2) ∗ e−t

Subject to: □=□□i=1ng(αig)Ov+ i,j(i≠j)ng(θijg)Ov + i,j(i≠j)ngfij`gαigCjgCigOv = 1

g=1mi=1ng□=1□□□□(□) + i,j(i≠j)ng□=1□□□□□(□) +

i,j(i≠j)ng□=1□fij`g(t)αig(t)CjgCig=1 g=1mi=1ng□=1□□□□(□) + i,j(i≠j)ng□=1□□□□□(□) +

i,ji≠jng□=1□fij`gt−1e−θijgtfij`gt−1kijgαig(t)CjgCig=1

$0 \le \alpha_i \le 100$, $0 \le \theta_{ij} \le 1$

Similar to the previous section, we should check the optimality conditions and test for the concavity of the maximized function. Let us first consider the objective function:

- Max S_T= Max □=□ □□=□ □□t=1T(Uig1+e−Cigαig(t) − Uig2) ∗ e−t

Adding the time component will not affect the results of concavity derived earlier. Even though the second derivative in this case will be with respect to αig(t) rather than αig but still the second derivative is the same and still < 0 for allαig(t), implying a concave objective function.

Second we must consider the constraint and check as well its convexity, since any concave function on a convex set should have a unique global optimal.

- g=1mi=1ng□=1□□□□(□)   + i,j(i≠j)ng□=1□□□□□(□)   +
  i,ji≠jng□=1□fij`gt−1e−θijgtfij`gt−1kijgαig(t)CjgCig

Considering each term by itself we get: i=1ng□=1□□□□(□) and i,j(i≠j)ng□=1□□□□□(□) are both convex and concave for all $\alpha_i^g(t)$ and $\theta_{ij}{}^g(t)$ respectively. To prove convexity for the last term of the summation we adopt the linear version discussed in the previous section and we assume that T=2 and by induction convexity applies for all values of T especially that we have a recursive equation of $f_{ij}{}^{`g}(t)$.

For T = 2 and the linear version assumption we have:

fij`gt=1αig(t=1)CjgCig+fij`gt=2αig(t=2)CjgCig

⇒□□□□1−kijgfijgθijg(t=1)αig(t=1)CjgCig+fij`g(t=1)1−kijgfijgθijg(t=2)αig(t=2)CjgCig

We take again each term separately. The first term (when t=1) is already proven concave in the one shot investment, then we are rest with proving concavity for the second term and then multiplying the whole constraint with a negative sign to obtain a convex constraint.

fij`g(t=1)1−kijgfijgθijg(t=2αig(t=2)CjgCig=fijg1−fijgkijgθijg(t=1)1−fijgkijgθijg(t=2)αi g(t=2)CjgCig

Dropping out the constants, and deriving with respect to: αig(t=2)CjgCig , θijg(t=1), and θijg(t=2) we get the below 3x3 Hessian matrix.

H = 0−□□□□□□□□+ □□□□□□□2□□□□(□=2) □□□□□□□□+

□□□□□□□□2□□□□(□=1) −□□□□□□□□+

□□□□□□□□2□□□□(□=2) 0□□□□□□□□2□□□(□=2) □□□□□□□□+

□□□□□□□□2□□□□(□=1) □□□□□□□□2□□□(□=2) 0

and x^THx = □1□2□3H□1□2□3= 2□1□2−□□□□□□□□+ □□□□□□□□2□□□□□=2+ 2□1□3□□□□□□□□+ □□□□□□□□2□□□□□=1+ 2□2□3□□□□□□□□2□□□□=2>0 for all x ≫□□□□□□

x^THx being greater than zero, it implies convexity. Therefore, we can conclude that part of the periodic constraint is concave and part is convex. Since decision variables exist in both parts, no comparison or further work can be done.

4.3.1. Illustrative Example

Consider the same product in the previous example which was composed of only one group and six modules related together based on Fig. 4.1 and based on the data matrix of Fig. 4.2. We will assume in this example that T=3, i.e. we have three periods of investments and we are seeking optimal investments decisions in advance of the three periods.

4.3.1.1. Excel Solver Results

Using Excel-Solver, we optimized the above mentioned example during three time periods and we got an optimal total product performance of 96.76 and the optimal investments amounts in design rules and modules are shown in the below table:

Table 4.7: Decision variables values after optimization

	t = 1	t = 2	t = 3
α_1 (t)	0 %	0 %	0 %
α_2 (t)	5.87 %	4.42 %	2.85 %
α_3 (t)	4.45 %	3.14 %	1.77 %
α_4 (t)	3.38 %	2.15 %	1.03 %
α_5 (t)	4.93 %	3.54 %	2.06 %
α_6 (t)	15.14%	15.14%	15.14%
θ_{12} (t)	0 %	0 %	0 %
θ_{15} (t)	0 %	0 %	0 %
θ_{16} (t)	0 %	0 %	0 %
θ_{23} (t)	4.42%	0 %	0 %
θ_{24} (t)	4.72%	0 %	0 %

From Table 4.7 we notice that around 85% of the budget went to re-designing modules, 9.14% for developing design rules and the 5.86% left are for updating modules to attain compatibility with the changes done. We notice as well that module's six investments were the highest between all the modules among all the periods where 15.14% from the budget each period should be spent on improving module's six performance. On the contrary, the lowest alphas are that of module one where none of the money should be spent in re-designing it. Concerning the rest of the modules (two, three, four, and five), all their alphas decrease by time and this is due to the performance function $P_i(t)$ assumed in the previous section. If we sum the alphas for each module along t, we see that 45.42% of the budget is invested in module six, around 13% in module 2, 10.5%, 9.35%, 6.55% and 0% in modules five, three, four and one respectively.

Concerning design rules, since no investments plans are advised for module one along the periods, then no efforts should be exerted to reduce the interdependency between modules one and two, one and five, and one and six. 4.72% of the budget must be spent on developing design rules between modules two and four, and 4.42% must be spent to decrease the dependency between modules two and three.

To emphasize the importance of the design rules, the reduction in the fraction of updates from f_{ij} to $f_{ij}`(t)$ and the amounts that should be spent to update module $j$ $(\alpha_i f_{ij}`(t)$ $(C_j / C_i))$ once changes are applied to module i should be taken into consideration. Consider the below table:

Table 4.8: $f_{ij}(t)$ *vs* $f_{ij}`(t)$ *and* $\alpha i\, f_{ij}`(t)$ (C_j / C_i)

M_i M_j	M_1 M_2	M_1 M_5	M_1 M_6	M_2 M_3	M2 M4
f_{ij}	0.3	0.7	0.4	0.85	0.9
$f_{ij}`$ (t=1)	0.3	0.7	0.4	0.2	0.16
$f_{ij}`$ (t=2)	0.3	0.7	0.4	0.2	0.16

$f_{ij}`$ (t=3)	0.3	0.7	0.4	0.2	0.16
$\alpha_i f_{ij}`(t=1)\ (C_j / C_i)$	0 %	0 %	0 %	1.38 %	1.24 %
$\alpha_i f_{ij}`(t=2)\ (C_j / C_i)$	0 %	0 %	0 %	1.04 %	0.93 %
$\alpha_i f_{ij}`(t=3)\ (C_j / C_i)$	0 %	0 %	0 %	0.67%	0.6 %

Table 4.8 shows that $f_{ij}`(t)=f_{ij}$ in the absence of design rules where θ_{12}, θ_{15} and $\theta_{16} = 0$. We notice a huge decrease in the fraction of update between modules two and four, where prior of investing in design rules, 90% ($=f_{24}$) of module four must be re-designed to attain compatibility with module's two changes while only 16% ($=f_{24}`$) now must be re-designed after spending part of the budget on developing design rules which clearly decreased the dependency between these two modules. The fraction of re-work between modules two and three decreased from 85% to 20%. Since no investments in design rules is witnessed in periods two and three then: $f_{ij}`(3) = f_{ij}`(2) = f_{ij}`(1)$. The last three rows in Table 4.8 show the percentages of the budget that should be assigned to update the dependent modules after investing the optimal alphas amounts in improving the modules' performances. Since the alphas are decreasing by time, then definitely the updates' amounts will diminish as well even though the fraction of re-works of periods two and three are the same as that of period one. Moreover, since α_1 is zero, then all updates related to module one are zero as well.

4.3.1.2. Analysis

Similar to the previous analysis section, we will start first by sorting the optimal alphas in a descending order where the first listed alpha refers to the highest amount invested in a certain module and the last listed alpha refers to the least amount of investment between the modules. Since we did not change the data matrix nor the architecture of the product between the two examples, then one could expect to get the same order of alphas we previously obtained even though the amounts of performances and investments differ (this is due to time component introduction). The sorted alphas are as follows: α_6, α_2, α_5, α_3, α_4, and α_1 and they are exactly the same as example 4.2.1. The highest investment amount went to module six which has the second highest upper limit $U_6 = 55$ and the lowest investment amount went to module one which has the highest upper limit $U_1 = 70$. The same conclusion applies here, which highlights the cause on the complexity factor C_i which is affecting the optimal results.

Note that we cannot directly compare the periodic model to the one shot model due to the difference in the objective function; however, we can have a relative comparison. As you see from Table 4.9, there is difference in the total product performance between the two types of investments and this due to α_6 being 45.42% in the periodic investments as opposed to 27% in the one-shot model. This is due to that fact that we are multiplying P_i by e^{-1}, e^{-2} and e^{-3} thus more investment amounts are needed to achieve the same performance of the deterministic model. Since module six is the most important module, more dollars were assigned to that module as opposed to approximate same amounts of dollars to the other modules. The important fact is that the sorting of alphas is the same between the two models but only the amount invested in module six differs. There is no much benefit from having multiple investment time points since we can perfectly predict the performance estimation of modules.

Table 4.9: Comparison between one-shot and periodic investments

	One- Shot Investment	Periodic Investments
α_1 / P_1	0.36% -- 0.06	0% -- 0
α_2 / P_2	12% -- 19	13% -- 10.02
α_3 / P_3	8.09% -- 8.49	9.35% -- 4.31
α_4 / P_4	6.56% -- 3.99	6.55% -- 1.89
α_5 / P_5	9.21% -- 10.99	10.5% -- 5.64
α_6 / P_6	27% -- 27.35	45.42% -- 74.88
S_T	69.78%	96.76
□=16□□	63.23%	85%
□,□=1 □≠□ 6□□□	14.06%	9.14%
□,□=1 □≠□ 6□□□`□□(□□□□)	22.71%	5.86%

Similar studies concerning sensitivity on *C_i* and comparison between optimal and maximum alphas could be done and will lead to the same result derived earlier which states that a module with a high upper limit and no other modules depending on it is expected to take the largest percent of the budget for its performance improvement. Being highly complex, module one will demand the entire budget to attain its maximum alpha and will require a high percentage of the budget for updating modules two, five and six. One can conclude then, that three factors must be taken into consideration once deciding on investing in a certain module: the upper limit, complexity factor and architectural link between modules.

In addition to that, one should also pay attention to the analysis of the optimal θ_{ij}. From Table 4.7 we knew that it is optimal to spend money on developing design rules between modules two and three, and two and four. However, it is not optimal to spend money on the design rules between modules one and two, one and six and one and five. Since it is not optimal to invest in module one ($\alpha_1 = 0$) so no need to update any modules dependent on it. If we did not allocate money for developing the design rules between M2M3 and M2M4 then higher amounts would go for upgrading the modules due to the interdependencies between them. Consider Table 4.10 which shows the amount of money to update the modules in the absence of design rules i.e. when $f_{ij}`(t) = f_{ij}$:

Table 4.10: $\alpha_i f_{ij}`(t)$ (C_j / C_i) % in absence of design rules

M_i M_j	M_1 M_2	M_1 M_5	M_1 M_6	M_2 M_3	M2 M4
$f_{ij} = f_{ij}`(t)$	0.3	0.7	0.4	0.85	0.9
$\alpha_i f_{ij}`(t=1)$ (C_j / C_i)	0 %	0 %	0 %	5.76 %	6.77 %
$\alpha_i f_{ij}`(t=2)$ (C_j / C_i)	0 %	0 %	0 %	4.34 %	5.11 %
$\alpha_i f_{ij}`(t=3)$ (C_j / C_i)	0 %	0 %	0 %	2.8 %	3.29 %

As you notice from Table 4.10, larger amounts are needed to update the modules in the absence of design rules where 28.07% of the budget will go to updating the modules and the rest will go for improving the performance of the modules. On the other hand, in the presence of design rules, 5.86% (refer to Table 4.8) of the budget is needed for updating the modules and 9.14% (refer to table 4.7) of the budget for developing design rules which makes a total of 15% (5.86% + 9.14%) which is less than the 28.07 % in the absence of design rules.

Concerning the total product performance, at t=1 we have a total performance of 40.12, at t=2 we have an increment of 30.17 making the total performance at time two equals to 70.28, and at t=3, we have an increment of 26.48 making the cumulative total product performance equals 96.76. As discussed in the previous section, the product performance will evolve at a decreasing rate where for a large T we expect minimal increments, thus stabilization for product performance and no more improvements. Consider the below two graphs which show the growth of the modules' performances and total product performance as a function of time for the above three periods' example:

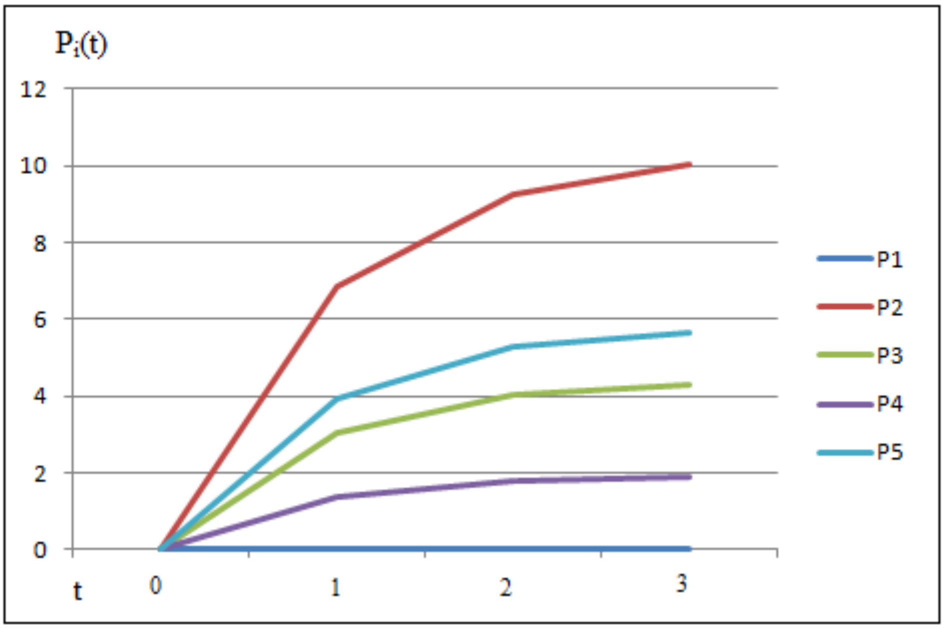


Fig.4.4: Modules' performances in function of time

Fig. 4.4 shows the performances of modules one to five as a function of time. As you notice the performance of any module will increase with time. The product highly reacts in the first period and then enters a steady state once t gets larger. The below graph shows: the total product performance and the performance of module six as a function of time. Since module six affects mostly the total performance, it is better to group them in one figure.

Fig. 4.5 below shows the Total product performance increasing at a decreasing rate and shows as well the performance of module six how it is constantly increasing depending on the optimal equal alphas invested in each period.

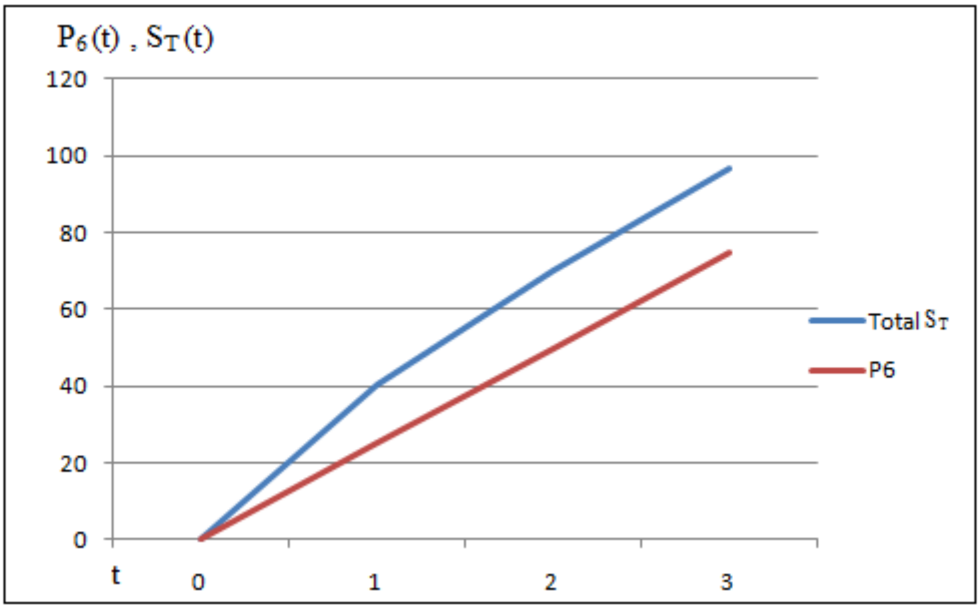


Fig. 4.5: Total product and module's six performances in function of time

After establishing the deterministic model through two types of investments (one shot and periodic), we will move next to develop the stochastic model in those two investments strategies as well. The only assumption made in this chapter was that concerning the periodic investments model where the performance was assumed to increase at a decreasing rate.

CHAPTER 5

STOCHASTIC MODEL

5.1. Overview

As discussed previously, uncertain modules may exist in a given product and are expected to have a higher performance than any other certain module for the same amount of investment. Thus they are considered to be the top features of the product causing a high market capturing rate. In this chapter, the notion of uncertainty will be introduced into the two types of investments: one shot investment and periodic investments. Similar to the earlier chapter, illustrative examples followed by analysis will be provided to ensure full knowledge of stochastic events.

5.2. One Shot Investment Model

In the last chapter, all modules whether complex or simple in design had a deterministic and fixed known U_i denoting the upper limit value the performance function can attain. But as we saw in the Dell case-study, not all modules have *proven technologies* where they provide limited but guaranteed performance. Some modules are considered to be a *prospective technology* choice where they offer high but risky performance improvement. So "certain" modules would yield low but guaranteed performance improvement where as "uncertain" modules would yield high but not guaranteed improvements. Since complex design modules having low C_is are assumed to achieve higher U_is, complex modules then are the ones which likely hold uncertainty in their performances. Simple modules cannot be uncertain, because the designer would definitely not choose to spend money on re-designing a low and uncertain U_i.

For uncertain modules, we will assume that U_i will vary based on a known probability distribution. In this thesis we will assume a Uniform distribution function: U~ (a_i, b_i) where a_i and b_i are the minimum and maximum values for the upper limit value U_i of module *i*. Accordingly, the performance of any group would be the sum of all expected P_is present in that group:

□□=□=□ □□□(□□) (5.2.1)

For example, if module i was considered to be a risky module having an upper limit U_i where U_i is a Uniform random variable: U(a_i,b_i), then an expected value of the upper limit U_i of module *i* will be used and it is equal to $(a_i+b_i)/2$. Note that module *i* has the same chance of attaining a_i, b_i or any value in between that’s why uncertain modules are risky: as they can attain larger performances than certain modules, they are equally probable to behave worse and attain lower performances.

□ □□=□ Ui 1+e−Ciαi – Ui2 = EUi 1+e−Ciαi – EUi2

= (ai+bi)2 1+e−Ciαi – (ai+bi)22

(5.2.2)

Note that if module *i* is a deterministic module then E (P_i) = P_i. Note as well that k_{ij} can be stochastic also, where the designer’s knowledge of the relationship between the modules can vary uniformly between (k_{ij}^{low}, k_{ij}^{high}) as well thus E (f_{ij}`) can be used which is equal to: E (fij`)=fijg(e−θijgfijgE(kijg)). E(k_{ij}) is equal to (k_{ij}^{low} + k_{ij}^{high}) /2.

In the stochastic model, our objective will not only be maximizing total product performance but rather as well minimizing the variability. The notation V_g will be used to denote the variance of group *g* and V_i to denote the variance of module i. We assume that groups are independent of each other, then no correlation exists between various groups in the system. Within a certain group, modules do depend on each other but we also assume that no correlation exists between the upper limits of the modules. Thus the variance of a certain group will be the sum of all the modules' variances only.
The variance of group g and variance of module *i* will be expressed as follows where $1 \leq g \leq m$ and $1 \leq i \leq n_g$:

□□= □=*1*□□□□ (5.2.3)

□□=□ □□*2*−*[*□ □□*]2* (5.2.4)

where $E(U_i) = (a_i + b_i)/2$ and $E(U_i^2) = [(b_i - a_i)^2/12] + [(a_i + b_i)/2]^2$

For formulating the optimization problem, we will consider a product composed of m groups and each group contains n_g modules some of them certain and others risky with a $U_i \sim U(a_i, b_i)$. Our objective is to maximize total product performance and minimize the variance in a one shot investment where no time component is present. For the objective function to have a unique performance unite, the standard deviation multiplied by a weight w_i will be subtracted from each modules' performance instead of the variance. The designer can choose w_i ($0 \leq w_i \leq 1$) depending on his preferences where $w_i = 0$ cancels the objective of minimizing variability and $w_i = 1$ ensures full minimization of risk (standard deviation is a measure of risk). Then the standard deviation of a certain group will be the sum of all standard deviations multiplied by their respective weights for all the modules present in that group:

□□.□□□.= □□= □=1□□□□ □□ (5.2.4)

A normalized budget of 1 for each group is considered: $B_g = 1$ for $1 \leq g \leq m$. Since some modules are uncertain then E (P_i) is always used. Remember that the superscript *g* will be used to differentiate between the modules' parameters among various groups:

Max S_T = Max g=1mSg− Vg = Max □=□ □Vigwig

= Max□=□ □E (Pig)2−[E Pig]2wig

Let □□+□−□□□□□□−□□=□□□ then:

- □□□=□□□□□+□−□□□□□□−□□= □□□□□□
- □□□□= □□□+□□□□ □□□

. □ (□□□)2= □□□− □□□212+□□□+□□□22□□□2

. [□ □□□]2= □□□+□□□22□□□2

Replacing all the above in the objective function we get:

Max S_T = Max □=□ □□=□ □□□□□+□□□□□□□□−□□□−□□□□□□□□□□□

= Max□=□ □□=□ □□□□□□□□□□−□□□□+□□□□+□□□□

= Max□=□ □ □□+□□−□□□□□□−□□□□□□−□□□□+□□□□+□□□□

Subject to: □=1□i=1ngαig + i,j(i≠j)ngθijg + i,j(i≠j)ngfij`g αigCjgCig =1

□=1□i=1ngαig + i,j(i≠j)ngθijg + i,j(i≠j)ngfijg(e−θijgfijgkijg) αigCjgCig

=1

$0 \leq \alpha_i \leq 100, \quad 0 \leq \theta_{ij} \leq 1$

Note that for a certain module, $a_i = b_i = U_i$ and no variability exists ($w_i = 0$), that is:

□□+□□−□□□□□□□−□□□□□□□−□□□□+□□□□+□□□□

=□□□□□□+□□−□□□□□□−□□=□□□□□+□−□□□□□□−□□=□□□

Since we are maximizing the objective function, it is of vital importance to check the optimality conditions and test for the concavity of the maximized function and convexity of constraint. Let us first consider the objective function:

- Max S_T= Max□=□ □ □□+□□−□□□□□□□−□□□□□□□−□□□□+□□□□+□□□□

Since any weighted sum of a concave function is concave, then after omitting the constants, it is enough to show that 1 2+2e−Cigαig is concave for any α_i^g by showing that second order derivative with respect to α_i^g is nonpositive. Taking the second derivative we get the following:

□2□(□□□)21 2+2e−Cigαig =Cig2e−Cigαige−Cigαig−121+e−Cigαig3 <0 for all α_i^g

Therefore, we can conclude that the objective function is concave for all α_i^g.

Concerning the constraint, the same proof of the one shot investment model applies here and implies a concave constraint.

After defining the model and testing the optimality conditions, we move next to an illustrative example.

5.2.1. Illustrative Example

We will consider the same example of chapter 4, the product which is composed of only one group having six modules. The modules are dependent on each other based on the architectural diagram provided earlier (Fig. 4.1). The data matrix of the previous chapter still hold as well except for the upper limit of modules one and six which are assumed to be uncertain having uniform distributions: U_1~ U (40, 120) and U_6 ~ (30, 85). Since only complex modules can have risky performances, then the other modules (two, three, four and five) are certain and have the same upper limit defined previously (Fig. 4.2). We will assume that the designer's objective is to maximize total product performance and fully minimize the variability in one shot investment. Accordingly we will set $w_1 = w_6 = 1$.

5.2.1.1. Excel Solver Results

Using Excel-Solver, we optimized the above mentioned example based on the formulation provided in the earlier section where uncertainty and reducing variability were taken into account. We got an optimal total product performance of 63.30 and the optimal investments amounts in design rules and modules are shown in the below table:

Table 5.1: Decision variables values after optimization

α_1	0.02%	θ_{12}	0%

α_2	11.95%	θ_{15}	0 %
α_3	10.07%	θ_{16}	0%
α_4	8.43%	θ_{23}	4.12%
α_5	10.99%	θ_{24}	4.45%
α_6	42.25%		

From Table 5.1 we notice that around 83.71% of the budget went to re-designing modules, 8.57% for developing design rules and the 7.72% left are for updating modules to attain compatibility with the changes done. We notice as well that re-designing module six demanded 42.25% from the budget followed by module two where 11.95% of the budget was allocated for its performance improvement work. Modules five, three and four took from the budget 10.99%, 10.07% and 8.43% respectively. The lowest alpha is that of module one where only 0.02% of the budget should be spent in re-designing it which is a very minimal amount and can be omitted. Concerning design rules, no work should be done to reduce the interdependency between modules that are linked to module one as modules two, five and six. 4.45% of the budget must be spent on developing design rules between modules two and four, and 4.12% must be spent to decrease the dependency between modules two and three.

Other important results are the reduction in the fraction of updates from f_{ij} to $f_{ij}`$ and the amounts that should be spent to update module j once changes are applied to module i: $\alpha_i f_{ij}` (C_j / C_i)$.

Table 5.2: f_{ij} *vs* $f_{ij}`$ *and* $\alpha i f_{ij}` (C_j / C_i)$

M_i M_j	M_1 M_2	M_1 M_5	M_1 M_6	M_2 M_3	M2 M4
f_{ij}	0.3	0.7	0.4	0.85	0.9
f_{ij}`	0.3	0.7	0.4	0.22	0.18
$\alpha_i f_{ij}$` (C_j / C_i)	0.48%	1.2%	0.16%	3.10%	2.78%

Table 5.2 shows that f_{ij}`$= f_{ij}$ in the absence of design rules where θ_{12}, θ_{15} and θ_{16} = 0. We notice a huge decrease in the fraction of update between modules two and four, where prior of investing in design rules, 90% (=f_{24}) of module four must be re-designed to attain compatibility with module's two changes while only 18% (=f_{24}`) now must be re-designed after spending part of the budget on developing design rules which clearly decreased the dependency between these two modules. The fraction of re-work between modules two and three decreased from 85% to 22%. The last row in Table 5.2 shows the percentages of the budget that should be assigned to update the dependent modules after investing the optimal alphas amounts in improving the modules' performances.

5.2.1.2. Analysis

Similar to the previous chapter, if we want to sort the optimal alphas in a descending order then we will get the same order of alphas we previously obtained even though the upper limit U_1 and U_6 changed. What is different in this model is the huge amount assigned to re-designing the modules where 83.71% of the budget was allocated for investments compared to 63.23% for the deterministic model. When the upper limit U_1 varied uniformly between 50 and 120, no advice was given to invest in such a risky module especially that it has a high complexity factor and many architectural links with other modules. In contradiction to module one, when its upper limit U_6 varied uniformly between 30 and 85 more budget was allocated for the investments in module six. When module six was certain, it was optimal to invest in it 27% of the budget, and when it became risky the optimal amount increased to 42.25%. In the deterministic module, since none of the modules depend on module six, and since U_6 was large in value it was advisable to invest the largest part of the budget in re-designing module six. Now in the stochastic model, we notice that even though module six became uncertain more money was allocated to such a module. The reason for that lies also in the non-existence of modules depending on module six but moreover on the upper limit of module six where it is equally probable to attain any value between 30 and 85 with an expected value of 57.5, the second largest between all the modules and larger than U_6 of the deterministic case. Accordingly more budget will go to module six and less for module one which implies less for design rules and updates since most of the modules depend on module one. Consider the below table which shows a comparison between the deterministic and the stochastic one-shot model results:

Table 5.3: Comparison between deterministic and stochastic one-shot model

	Deterministic One- Shot Investment	Stochastic One-Shot Investment
α_1	0.36%	0.02%
α_2	12%	11.95%
α_3	8.09%	10.07%
α_4	6.56%	8.43%

α_5	9.21%	10.99%
α_6	27%	42.25%
S_T	69.78	63.30
□=16□□	63.23%	83.71%
□,□=1 □≠□ 6□□□	14.06%	8.57%
□,□=1 □≠□ 6□□□`□□(□□□□)	22.71%	7.72%

If we compare the total product performance between the deterministic and the stochastic model from Table 5.3 we notice that the deterministic attained larger total product performance even though less investments amounts were allocated to its modules. This can be explained by the fact that we are minimizing variability in the stochastic model which implies subtracting the weighted standard deviation from each module's performance. More money went to modules' re-design stage but less total product performance reached. The difference is not huge (69.78 vs. 63.30) but still significant if the amount spent on the modules was taken into consideration.

Similar table to that of the previous chapter can be drawn to compare the optimal and maximum alphas. All maximum alphas in Table 4.3 hold except for that of module six, where its maximum alpha now equals 44 and optimal alpha 42.25. We notice that in this model all the optimal alphas were approximately equal to their maximum alphas except for module one. And this is due to the lack of investment in module one which increases the available budget for the others modules. Remember that if we did not invest in module one, then we will not need to spend much money on developing design rules and updating modules for attaining compatibility.

Similar sensitivity study on the value of C_1 can be done, and same results will be obtained; where for $C_1 \geq 0.07$, module one will be ranked the second largest between all the alphas. But as said previously, module one cannot decrease in complexity (C_1 increases) and still attain the same high upper limit U_1.

Rather than the analysis of the optimal alphas, we can pay attention as well to the analysis of the optimal θ_{ij}. From Table 5.1 we knew that it is optimally to spend money on developing design rules only between modules two and three, and two and four. No advice on spending money on the design rules related to module one since it is not optimal to invest in the risky module one at all. If we didn't allocate money for developing the design rules between M2M3 and M2M4 then higher amounts would go for updating the modules due to the interdependencies between them. Consider Table 5.4 which shows the amount of money to update the modules in the absence of design rules i.e. when $f_{ij}` = f_{ij}$:

Table 5.4: $\alpha_i f_{ij}` (C_j / C_i)$ % in absence of design rules

M_i M_j	M_1 M_2	M_1 M_5	M_1 M_6	M_2 M_3	M2 M4
$f_{ij} = f_{ij}`$	0.3	0.7	0.4	0.85	0.9
$\alpha_i f_{ij}` (C_j / C_i)$	0.48%	1.2%	0.16%	11.72%	13.79%

As you notice from Table 5.4, larger amounts are needed to update the modules in the absence of design rules where 27.35% of the budget will go to updating the modules and the rest will go for improving the performance of the modules. On the other hand, in the presence of design rules, 7.72% (refer to Table 5.2) of the budget is needed for updating the modules and 8.57% (refer to table 5.1) of the budget for developing design rules which makes a total of 16.29% (7.72% + 8.57%) which is less than the 27.35% in the absence of design rules. One more time the power of design rules in decreasing the amount spent on upgrading modules and increasing the available budget for investments in the modules is being shown. By comparing these results to that of the deterministic model, we notice that the role of the design rules is extremely important in the deterministic example and less important in the stochastic one and this due to omitting investments in the uncertain module (M1) thus omitting developing design rules (M1M2, M1M5, and M1M6).

After illustrating the one shot stochastic model, similar work will done to the periodic model presented in the below section.

5.3. Periodic Investments Model

In this section both time and uncertainty components will be introduced to our model. It will be assumed that T periods exist in the development process and the designer must be aware of all his periodic optimal investments decisions prior to starting re-designing his modules. All the equations derived in section 4.3 concerning $P_i\ (t), f_{ij}^{`}\ (t), P_i^{Ov}, \alpha_i^{Ov}, \theta_{ij}^{Ov},$ and $(f_{ij}^{`}\alpha_i C_j/C_i)^{Ov}$ hold. Concerning uncertainty, it is still assumed that only complex modules can be risky in performance where U_i will vary based on a Uniform distribution function: U~ (a_i, b_i) where a_i and b_i are the minimum and maximum values for the upper limit value U_i of module i.

Our objective in this section will be to maximize total product performance and minimize variability in multiple investments periods. The optimization problem for the stochastic periodic investments model where we consider a product composed of m groups, each group contains n_g modules some of them certain and others risky with a $U_i \sim U(a_i, b_i)$, T periods exist, normalized budget of 1 for each group, and an objective of maximizing total product performance and minimizing weighted standard deviation in multiple shots investments all known in advance will be as follows:

MaxST=Max= (=Max= = t=1T (Vigtwig(t)

=Max = = t=1T E (Pig(t))2−[E Pig(t)]2

Let +−− = then:

- = +−()− −= −

. = + −

. ()2= − 212++22−2

. 2= +22−2

Replacing all the above in the objective function we get:

Max S_T = Max= + −−− −

= Max= −−+ +

= Max= −+−()− −+ +

Subject to: =i=1ng(αig)Ov+ i,j(i≠j)ng(θijg)Ov + i,j(i≠j)ngfij`gαigCjgCigOv = 1

⍰=1⍰i=1ng⍰=1⍰⍰⍰⍰(⍰)   + i,j(i≠j)ng⍰=1⍰⍰⍰⍰⍰(⍰)   +

i,j(i≠j)ng⍰=1⍰fij`g(t)αig(t)CjgCig=1 ⍰=1⍰i=1ng⍰=1⍰⍰⍰⍰(⍰)   + i,j(i≠j)ng⍰=1⍰⍰⍰⍰⍰(⍰)   +

i,ji≠jng⍰=1⍰fij`gt−1e−θijgtfij`gt−1kijgαig(t)CjgCig= 1

$0 \le \alpha_i \le 100, \quad 0 \le \theta_{ij} \le 1$

Similarly to the previous section, we should check the optimality conditions and test for the concavity of the maximized function and constraint. Let us first consider the objective function:

- Max S_T= Max⍰=⍰ ⍰⍰−⍰⍰⍰+⍰⍰−⍰⍰⍰⍰⍰⍰(⍰)− ⍰⍰⍰⍰⍰⍰−⍰⍰⍰⍰+⍰⍰⍰ ⍰+⍰⍰⍰⍰

Adding the time component will not affect the results of concavity derived earlier. Even though the second derivative in this case will be with respect to αig(t) rather than αig but still the second derivative is the same and still < 0 for allαig(t), implying a concave objective function.

Concerning the constraint, the same proof of the periodic deterministic investments model applies here.

After defining the model and testing the optimality conditions, we move next to an illustrative example.

5.3.1. Illustrative Example

We will continue with the same example of the product composed of only one group and six modules related together based on Fig. 4.1. The data matrix of the previous chapter still hold as well except for the upper limit of modules one and six which were assumed in the previous section to be uncertain having uniform distributions: $U_1 \sim U\ (40, 120)$ and $U_6 \sim (30, 85)$. Since only complex modules can have risky performances, then the other modules (two, three, four and five) are certain and have the same upper limit defined previously (Fig. 4.2). We will assume that the designer's objective is to maximize total product performance and fully minimize the variability in one shot investment. Accordingly we will set $w_1 = w_6 = 1$.We will assume that T=3, i.e. we have three periods of investments and we are seeking optimal investments decisions in advance for the three periods.

5.3.1.1. Excel Solver Results

Using Excel-Solver, we optimized the above mentioned example during three time periods where uncertainty and minimizing variability where taken into consideration. An optimal total product performance of 32.79 was obtained and the optimal investments amounts in design rules and modules are shown in the below table:

Table 5.5: Decision variables values after optimization

	t = 1	t = 2	t = 3
α_1 (t)	0 %	0 %	0 %
α_2 (t)	6.05 %	4.73 %	3.33 %
α_3 (t)	4.89 %	3.73 %	2.47 %
α_4 (t)	3.73 %	2.64 %	1.29 %
α_5 (t)	5.47 %	4.24 %	2.92 %
α_6 (t)	18.85%	13.44%	6.8%
θ_{12} (t)	0 %	0 %	0 %
θ_{15} (t)	0 %	0 %	0 %
θ_{16} (t)	0 %	0 %	0 %
θ_{23} (t)	4.64%	0 %	0 %
θ_{24} (t)	4.91%	0 %	0 %

From Table 5.5 we notice that around 84.58% of the budget went to re-designing modules, 9.55% for developing design rules and the 5.87% left are for updating modules to attain compatibility with the changes done. We notice as well that module’s six investments were the highest between all the modules among all the periods where approximately 39% from the budget was allocated for improving module’s six performance. On the contrary, the lowest alphas are that of module one where none of the money should be spent in re-designing it. Concerning the rest of the modules (two, three, four, and five), all their alphas decrease by time and this is due to the performance function $P_i\,(t)$ assumed in the earlier chapter. If we sum the alphas for these modules along t, we see that around 14.11% should be spent on re-designing module 2, 12.63%, 11.09%, and 7.66% modules five, three, and four respectively.

Concerning design rules, since no investments plans are advised for module one along the periods, then no efforts should be exerted to reduce the interdependency between modules one and two, one and five, and one and six. 4.91% of the budget must be spent on developing design rules between modules two and four, and 4.64% must be spent to decrease the dependency between modules two and three.

Similar to other sections, we will show, the reduction in the fraction of updates from f_{ij} to $f_{ij}`(t)$ and the amounts that should be spent to update module $j$ $(\alpha_i f_{ij}`(t)\ (C_j / C_i))$ once changes are applied to module i. Consider the below table:

Table 5.6: $f_{ij}(t)$ *vs* $f_{ij}`(t)$ *and* $\alpha i f_{ij}`(t)\ (C_j / C_i)$

$M_i\ M_j$	$M_1\ M_2$	$M_1\ M_5$	$M_1\ M_6$	$M_2\ M_3$	M2 M4
f_{ij}	0.3	0.7	0.4	0.85	0.9
$f_{ij}`$ (t=1)	0.3	0.7	0.4	0.19	0.15
$f_{ij}`$ (t=2)	0.3	0.7	0.4	0.19	0.15
$f_{ij}`$ (t=3)	0.3	0.7	0.4	0.19	0.15
$\alpha_i f_{ij}`(t=1)\ (C_j / C_i)$	0 %	0 %	0 %	1.33 %	1.19 %
$\alpha_i f_{ij}`(t=2)\ (C_j / C_i)$	0 %	0 %	0 %	1.04 %	0.93 %

$\alpha_i f_{ij}'(t=3)\ (C_j / C_i)$	0 %	0 %	0 %	0.73%	0.65 %

Table 5.6 shows that $f_{ij}'(t)=f_{ij}$ in the absence of design rules where θ_{12}, θ_{15} and θ_{16} = 0. We notice a huge decrease in the fraction of update between modules two and four, where prior of investing in design rules, 90% (=f_{24}) of module four must be re-designed to attain compatibility with module's two changes while only 15% (=f_{24}') now must be re-designed after spending part of the budget on developing design rules which clearly decreased the dependency between these two modules. The fraction of re-work between modules two and three decreased from 85% to 19%. Since no investments in design rules is witnessed in periods two and three then: $f_{ij}'(3) = f_{ij}'(2) = f_{ij}'(1)$. The last three rows in Table 5.5 show the percentages of the budget that should be assigned to update the dependent modules after investing the optimal alphas amounts in improving the modules' performances. Since the alphas are decreasing by time, then definitely the updates' amounts will diminish as well even though the fraction of re-works of periods two and three are the same as that of period one.

4.3.1.2. Analysis

Similar to the previous analysis sections, if we want to sort the optimal alphas in a descending order then we will get the same order of alphas we previously obtained even though the upper limit U_1 and U_6 changed and periodic investment is assumed.

We notice in this chapter that the amounts allocated to re-designing the modules, investing in design rules and updating the dependent modules are approximately the same as the deterministic periodic model but with less total product performance. Consider the below table which compares the periodic investments between the deterministic and the stochastic model.

Table 5.7: Comparison between deterministic and stochastic investments

	Deterministic Periodic Investments	Stochastic Periodic Investments
α_1	0%	0%
α_2	13%	14.11%
α_3	9.35%	11.09%
α_4	6.55%	7.66%
α_5	10.5%	12.63%
α_6	45.42%	39%
S_T	96.76	32.79
⍰=16⍰⍰	85%	84.58%
⍰,⍰=1 ⍰≠⍰ 6⍰⍰⍰	9.14%	9.55%
⍰,⍰=1 ⍰≠⍰ 6⍰⍰⍰`⍰⍰(⍰⍰⍰⍰)	5.86%	5.87%

If we compare the total product performance between the deterministic and the stochastic model for the periodic type investments, we notice that the deterministic attained a significant larger total product performance (96.76) compared to the stochastic model (32.79) even though identical investments amounts were allocated to the modules. This can be explained first by the fact that we are minimizing variability in the stochastic model which implies subtracting the weighted standard deviation from each module's performance. Remember that w_6 was assumed to be equal to 1 which resulted in a total performance for module six equal to 10.38 compared to 74.88 in the deterministic case. Second in the stochastic model, U_6 can vary between 30 and 85 with equal probability compared to certain upper limit of 55 in the deterministic case. Accordingly when the same amount of money is spent on developing and updating the modules, the deterministic model (in our example assumed) attained larger total product performance.

Similar studies concerning sensitivity on C_i and comparison between optimal and maximum alphas could be done and will lead us to the same result derived earlier which states that a module with a high upper limit and no other modules depending on it is expected to take the largest percent of the budget for its performance improvement. Being highly complex and uncertain, module one will demand the entire budget to attain its maximum alpha and will require a high percentage of the budget for updating modules two, five and six. One can conclude then, that four factors must be taken into consideration once deciding on investing in a certain module: the upper limit, complexity factor, uncertainty and architectural link between modules.

We can as well demonstrate the power of design rules by deriving a very similar table to that of Table 4.8 where identical results will be shown: larger amounts are needed to update the modules in the absence of design rules where 28.52% of the budget will go to updating the modules and the rest will go for improving the performance of the modules. On the other hand, in the presence of design rules, 5.87% (refer to Table 5.5) of the budget is needed for updating the modules and 9.23% (refer to table 5.4) of the budget for developing design rules which makes a total of 15.1% (5.87% + 9.23%) which is less than the 28.52 % in the absence of design rules.

Concerning the total product performance, at t=1 we have a total performance of 22.6, at t=2 we have an increment of 7.85 making the total performance at time two equals to 30.85, and at t=3, we have an increment of 2.34 making the cumulative total product performance equals 32.79. As discussed in the previous section, the product performance will evolve at a decreasing rate where for a large T we expect minimal increments, thus stabilization for product performance and no more improvements. We notice that in the stochastic model, the increments are too low foreshadowing an early steady state faster than that of the deterministic. Consider the below two graphs which show the growth of the product as a function of time in the stochastic and deterministic case:

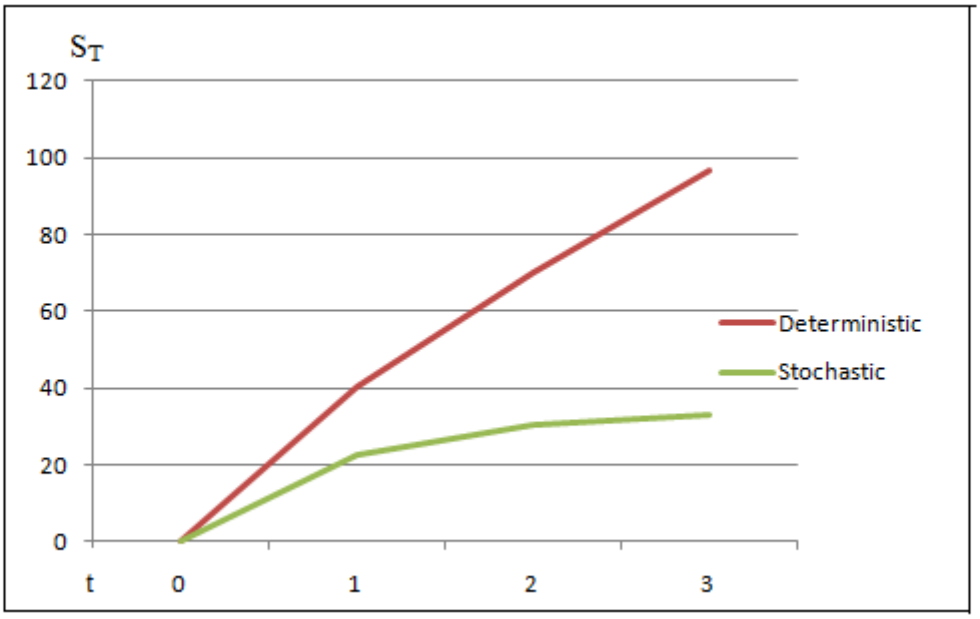


Fig. 5.1: Total product and module’s six performances in function of time

From Figure 5.1 we notice how both functions increase at a decreasing rate and how the deterministic model in our example reached higher performance amount than the stochastic one. It shows as well how the stochastic function stabilizes faster than the deterministic due to the less amounts of increments taking place.

CHAPTER 6

MANAGERIAL INSIGHTS

6.1. Overview

After defining the deterministic and stochastic model, and after establishing a methodology to measure product's performance, we come to identify some managerial guidelines which can give quick hints about investments strategies. Those insights will be based upon the results derived earlier and upon the analysis work and sensitivity studies done in the previous sections. Below are some guidelines for understanding the architecture of the product and some investments hints that can be utilized prior to optimizing the total product performance.

6.2. Guidelines

- Understand the architecture of the product.

Before investing in any module, it is important for the designer to understand the architecture of the product. The designer should clearly understand the data matrix which shows all the characteristics of the modules and explains the relationships between them. The designer should check the upper limits of all the modules and see which modules have the highest U_i. The designer should examine as well the links between the modules i.e. which columns in the data matrix are approximately fully filled. Such columns show which modules have many interdependencies relationships that demand from the rest of the modules present in the product some updates once changes are applied to them. In addition the designer should look for the most complicated or integral interdependency relationship by searching for the lowest k_{ij}. Moreover, the designer should check the fraction of updates; see by how much the modules are being affected with others' changes. By completely understanding the data matrix, the designer can have a good feel for the proper investments in design rules and modules, as discussed next

- Invest large amounts in high upper limits modules having no modules depending on them.

When a module has a very low C_i (usually accompanied by a high U_i), and having many modules depending on it then we are better not investing in such a module. Such modules, being highly complex, demand a huge amount of the budget for their re-design work. If such a large amount was assigned for performance's improvement, then a large amount would go for updates. So in such a case, we will exceed the budget and we will not obtain an optimal product performance. Consider the performance function of module *i*:

□□=Ui1+e−CiαiBg−Ui2

As you notice, when C_i is too low we need a very large α_i to decrease the denominator (1+e−CiαiBg) thus increasing P_i especially that the upper limit U_i is large when the module is complex (low C_i).

- Do not develop design rules if the relationship between the modules is extremely integral and the initial fraction of update is low.

When k_{ij} is small (integral relationship) and f_{ij} is small as well, we need a very large θ_{ij} (close to 1) to decrease f_{ij} to f_{ij}`. By this we mean, for $f_{ij}$` to be less than f_{ij} in the presence of integral module relationship and minimal updating requirements, the power of the exponential function in ⍰⍰⍰`=⍰⍰⍰⍰−⍰⍰⍰⍰⍰⍰⍰⍰⍰ should be high implying large θ_{ij}. In such a scenario, most of the budget will go to developing design rules rather than improving product performance. Accordingly, we better off not investing in design rules.

- Do not spend large investments amounts on simple modules.

Modules that are simple in designs i.e. have high C_i, do not require a huge amount of the budget for their performance improvement. The maximum alphas of such modules are small and very close to their optimal alphas as opposed to complex modules where their optimal alphas are much smaller than their maximum alphas and this due to their design complexity which demands a very large alpha for attaining maximum performance.

- If you decided on investing in a certain module, then invest as well in developing its respective design rules (if dependency exists) if the fraction of updates were noticeable (i.e. not too small).

As we saw in the example we used in the previous chapters, always the amount of updates in the absence of design rules is greater than both the amount of updates plus the amount spent on developing design rules.

As we know, f_{ij}` (□□□`=□□□□−□□□□□□□□□*)* is less than f_{ij} thus the amount of update after developing design rules is less than that without design rules:□□□□−□□□□□□□□□□□□□□□< □□□□□□□□□.

And when k_{ij} is not too small i.e. simple relationships exist between modules then the drop from f_{ij} to f_{ij}` would be huge making □□□□−□□□□□□□□□□□□□□□ much less than □□□□□□□□□ and leading to the result that amount spent on developing design rules θ_{ij} and amount spent on updating the modules is less than amount spent on updating the modules in the absence of design rules: □□□□−□□□□□□□□□□□□□□□+□□□< □□□□□□□□□

- If the complexity of a certain module was decreased, then the amount invested in that module increases or stabilizes but never decreases.

For the complexity to decrease, C_i must increase and since our objective is to maximize performance i.e. increase P_i thus we have to decrease the denominator of P_i.

□□=Ui1+e−Ciαi−Ui2

To increase P_i we have to decrease 1+e−Ciαi thus decreasinge−Ciαi. To do so we have to increase Ciαi . As C_i increases, α_i increases too or stabilizes but never decreases. This was proven in the previous chapters once sensitivity analysis on the complexity of module one was provided. As the module tends to become simpler as its respective α_i increases to a certain limit and then stabilizes. Remember that modules that are too simple design do not require large amounts of investments.

CHATER 7

CONCLUSION

7.1. Summary

The main objective listed in Chapter 1 was to develop a mathematical model which maximizes total product performance and suggest optimal investments decisions. Such decisions can target either the modules by themselves or the design rules that describe the link between the dependent modules. As we saw in the preceding chapters, a systematic methodology to optimize the performance of any architectural product was suggested. Given a certain budget, the model proposes optimal investments strategies.

This thesis have offered two kinds of models; one is deterministic where the performance of any module is guaranteed and the other is stochastic where some modules behave in a risky way where their upper limits fluctuate based on a uniform distribution function thus resulting in an uncertain return on investments.

In each model we have introduced two types of investments: one shot investment and periodic investments. Since all the decisions are assumed to take place at t=0 i.e. prior to investing in any module, the periodic model did not show any change in the results from that of one shot model. Only time component was introduced and such model can be useful if the budget was not fully given at t=0 rather parts of the budget are given each period. Accordingly the periodic model suggests optimal investments amounts per period which were decreasing per time leading to an increasing total product performance but at a decreasing rate. As it was assumed to performance of any product will increase by time and stabilizes as t tends to infinity.

Finally managerial guidelines were provided which gives quick hints about investments strategies. Those insights were based upon results, analysis work and sensitivity studies done in the previous chapters. Such guidelines are extremely important for any product development process.

7.2. Recommendations for Future Studies

Extensions of our model are possible in several directions:

- It may be productive to put more efforts on the periodic type investments whether in the deterministic or stochastic model. It would be beneficial if investment decisions can be updated throughout the development process i.e. in each period we optimize our total performance and we update our product based on investments done in previous period. No more all decisions are taken at t=0, but rather decisions should be taken in each period separately. Accordingly, the design complexity of each module should decrease by time and investment decisions are updated accordingly.
- It may be fruitful to assess uncertainty of the performance in the stochastic model with other measures (rather than uniform distribution) and would be interesting if we can take into consideration from period to period the *real* performance attained since at the end of each period the designer would know for certain the performance reached and such knowledge would affect the decisions of the next period.

REFERENCES

Allada, V., Lan, J. (2002). New Modules Launch Planning For Evolving Modular Product Families. *Journal of Intelligent Manufacturing* , 19 (2), 131-148.

Baldwin, C., Clark, K. (1999). *Design Rules: The Power of Modularity*. Cambridge, MA, USA: MIT Press

Button, R. M., Soeder, J. F. (2004). Future Concepts for Modular, Intelligent Aerospace Power Systems. *Soeder,NASA Glenn Research Center, Cleveland, OH*, 44135.

Eppinger, S., Ulrich. K. (2008). *Product Design and Development*. New York, NY: McGraw-Hill/Irwin.

Foster, R. (1986). The S-curve: A New Forecasting Tool. Chapter 4 in *Innovation, The Attacker's Advantage*. New York, NY: Summit Books, Simon and Schuster.

Huberman, B. A., Loch C. H. (1996). Collaboration, Motivation, and the size of Organizations. *Journal of Organizational Computing and Electronic Commerce,* 6, 109 – 130.

Huberman, B. A., Hogg T. (1995). Communities of Practice: Performance and Evolution. *Journal of Computational and Mathematical Organization Theory,* 1, 73-92.

Kashtan, N., Parter, M., Dekel, E., Mayo, A. E., Alon, U. (2009). Extinctions in Heterogeneous Environments and the Evolution of Modularity. *The Society for the Study of Evolution,* 63 (8), 1964-1975.

Krishnan, V., Bhattacharya, S. (2002). Technology Selection and Commitment in New Product Development: The Role of Uncertainty and Design Flexibility. *Journal of Management Science,* 48 (3), 313-327.

Mihm, J., Lock, C., Huchzermeier, A. (2003). Problem-Solving Oscillations in Complex Engineering Projects. *Journal of Management Science,* 49 (6), 733-750.

Samuels, R. (1998).Evolutionary Psychology and the Massive Modularity Hypothesis. *The British Journal for the Philosophy of Science*, 49(4), 575-602.

Sullivan, K. J., Griswold, W. G., Cai, Y., Hallen, B. (2001). The Structure and Value of Modularity in Software Design. *ACM SIG SOFT Software Engineering Notes*, 26 (5).

Thomke, S., Nimgade, A. (1999). Product Development at Dell Computer Corporation – Teaching Note. *Harvard Business School*, HBS No. 9-699-010.

Ulrich, K., (1995). The Role of Product Architecture in the Manufacturing Firm. *Journal of Research Policy,* 24, 419-440