The knowledge evolution framework: a knowledge management perspective on the impact of knowledge segregation on product development projects

by

Jeffrey C. Chi

Master of Arts, Cambridge University, United Kingdom (1994)
Master of Science, Massachusetts Institute of Technology, Massachusetts (1992)
Bachelor of Arts, Cambridge University, United Kingdom (1990)

Submitted to the Department of Civil and Environmental Engineering
In Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy
in
Systems Engineering and Organizational Knowledge

at the

Massachusetts Institute of Technology

June 2001

[illegible]

Signature of Author..
Jeffrey C. Chi
Department of Civil and Environmental Engineering
March 7, 2001

Certified by..
John R. Williams
Associate Professor
Civil and Environmental Engineering
Thesis Supervisor

Accepted by..
Oral Buyukozturk
Chairman, Departmental Committee on Graduate Students

The knowledge evolution framework: a knowledge management perspective on the impact of knowledge segregation on product development projects

by
Jeffrey C. Chi

Submitted to the Department of Civil and Environmental Engineering
on March 7, 2001 in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in the field of
Systems Engineering and Organizational Knowledge

ABSTRACT

Successful product development projects are critical to success in many industries. Developing offerings faster, better and cheaper than competitors has become critical to success. In response to these commercial pressures, many industries have shifted from a sequential, functional development paradigm to a concurrent, team-based paradigm. Increasing the concurrence and cross-functional development, however, also dramatically increases the dynamic complexity of development projects. This is added complexity on top of the increasing technological complexity of offerings.

Whilst traditional models based on tools such as the CPM and PERT have been used for the planning and management of such projects successfully in the past, the increase in complexity has shown that such tools are less capable of planning and predicting the outcome of projects. This is due to the dynamic nature of projects and the task-based approach used. Recent research using dynamic simulation tools such as system dynamics have indicated reasons that project duration and cost have been consistently underestimated. The differentiation is attributed to the ability of dynamic simulation tools to capture the iterative nature of work. Existing research has, however, concentrated on iteration as a result of errors, quality control and shifting targets. Although these factors do contribute to iteration, they seem almost exogenous factors where independent policies can be used to mitigate the need for iteration. Yet all texts on design and product development describe the process as an iterative one. There must, therefore, be some endogenous factors that result in an inherent need for iteration. With the level of complexity of projects, specialization becomes necessary. As a result, no individual within a project has "full" knowledge about the project and its current state of development. In other words, the information and knowledge is segregated to different parts of the project organization. This research proposes a shift towards a knowledge-evolution paradigm and uses it to investigate the impacts of such knowledge segregation on the performance of product development projects. This proposed framework acts, in part, to provide management levers and measurements in managing the knowledge within product development projects. These are measurements that the traditional task-based frameworks cannot directly provide.

A dynamic simulation model of a development project with multiple persons was built using the system dynamics methodology. The model portrays the segregation of knowledge and studies its impact on the rate of development and iteration. The model was applied to projects with a scope defined by the Delta Design Game. Though simple, the Delta Design game provides the boundaries in understanding the interactions amongst participants in a typical process and has been used in the past primarily to demonstrate this aspect of design. The model simulation bore results that closely resembled real life behavior of the Delta Design project.

The model was also applied to the investigation of differing policies for improved project performance. These policies include strategies involving conservatism in design, preemptive measures against iteration and reduction of knowledge segregation through the decoupling of the knowledge processes. The model structure provides insight as to the effectiveness of these strategies.

The research finds that rework and iteration happens inherently in development projects and its prevalence is interwoven into the fabric of the system architecture and project organization structure.

Finally, this research has shown value in the knowledge evolution paradigm by gathering insights through which task-based models could not. In so doing so, it is shown that there is value in developing this concept further to enable a better understanding of understanding product development projects and thus enabling better and more efficient means of managing them.

Thesis Supervisor:	John R. Williams
Title:	Associate Professor of Civil and Environmental Engineering

Acknowledgements

This work could not have been possible without the support and contributions many have provided. This section is dedicated to these people as a token of my appreciation and gratitude.

Gratitude goes out firstly to members of my doctoral committee, who have collectively and individually constantly provided direction and advice through their systematic challenges and sometimes piercing questions. In particular, to my research advisor, Prof. John Williams who first encouraged me to look into the field of knowledge management and challenged me to think how it could be used to more accurately predict the performance of projects. He has always been available for numerous discussions we had on the merits of dismerits of current systems. Prof. Jerome Connor, for his mentorship throughout my academic career here at MIT. His sound advice at critical stages of my academic career here has made my experience here at MIT both fulfilling and focused. Prof. Feniosky Peña-Mora, for his valuable insights into project management problems, always challenging my research and I in the right direction.

Others at MIT who have contributed significantly to this research include Prof. Sarah Slaughter for her advice on research methodologies and being available as a sounding board for this research, Prof. John Sterman for his amazing introduction to the world of system dynamics, Prof. Louis L. Bucciarelli for allowing the use of the Delta Design Game for the purpose of this research as well as for the discussions we had on the design process and Prof. Herbert Einstein for allowing the use of copies of the Delta Design Game. Thanks must also go out to current and previous fellow students for their assistance and camaraderie. In particular, Hank Taylor who helped provide direction in the earlier days of this research and Moonseo Park for the few discussions we had on some of the finer implementation issues in System Dynamics. My colleagues at Spandeck Engineering, Singapore who over the years have provided much of my background experience and motivation in pursuing this area of research have also indirectly contributed to this research.

I also owe thanks to those around me that have provided the emotional and moral support that has kept me going the last few years. In particular, my wife Lena has really been a pillar of support for me in this area being able to encourage me all this will while living with the financial sacrifices we had to make in pursuit of this dream. My parents and sister have also been extremely supportive in this regard.

Finally, but definitely not the least, gratitude goes out to the many individuals who have provided me with a life outside of academic research. These individuals are the persons that I have come into contact during my brief but meaningful tenure here at MIT including colleagues from the Intelligent Engineering Systems Laboratory, staff and faculty of the Leaders for Manufacturing and Systems Design and Manufacturing Programs, staff and faculty of the Department of Civil and Environmental Engineering (especially Cynthia, Jessie and Joan), fellow founding team members of e-MIT and certain members of the Sloan MBA classes of 2000, 2001 and 2002.

TABLE OF CONTENTS

List of Tables

List of Figures

Chapter 1 Introduction

1.1 Knowledge evolution in product development

In recent years, there has been much euphoria on the economy moving from an industrial economy to an information economy and the eventual transition to a knowledge-based economy. This is an economy where value is added not through physical work on a product (industrial economy), not through providing information (information economy) but through the provision and application of knowledge or know-how.

As the economy progresses from industrial to information to knowledge-based, so must there be a transition through which we manage and understand our product development projects. Whether these product development projects are for office buildings, software packages or automobiles, the efficient management of these projects is becoming increasingly important. This can be observed through the history of product development where industry have moved from solving physical constraints through the analysis of bottlenecks and supply chains in the industrial economy to building ERP systems in providing timely information to various divisions of an organization. As we move into the knowledge economy, an understanding of the knowledge and knowledge transfer issues in product development needs to be further understood.

Current project management methodologies are task-based and concerned with the organization and sequencing of tasks. Though they have been and will continue to be useful, they do not provide us with the direct measurements and levers to understand and manage the know-how within project organizations. This research proposes the knowledge evolution framework as an alternative paradigm for modeling projects.

Recent initiatives to improve the product development process include concurrent engineering, cross-functional development teams and product platforms. These initiatives have all been implemented based on commercial pressures to develop products faster, better and cheaper than competitors. On the surface, it seems fairly obvious what these initiatives propose to achieve. They are all intended to hasten the product development process through more efficient use of knowledge and resources. See section 2.4.2 for a more complete discussion. These initiatives are therefore some form of knowledge management implementations. Concurrent engineering allows work that has no knowledge or information pre-requisites to proceed without hindrances from other functions. As will be later demonstrated, in order for concurrent engineering to be successful, the relationships of these pre-requisites need to be managed well. Cross-functional teams have been implemented to allow shorter iteration cycles by ensuring that all the knowledge required to proceed is close at hand. Product platforms ensure knowledge of the product is re-used over a family of products instead of re-inventing the wheel each time. All of these initiatives can therefore be considered implementations of more efficient creation, use and transfer of knowledge over traditional product development processes.

As product development projects embark into a new era, so must the processes and tools that we use to manage these projects. In an environment where most projects overrun both schedule and costs, and where the rule of thumb to estimate project duration is to take an accurate guess and double or triple it, traditional task-based project management tools such as CPM and PERT have not proven to be accurate predictors of project performance. The situation is further aggravated by the increase in complexity in product development projects. Many researchers consider that part of the reason for this is the failure to capture iteration and the dynamic nature of projects. Some researchers have captured iteration through dynamic modeling of projects but have defined iteration as rework due to errors, mistakes or change in scope of project. This is because the paradigm used is still task based and is only able to capture iteration as work that is complete but not acceptable.

Clearly, however, iteration is in the very nature of product development projects and cannot be avoided through better quality control or other measures taken to minimize the iteration identified above. In order to make such an observation, a shift in paradigm to one that captures information or knowledge needs to be used. This research proposes the knowledge evolution paradigm where the state of the knowledge is tracked as the project is in progress. In such a framework, iteration is represented as a reduction in the state of knowledge. This does not mean that knowledge is lost but that information and knowledge that was generated is no longer relevant. That information and knowledge, however, was a necessary step in the generation of further information.

The knowledge evolution paradigm is based on the fact that knowledge and information is generated and increases through the various stages of development of any development project. This knowledge and information is captured both tacitly in individuals associated with the project as well as explicitly in documents, drawings, prototypes and partially built systems. Thus, knowledge about the product evolves through the development of the product. Understanding the issues that affect its evolution perhaps provides a deeper understanding on the product development process and provides us with insights as to how we should manage product development projects in a knowledge-based economy.

1.2 Increasing complexities in product development

Recently, product development projects have become much more complex due to two primary reasons. There is, firstly, an increase in complexity as a result of more sub-systems and components working together. Secondly, the project itself has become more complex through larger organizations and more processes being executed concurrently.

Technological activities and advances have generated much information and knowledge. Xerox, for example, estimated that the volume of knowledge available to the public would double every 73 days by the year 2000 (Brethenoux, 1997). Studies showing exponential growth rate of journal articles and manpower give concern over the enormous amount of scientific information being generated (Price, 1965). This technological explosion has resulted in engineers and architects having to specialize into specialties and even sub-specialties just to keep abreast of recent scientific and technological advances in various fields. Product development project organizations have also become more complex as different participants with different specializations and perform only specific functions.

This results in individuals that specialize in their domain of expertise and operate only with the information they need to know. Their knowledge of the project is hence segregated from other specialists in other domains. So, not only have the products become more complex, the process in developing these products has become more complex as well.

The process has also been made more complex in response to commercial pressures to develop products faster, better and cheaper than competitors. Many industries have shifted from a sequential, functional development paradigm to a concurrent, team-based paradigm. Although supposedly increasing project performance (lowering cost, shorter project duration, higher quality), there are increased dynamic complexities (Smith and Eppinger, 1995; Wetherbe, 1995; Osborne, 1993).

1.3 Projects, Teams and Knowledge

Complex development projects are seldom handled by small teams any more. The amount of detailed knowledge needed requires deep functional expertise in many areas. A project team often consists of many team members working for different organizations with different functional expertise and different business interests. One such example is in the development of an office building. The different team members include architects, civil and structural engineering, HVAC engineers, electrical engineers, general contractors and specialist contractors. Team members have different functional expertise, different responsibilities and possibly different business interests in the development project. A successful product development project, however, requires that all the different team members to contribute in their specific domain of expertise.

In the individual contributions, members make use of information that is available (either from their past experience or from information provided by other members) to generate more information about the development project and the final product. This information is in turn used by other members on the team to make their contribution. In this fashion, information about the offering is created and built up to form the product knowledge. Collectively, as a whole team, knowledge about the offering increases over time as the offering moves from concept to conceptual design to detail design to prototype to physical product. At the various stages of development, various mediums are used to contain information/knowledge about the product.

1.4 Motivation for Research

In an environment where most estimates for project duration and costs rely on large "fudge" factors, traditional project management tools such as CPM and PERT appear to be missing some critical ingredients. Earlier research has exposed that these tools are limited by their use of an indirect project measure (time) and by bundling the characteristics of and relationships among scope, resources and processes in each activity into a single duration estimate. They also tend to ignore iteration or require that iteration be implicitly incorporated into duration estimates and precedence relationships (Ford and Sterman, 1998).

More recent research approaches do overcome some of these shortcomings by identifying some of the dynamic consequences of different project structures on project performance. Crucial dimensions that have been identified include processes, resources, scopes and targets. Like traditional project management tools, these studies are also task based and models iteration as rework to tasks that have already been performed and need to be redone due to errors, mistakes or failure to meet quality standards. What is implied, however, is that as long as errors and mistakes are minimized and as long as quality is properly controlled, iteration and rework can be minimized.

The mere process of product development, however, is an iterative one. Any text on design will introduce design as an iterative process. They will also state that design is more art than science. Any text on project management will emphasize the importance of change management, not on how to prevent changes but on how to manage them. It seems, that on certain aspects of the project, iteration is unavoidable. It therefore becomes interesting to ask why iteration actually occurs and to try understand its nature. One way of finding the solution is in asking why designers can't get it right the first time round. The solution seems to lie in the fact that not all participants in the design process has knowledge of or is able to make entire decisions on the product being developed. In fact, sometimes design decisions are based on the outcomes of others' design.

Whilst better understanding the true nature of under performance of projects is useful for more accurate control and ultimately lead to better performance, the commercial pressures to develop products cheaper, faster and at lower costs have resulted in the adoption of new methods such as concurrent engineering, product platforms and cross-functional teams. On the surface, the logic behind these implementations is quite apparent. In fact, in some cases there have been significant improvements to the adoption of these new methods (Merrils, 1991; Nevins and Whitney, 1989). However, aggregated results have been fairly mixed (Iansiti, 1993; Clark and Wheelwright, 1993; Dean and Susman, 1991). It seems that there may therefore be some missing "ingredients" that may act to contradict the logical intention of increasing project performance. Researchers have attributed this contradiction to increased complexities and tightened constraints imposed by the interdependencies requiring higher levels of coordination (Ulrich and Eppinger, 1994; Malone and Crowston, 1990). It has also been observed (Goldratt, 1997) that when responsibility to the project is distributed, participants in the development process tend to build in buffers. Although it has been recommended that these buffers be managed centrally, it is not quite clear its true impact on project performance.

The motivation for this research is to uncover some of the dynamic complexities of interdependencies, project complexities and this inherent nature of iteration. It is also intended to provide some insight as to the effectiveness of implementing initiatives such as concurrent engineering and cross-functional teams. As traditional project management methods using task-based models seem unable to capture these interdependencies and complexities, it is hoped that information based on the knowledge evolution paradigm will be able to provide some insight.

1.5 The Problem

Current research relating to iteration in product development projects either models iteration as separate tasks or as rework to mistakes, errors or lack of quality control. The problem with these models of iteration is that they do not capture the true nature of iteration, which is embedded in having multiple individuals involved in product development projects. The question at hand is then whether interdependencies or project complexities do impact iteration in projects.

In answering the above question, it is first necessary to determine what constitutes interdependencies and project complexities and their relationship to knowledge segregation. The concept of interdependencies and the need to minimize them has been identified by quite a few researchers (Suh, 1990; Steward, 1981). Though quite similar in nature, there does seem to be differing definitions. Suh's (1990) definition (under axiomatic design) is related to a particular parameter affecting more than one property whilst Steward's (1981) definition (under the design structure matrix) is related to processes requiring information from other processes.

Complexities in product development projects mostly refer to the number of systems, sub-systems and components. In the context of this research, however, it seems that somehow the issue of interdependencies needs to be factored in as well.

In order to tie these relationships to iteration, it seems necessary to move away from task-based models used in previous research. Task based models imply pre-determined scopes. It is relatively clear that scope cannot be determinate especially in the early stages. A new knowledge evolution paradigm could provide the insights needed. A secondary problem that needs addressing is how these findings should affect the policies that occur in daily project management especially, policies that are related to segregation of responsibilities and knowledge.

1.6 Research Approach

The purpose of this research is to understand the relationship between project organization structure, how knowledge is dispersed and its impact on project performance. While it cannot be possible to provide a complete understanding of these relationships, this thesis' contribution is in the identification of knowledge segregation and its impact on iteration and hence project performance. This is achieved by firstly establishing a new paradigm of knowledge evolution view of projects.

This thesis uses the system dynamics methodology (Forrester, 1961) for modeling complex systems. System dynamics describes cause and effect relationships with stocks, flows and feedback loops. Stocks and flows are used to model the state of knowledge of the project as well as the processes that act to increase or decrease those states. Feedback loops are used to model organizational structures as well as project management decisions and policies. As it is unknown whether a particular design cycle will be the final one, iteration is probabilistic. This is also modeled explicitly.

Apart from the modeling, results are also observed from projects. The Delta Design game is an abstraction of an actual design project and is primarily used to demonstrate social interactions in a design process (Bucciarelli, 1991). The Delta Design game was used primarily because of its simplification of the design parameters. The fact that participants only work together for 4 hours, that previous knowledge in any domain is not of any benefit in the delta design process isolates the main crucial components of the actual design process for observation. As there are 4 distinct roles in the delta design process, each with specific domains of knowledge and expertise, knowledge is segregated and making the project ideal for the purposes of this research.

The system dynamics model and the results from the Delta Design process are compared and contrasted providing insights to these dynamics of the design process. The model is then used to investigate how project performance varies with variations in key variables. These results can then be used to evaluate the merits of certain policies intending to improve project performance.

1.7 Outline of thesis

This section is intended to provide readers with an understanding of how the content is laid out in the remainder of this thesis. Chapter 2 provides a review of the current research and literature on the variety of topics that are related. As the knowledge evolution framework proposed lies in the intersection of various fields of research including knowledge management, organizational management, product development, design and project management. Chapter 2 covers to some detail the current state of related research in each of these fields.

On the knowledge management front, literature explaining the concept of embodiment of knowledge in physical artifacts, the differentiation between tacit and explicit knowledge and research on measuring the state of knowledge are introduced. Basic concepts in the field of organizational theory along with why there is a need to organize are also explained. Under product development, research in the space of concurrent engineering and cross-functional teams are explored in a little more detail. The various project management tools including CPM, PERT, DSM, Axiomatic design and previous project management models in system dynamics are reviewed as a pre-cursor to this research.

With the current state of research reviewed, Chapter 3 puts some terms into context. Definitions of terms such as interdependencies, project complexities and knowledge segregation are put into the context of this research and the knowledge evolution framework. Chapter 3 also introduces the concept of knowledge evolution and builds a basic system dynamics model representing the knowledge evolution of development projects. The model consists of several components. Firstly, the knowledge repository represents the state of knowledge at any particular point in time. The state of knowledge either increases or decreases through the processes that actually contribute towards them. Knowledge processes are the individuals or groups of individuals that contribute towards increasing the state of knowledge in the repositories. However, as certain individuals do not have full access of knowledge and information they need to complete their design, there is a probability that the initial design is not workable and in need of changes. This constitutes the feedback loop that determines the iteration.

Chapter 4 introduces the Delta Design game and how it is used to validate the knowledge evolution framework. The knowledge evolution model is applied to the parameters of the Delta Design and its results are compared with the actual results. The Delta Design process was chosen specifically for its simplification of parameters vis-à-vis a more complicated project. Although simple, the Delta Design process is one that demonstrates knowledge segregation amongst individuals and is simple enough to execute multiple projects. Six projects with the same scope were concurrently executed and comparisons were made both between the projects and with the simulated knowledge evolution model.

Chapter 5 then analyzes how specific variables affect the performance of product development projects. This analysis yields some insights of which some are intuitive and others are not. These insights have implications as to whether specific strategies and policies intended to improve project performance will actually work. In particular, the issue of whether "holding back" and building in buffers or conservatism at the individual level benefits the project at a global level is explored. Suggestions as to why implementations of concurrent engineering and cross-functional teams do not universally yield positive results are also provided in this context.

Finally, in Chapter 6, the results of this research is summarized and concluded. Further research topics are proposed in the interest of developing this new framework.

1.8 Summary

In response to commercial pressures and increased competition, product development projects have had to face increased complexities. It therefore becomes ever more crucial to be able to control and manage project performance with regards to lowering cost, reducing project time and have higher qualities.

Although recent research has identified iteration as one of the key reasons for the lack of predictability of project performance, the iteration is implied to be due to mistakes, errors or failure to meet quality requirements. This seems awkward since most design texts will describe product development and design as iterative processes; not so much due to errors or quality but just inherently iterative. As our product development projects become more complex, it becomes more crucial to understand the sources of these iteration.

In order to so, our project management methods need to move away from the traditional task-based methods to information or knowledge evolution based methodologies. The hypothesis of this research is that if information and knowledge is segregated, iteration cannot be avoided. It is, therefore important to understand the impacts of separating information and knowledge on the dynamics of project performance. With a suitable model for understanding this aspect of project dynamics, we can apply it to the investigation of policies that will enable project managers to make more informed decisions on project structure and execution.

Chapter 2 Literature Review

2.1 Introduction

This chapter describes and evaluates the literature as it pertains to this research. As this research draws upon topics in several existing fields of research. A brief overview of relevant topics in each of these fields is reviewed. Firstly, research in the field of knowledge management is reviewed for their applicability in product development projects. Specifically, the differentiation between tacit and explicit knowledge and the various forms of knowledge representation used in the product development process. Then, some of the recent initiatives in product development processes are reviewed and compared with the traditional product development process. These initiatives include the use of concurrent engineering, cross-functional teams and product platforms. In view of the knowledge management definitions, these recent initiatives of the product development process may also be viewed as knowledge management initiatives. There is also a review of the current project management tools used to plan and manage product development processes including CPM, PERT, DSM and Axiomatic Design. Recent research involving the use of system dynamics are studied and some of the more important structures such as the work availability, rework, labor, quality, scope and schedule are briefly explained. Finally, gaps in the current literature are identified as the starting point for the specific work of this research.

2.2 Literature Review on Knowledge management issues

Knowledge and knowledge management has become infused with almost any meaning somebody wants to associate with it (Gates, 1999). Collaboration, data mining, web casting, intranets, knowledge based systems, artificial intelligence are but a few of the modern day fields that claim to have the solutions to knowledge and knowledge management. With the advancement of information systems and in fields such as those mentioned, it is no wonder that the distinction between knowledge and information is somewhat blurred. As there is yet to be much consensus on a proper definition, it becomes difficult to really define what is meant by knowledge and knowledge management. The purpose of this review is not really to provide that definition, nor is it to debate what the various definitions are. It is, however, meant to provide some understanding of the current thinking in knowledge management and to set out some distinctions between the terms information and knowledge.

Knowledge and information are distinct identities. Although it has been commonly misconceived that knowledge resides in a collection of information or even that it is a collection of information, knowledge actually resides within the user of information which are people (Churchman, 1971). In fact, knowledge is embedded in people, and knowledge creation occurs in the process of social interaction (Sveiby, 1997). It is also further emphasized that only human beings can take the central role in knowledge creation and that computers are merely tools, however great their information processing capabilities (Nonaka, 1995).

2.2.1 Tacit vs. Explicit Knowledge

When dealing with knowledge management issues it also becomes necessary to differentiate between explicit and tacit knowledge (Nonaka, 1991). Explicit knowledge or codified knowledge is knowledge that can be codified or easily recorded and retrieved. Such codification could be in the form of writing or even embodied in a physical product. The key is that another person can easily grasp the knowledge by reading or studying the codification. Tacit knowledge, on the other hand, is knowledge that is uneasily recorded. For example, a baker relies very much on his experience when kneading dough to get the right consistency, texture etc. Variables like consistency and texture are not exactly easily codifiable. However, alternative means of codifying the knowledge of baking bread can be easier through means such as recipes, mixing times etc. The former type of knowledge is tacit but can be eventually codified to some form of explicit knowledge (this is the articulation process as described later).

Tacit knowledge consists partly of technical skills – the kind of informal, hard-to-pin down skills captured in the term "know-how." At the same time, tacit knowledge has an important cognitive dimension. It consists of mental models, beliefs and perspectives so ingrained that we take them for granted, and therefore cannot easily articulate them.

When an employee leaves a firm, some knowledge of the employer's operations, experience and current technology leaves as well. There are some aspects of the employee's knowledge that are not codified in some manual or explicitly documented anywhere. Likewise, researchers are also realizing that people are the best carriers of information and knowledge and that the best way to transfer knowledge between organizations is to physically transfer a human carrier. (Shapero, 1969; Roberts and Wainer, 1971).

Further, Michael Polanyi proposes (Sveiby, 1997; Polanyi, 1958; Polanyi, 1966) that "we can know more than we can tell" and that explicit knowledge, expressed in words or numbers is therefore only the tip of the iceberg. Furthermore, "while tacit knowledge can be possessed by itself, explicit knowledge must rely on being tacitly understood and applied. Hence all knowledge is either *tacit* or *rooted in tacit knowledge*.

The distinction between tacit and explicit knowledge suggests four basic patterns for creating knowledge in any organization. These are socialization (tacit to tacit), articulation (tacit to explicit), internalization (explicit to tacit) and combination (explicit to explicit) (Nonaka, 1991) as illustrated in Table 2-1. All four of these patterns exist in dynamic interaction, a kind of spiral of knowledge. New knowledge always begins with the individual. That knowledge is rooted in tacit and needs to be described to other members in the organization (articulated). The team then combines different pieces of knowledge keeping it explicit in the form of documents or physical products (combined). The team members then individually understand the knowledge (internalize) these are then grasped in further detail with the individual's existing knowledge creating new knowledge (socialized). The process is then repeated in a spiral.

		To	
		Tacit	Explicit
From	Tacit	Socialization	Articulation
	Explici	Internalization	Combination

Table 2-1 Nonaka's 2X2 model

2.2.2 Artifacts as codified knowledge

If we accept Polanyi's position that knowledge is inherently tacit. One of the challenges for managers must be to put organizational knowledge into a form that makes it accessible to those who need it. This is indeed the aim of codification which literally turns knowledge into a code (not necessarily a computer code) to make it as organized, explicit, portable and easy to understand as possible.

One analogy is given by Allen (1977) in describing the similarities (and differences) between science and technology. Both science and technology are ardent consumers of information (and knowledge). While scientists take information (and knowledge) in verbal form (including diagrams, equations, documents) and transform them to some other verbal form, technologists transforms them not only into verbal form but also into a physically encoded form such as a product (Allen, 1977). This is best illustrated by Figure 2-1.

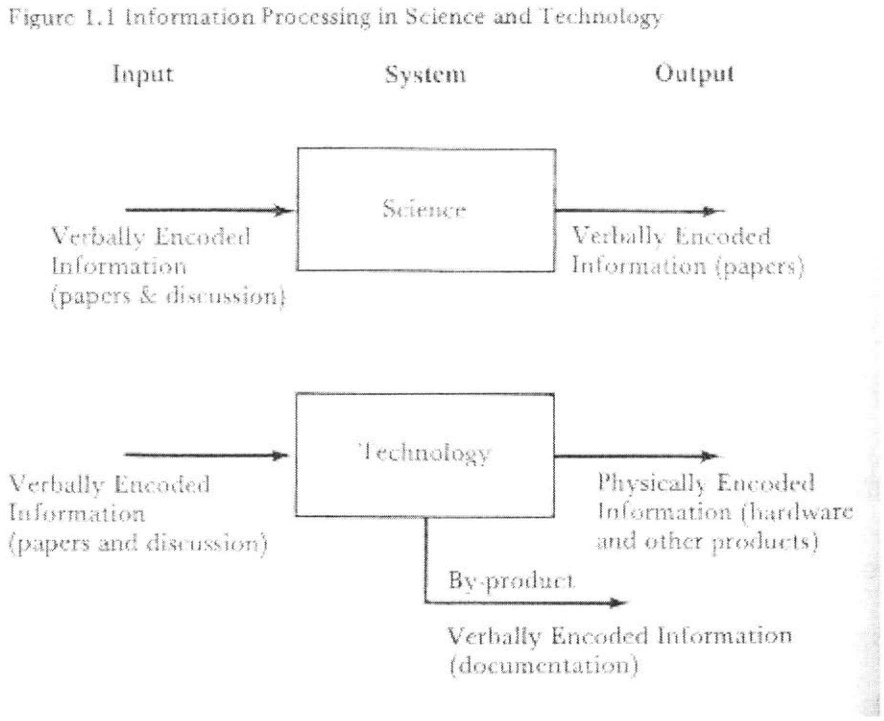


Figure 1.1 Information Processing in Science and Technology
Input
System
Output
Science
Verbally Encoded Information (papers & discussion)
Verbally Encoded Information (papers)
Technology
Verbally Encoded Information (papers and discussion)
Physically Encoded Information (hardware and other products)
By-product
Verbally Encoded Information (documentation)

Figure 2-1 Information Processing in Science and Technology

(Source: Allen, 1977)

One of the primary difficulties in codification work is how to codify knowledge without losing its distinctive properties turning it into less rich information. This is especially the case for tacit knowledge which is by nature difficult to articulate. By definition, tacit knowledge cannot be effectively codified, at least in print. A document cannot capture a child's experience, know-how, skills, and sense of balance on riding a bicycle.

There are two means in which codifying tacit knowledge can be made more effective (Davenport and Prusak, 1998). The first is through knowledge maps or basically pointers to which persons or groups through which knowledge can be accessed. The second is through richer media other than text. Media such as drawings, pictures and video provide richer content and more information. The age-old saying that “a picture is worth a thousand words” does hold true in this case.

In product development projects, the use of drawings, scaled models, prototypes all act to provide information that is otherwise difficult to describe in words to other participants in the development process. In that regards, these artifacts are used as a means of conveying and transferring knowledge in the product development process. One can therefore imagine that in the product development process, knowledge about the product evolves from concept (perhaps only function with very little shape and form) to total knowledge about the product (including shape, form, size and manufacturing process). As the knowledge evolves, this knowledge needs to be captured and codified so other participants can share the information. The means by which this knowledge is codified would depend on who is using the information generated as well as how explicit or tacit the knowledge is.

2.2.3 Measuring Knowledge

The knowledge evolution framework proposed in this research tracks project knowledge over the duration of the project. It may, therefore, be necessary to somehow measure the knowledge generated as it evolves. Whilst some may suggest, especially in this digital age, that a unit similar to the bit in information technology may be relevant, knowledge (and information for that matter) is never that precise. In the product development process, a more “flexible” unit such as the chunk defined by Simon (1969) may be suitable. However, in order for the measurement to represent something meaningful and track-able, such measurement must somehow account for the need to quantify the end of a project. Since it is not possible to predict how many chunks or bits need to be generated, it is perhaps not a suitable measurement to be used. Bohn (1994) proposed scale for measuring technical knowledge. He had identified eight stages of technological knowledge ranging from complete ignorance to complete understanding. Each stage describes the knowledge about a particular input variable on the process output. The stages are summarized in Table 2-2.

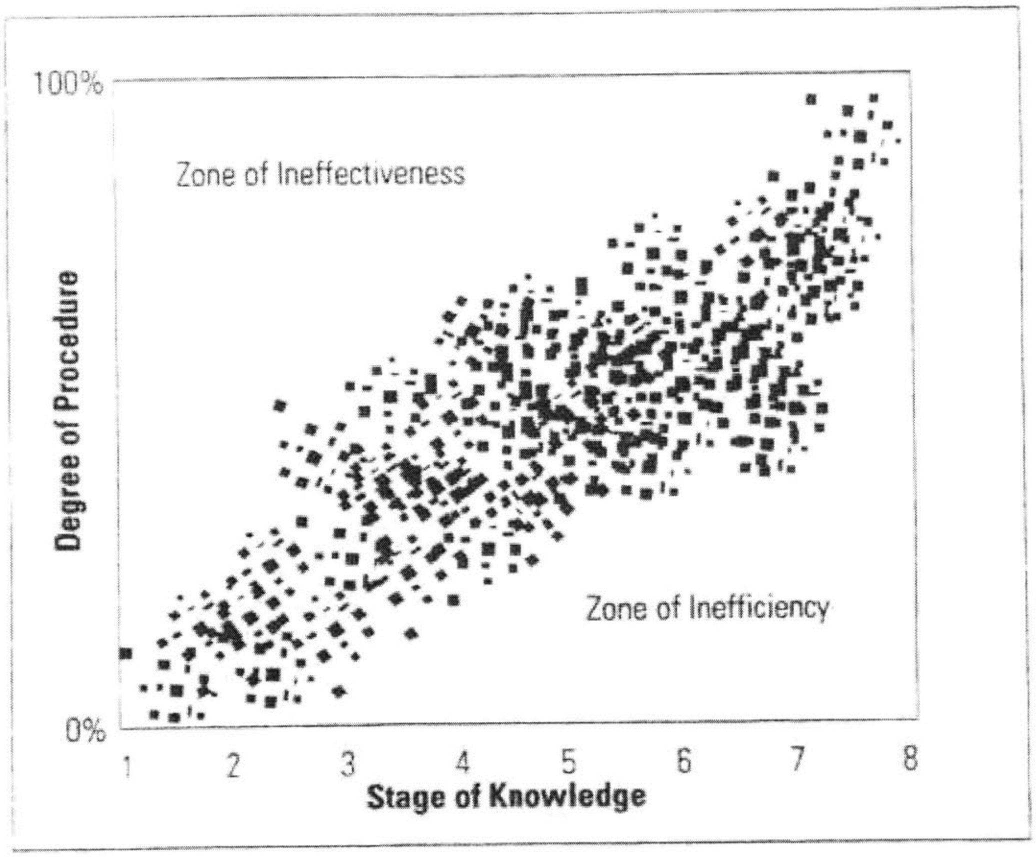
Stage	Name	Comment	Typical Form of Knowledge
1	Complete ignorance		Nowhere
2	Awareness	Pure Art	Tacit
3	Measure	Pre-technological	Written
4	Control of the mean	Scientific method feasible	Written and embodied in hardware
5	Process capability	Local recipe	Hardware and operating manual
6	Process characterization (know how)	Tradeoffs to reduce costs	Empirical equations(numerical)
7	Know why	Science	Scientific formulas and algorithms
8	Complete knowledge	Nirvana	

Table 2-2 Bohn's Stages of Knowledge

Variables in the first three stages are usually considered exogenous. This means that it is impossible to control them. Those in subsequent stages represent increasing understanding of causality and control over the outputs. As the nature of knowledge changes qualitatively with each stage of this framework, the process of learning from one stage to the next also changes.

There could be many variables governing a particular process. Ideally, a company would like to have a high stage of knowledge about all the important variables and a low stage of knowledge about variables that have negligible effect. As more is learned about part of the process, old variables are brought to higher stages but new variables also emerge from the mist of ignorance. The process as a whole can do no better than the knowledge about its most important drivers.

The knowledge stage of different process variables is important because it determines how to manage both the knowledge and the production process. The higher the stage of knowledge, the closer the process is to "science," and the more formally it can be managed. One of the basic system-design decisions is the degree of procedure. The spectrum of different means of performing a certain task consists of pure procedure at one end and pure expertise on the other end. Procedures require a specified set of rules about what to do under different circumstances whereas expertise requires experienced and skilled people who use their own judgment at each moment. These people have tacit knowledge and may not be able to explain how they carried out a task even though they could perform it. Figure 2-2 shows that there is a natural relationship between degree of procedure and stage of knowledge. This results from the fact that certain knowledge becomes necessary to be able to use procedures to manage the process.



Source: R.E. Bohn and R. Jaikumar, "The Development of Intelligent Systems for Industrial Use: An Empirical Investigation," in *Research on Technological Innovation, Management and Policy*, ed. R. Rosenbloom (London and Greenwich, Connecticut: JAI Press, 1986), pp. 213-262

Figure 2-2 Ideal operating method and stage of knowledge

(Source: Bohn, 1994)

It seems that a similar notion of stages of knowledge may be used to track product development processes. Stages do provide some measurements yet provide the flexibility of not being a hard quantitative value. In fact, an alternative view of Bohn's stages of knowledge can be viewed in a continuum of certainty of control. The more knowledge of the technical process that is available, the more certain one can be on the control of the process. Meaning there is low uncertainty in the outcome of the results. This concept of uncertainties is indeed very powerful and will be a concept that has implication in this research.

2.3 Literature Review on Organizational Issues

Whilst there is no doubt that organizational performance is one of the key factors in determining project performance. It is not the only factor. Nonetheless, having an efficient project organization, making use of limited organizational resources and dealing with issues such as communication is very important in any project.

2.3.1 Basic Preconditions for organizing

The term organization refers to the complex pattern of communication and relationships in a group of human beings (Simon, 1976). The purpose for building these organizations and for organizing overcoming limitations that particular groups of human beings may have. Groth (1999) identifies six areas where we can run into such limitations. These limitations include:

- *Capacity for work.* Both our need for organizations and their nature are strongly dependent on the nature and amount of work that we can carry out.
- *Memory performance.* The basis for any intellectual activity and crucial for accumulation of knowledge and for the management of complex relationships. Both the storage capacity and the retrieval capabilities of our long-term memory are of vital importance. So are the limitations of our short-term working memory.

- *Information processing capability.* Closely related to the question of work capacity, our ability for reasoning, problem solving, and decision making is directly related to the amount of complexity we can handle.
- *Communication bandwidth.* The first of communication's two aspects. The amount of information we can absorb and disseminate per unit of time is of obvious importance.
- *Communication range.* This is the second aspect of communication. How far and fast we can communicate is also central, as are the possibilities of communicating not only over distance but through time.
- *Emotions*
-

2.3.2 Features of organizations

The basic features of an organization are the division of labor and the need for coordination. The coordination efforts require both information processing and communications. The division of a greater, common task into smaller ones that are suitable for single persons is the defining feature of purposeful organizations. In principle, there are two ways of dividing work: everyone does the same thing in parallel or the total task can be divided into specialized subtasks. Once the overall task is divided into more or less specialized jobs, it becomes a challenge to structure those jobs by grouping them in a way that ensures both that the organization's mission is accomplished and that the efficiency of the operation is sufficient to ensure the survival of the organization.

2.3.2.1 Division of labor

A basic determinant for organizational performance is the grouping of tasks. Grouping is necessary to establish a system of coordination and supervision, of resource sharing, and of performance measurement (Mintzberg, 1979). The basis for grouping can either be by activity, output or customer. These three categories can be further decomposed – activity into function or skill, for instance. Most often, different bases for groupings are used at different levels in the organization. For instance, top management may be grouped according to function (marketing, finance, production etc.) the middle level according to market or product (or both) and production according to process or function. The reason is, that different criteria for grouping may apply at various levels. The criteria used for grouping normally reflect the interdependencies that are seen as most important. Mintzberg (1979) counts four such interdependencies: work-flow interdependencies (between separate tasks or stages), process interdependencies (within the separate stages themselves), scale interdependencies (economies of scale) and social interdependencies (social interaction). When organizations are drawn by conflicting criteria, they may be accommodated by creating various kinds of matrix organizations. Specifically in product development projects, these matrix organizations could come in the form of cross-functional teams as will be described in subsequent sections.

Grouping is in itself the primary instrument for coordination. Intuitively, we try to group together those functions that seem to have the most immediate interdependencies. Physical proximity allows richer communication and generally, the richer the communication, the closer, swifter and more flexible the coordination. The primary group, where coordination is effected through informal communication and where feedback is immediate, is the building block of all organizations. Once a group starts getting larger, informal communication cannot support the necessary coordination between the larger units in the organization. To accomplish this, the organization has to communicate and process large amounts of information across groups and units. In fact, many authors view communication and information processing as the main bottlenecks for organized activities.

2.3.2.2 A taxonomy of coordinating mechanisms

The basic coordinating mechanisms as proposed by Mintzberg (1979) and Groth (1999) are listed and described below and highlighted in Figure 2-3:

- *Mutual Adjustment.* The basic coordinating mechanism that occurs naturally through informal communication between the group members when a group is sufficiently small. This mechanism demands a high volume of communication and ideally every member must communicate with every other member. The corresponding structural configuration is the Adhocracy, a creative, project oriented organization living in a complex and dynamic environment.
- *Direct Supervision.* When a group expands beyond control of mutual adjustment, the required volume of communication rapidly saturates human communication capacity. At this stage, someone must take the lead and start planning and directing the work of others. The corresponding structure configuration is the Simple Structure, a centralized organization in a simple, dynamic environment

with a strong leader who keeps the organization simple and informal. Direct supervision also breaks down fairly rapidly as the limitations to how many workers one person can direct and supervise is a few tens. This can be solved through delegation in an hierarchical organization.

- The other is to reduce the need for supervision in the first place which can be achieved through standardization. Standardization of work and standardization of skills.

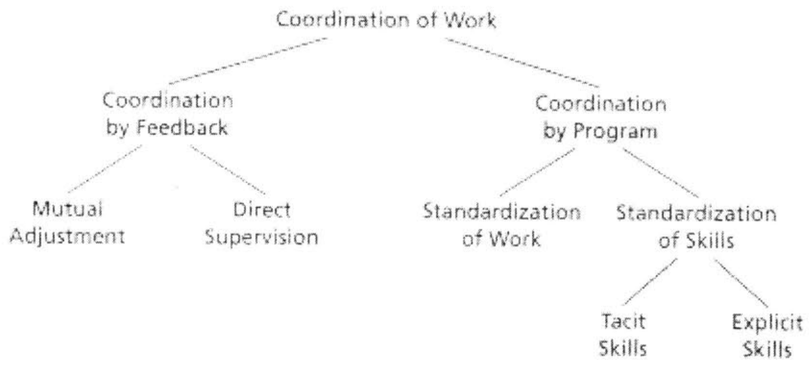


Figure 2-3 A taxonomy of coordinating mechanisms

(Source: Groth, 1999)

2.3.2.3 Information Processing and Communication Capacities

As organizations are needed to overcome the information processing and communication limitations of a group of individuals, organizations will seek to reduce the need for information processing and communication as much as possible. This is achieved through the mechanisms described above. However, information processing and communication is the very focus of product development projects (Galbraith, 1977).

As part of the planning process, managers will group tasks with great care, preplan as far as the environment allows, and they will only maintain the flexibility that is needed to cope with the variations the environment forces upon them. If they can, they will try to influence their environment to make it more stable, and if their competitive situation allows it, they may create slack resources or redundancy in order to tolerate lower, internal performance. Redundancy is important because it encourages frequent dialogue and communication (Nonaka, 1991). Redundancy also spreads new explicit knowledge. The organizational logic of redundancy helps explain why Japanese companies manage product development as an overlapping process where different functional divisions work together in a shared division of labor.

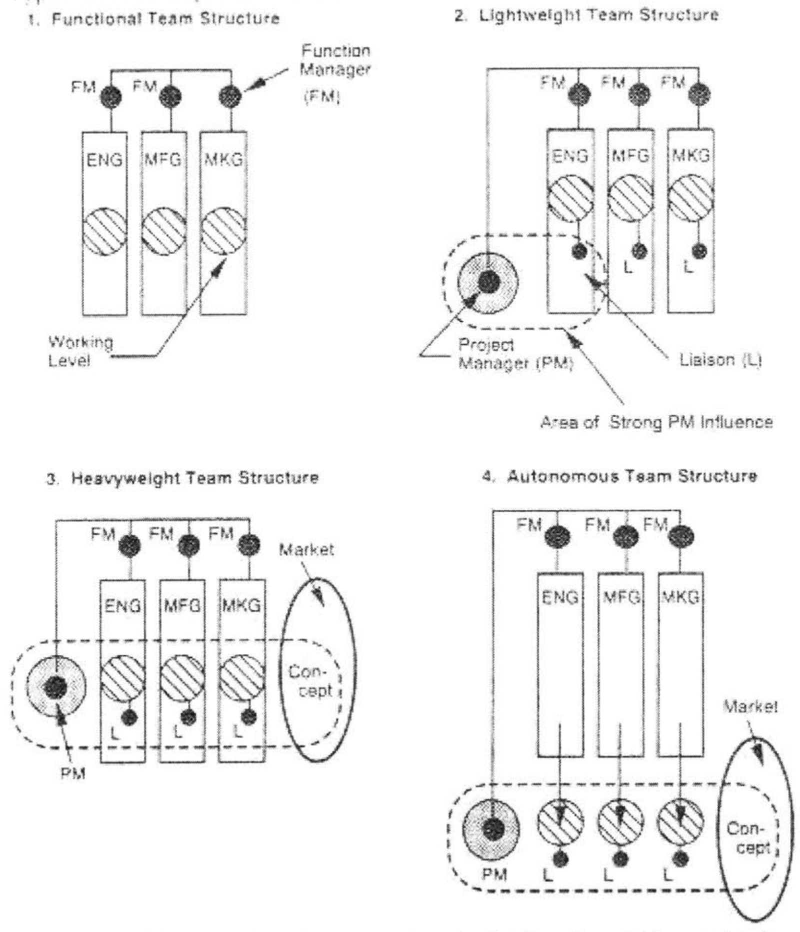
If these measures are insufficient, an organization will have to increase its information processing capacity in order to cope. Two main alternatives, increasing vertical information processing capacity or creating lateral relations, are open (Galbraith, 1977). The information processing capacity of a hierarchy is bound to be quite limited - if it attempts to coordinate the activities of different units by communicating through the formal structure, the organizational hierarchy is easily overloaded. The development of lateral relations is therefore seen as the main remedy. There are four basic types of relations or liaison devices, ascending from liaison individuals (persons with special responsibility to inspire cross-unit coordination by informing about certain aspects of their units' activities), to cross-unit committees or task forces (same purpose with more comprehensive participation), to integrating managers or departments (similar with a stronger mandate and more clearly defined responsibility) to the full matrix organization (there may be two, sometimes three intersecting chains of command). In that regard, many of the recent initiatives in improving product development project performances are about knowledge and organizational efforts.

2.3.3 Project Organization

2.3.3.1 Type of organizational structure

Wheelwright and Clark (1992) have identified four dominant organizational structures around which project activities can be organized. The four basic types of development team structures are the functional team, the lightweight team, the heavyweight team and the autonomous team and are illustrated in Figure 2-4. Each type has its own unique strengths and weaknesses. These are listed in Table 2-3.

Types of Development Teams*


1. Functional Team Structure
Function Manager (FM)
FM
ENG
MFG
MKG
Working Level
2. Lightweight Team Structure
L
Project Manager (PM)
Liaison (L)
Area of Strong PM Influence
3. Heavyweight Team Structure
Market
Con-cept
PM
4. Autonomous Team Structure

* This exhibit depicts four levels of teams: functional, where the work is completed in the function and coordinated by functional managers; lightweight, where a coordinator works through liaison representatives but has little influence over the work; heavyweight, where a strong leader exerts direct, integrating influence across all functions; and autonomous, where a heavyweight team is removed from the function, dedicated to a single project, and co-located.

Figure 2-4 Types of development teams

(Source: Wheelwright and Clark, 1992)

	Strengths	Weaknesses
Functional	• Managers control both performance and resource • Able to leverage prior knowledge	• Limited coordination and integration • Localized optimization rather than as a system
Lightweight	• Stronger coordination with other teams	• Less authority
Heavyweight	• Even stronger authority	• More authority
Autonomous	• Focus • Cross-functional integration	• Expands boundary of project definition • Risk to form new business unit • Possibly redesign of entire product rather than utilize existing opportunities

Table 2-3 Strengths and Weaknesses of different development teams

2.3.3.2 Diversity of team members

Recent studies also provide evidence that diversity influences both internal processes and external communications both of which are related to the project performance. The factors that have been found to be important are the diversity of tenure (McCain et al., 1983; Wagner et al., 1984; O'Reilly and Flatt, 1989) as well as diversity of function (Calanton and Cooper, 1981; Cooper, 1979; Voss, 1985). Similarity of tenure positively affects communication within the team (Goodman et al. , 1987; Zenger and Lawrence, 1989). Functional diversity positively influences communications outside the team (Dougherty, 1987). Members of similar functions share a common language and orientation, which makes communications easier (Kiesler, 1978). So, a high level of homogeneity within a group is likely to increase the cohesiveness and communication within the group (Festinger, 1954; Hoffman, 1985; Newcomb, 1961; Ward et al., 1985), but this same homogeneity acts to retard external communication (Ancona, 1987; Katz, 1982), which is the key in cross-functional teams. Since both internal and external communications are positively related to performance, diversity acts both positively and negatively in affecting performance. The impact of diversity on project performance therefore is quite complex.

In a study involving 45 product development teams in the computer, analytic instrumentation and photographic industries, Ancona and Caldwell (1989, 1990) have identified the correlations between team diversity, team size and project performance. The results of their findings can be summarized in Figure 2-5 and Figure 2-6. These results are consistent with others researchers findings (O'Reilly and Flatt, 1989, Dougherty, 1987) and provides a dilemma as to whether cross functionality is really beneficial. In order to sufficiently solve this dilemma, a more holistic framework will be necessary.

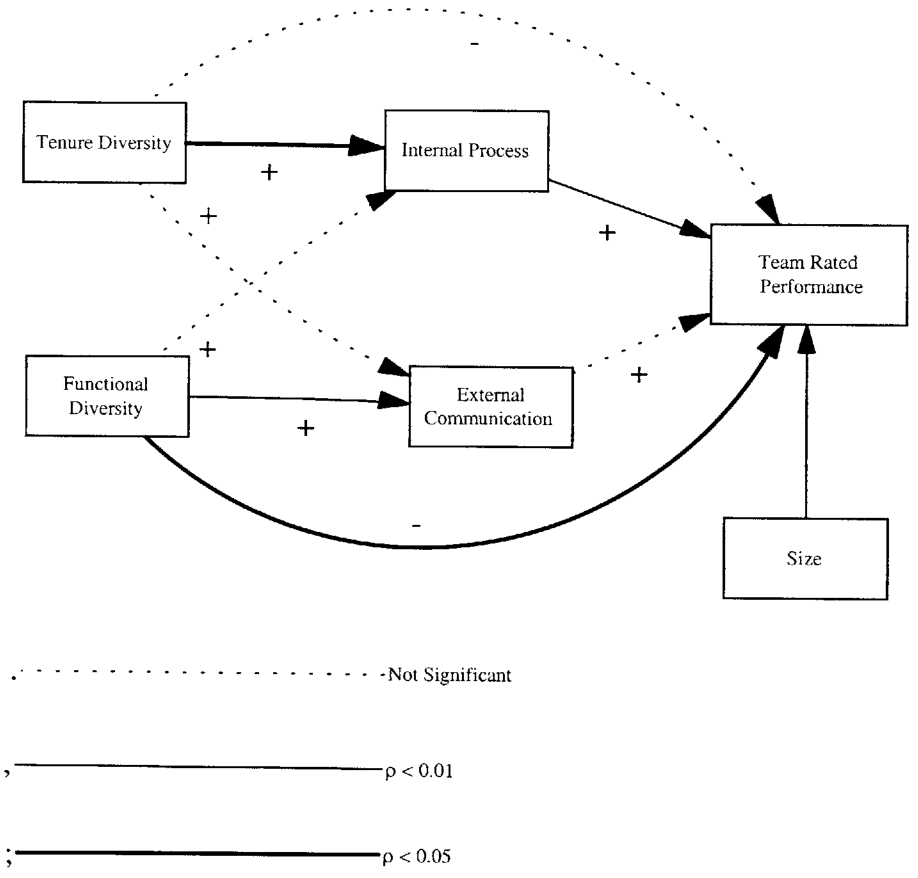


Figure 2-5 Team Diversity on Team Rated Performance

(Source: Ancona and Caldwell, 1989, 1990)

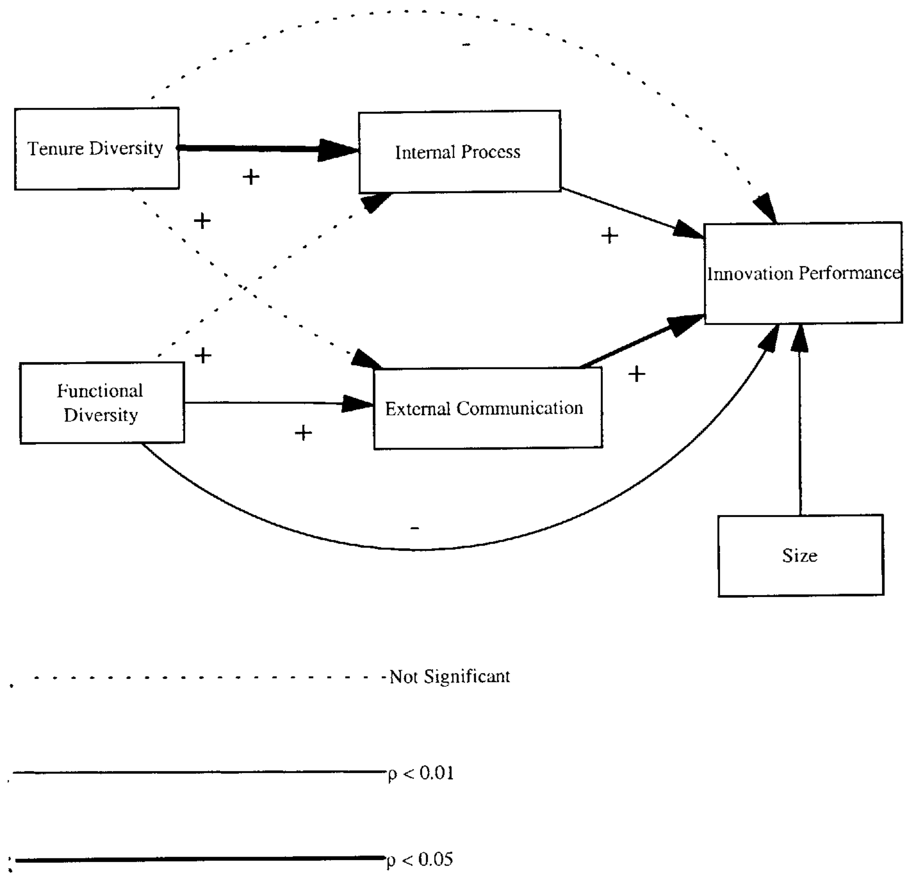


Figure 2-6 Team Diversity on Innovation Performance

(Source: Ancona and Caldwell, 1989, 1990)

2.4 Literature Review on the product development process

2.4.1 The traditional product development process

Traditionally, product development processes and their respective project organizations are based upon a sequential and functional approach to development (Wheelwright and Clark, 1992; Zaccai, 1991). These have been described by many researchers with given examples from industry (Wheelwright and Clark, 1992; Womack et al., 1990; Nevins and

Whitney, 1989; Hayes et al., 1988). In this paradigm, the development process is a series of development activities from conceptualization to product introduction. .

In this traditional product development process, the three traditional measures of project success are time, cost and quality/scope. These measures are increased or decreased to improve total project performance. This can take the form of trading performance among the three measures in a zero-sum environment (Rosenau and Moran, 1993). Substandard project performance can be caused by friction among functional groups, little and poor coordination and bottlenecks in the flow of products through the development process (Ulrich and Eppinger, 1994; Hayes et al, 1988). Researchers have also observed pitfalls in the traditional methods where development project performances fall below expectations (Wheelwright and Clark, 1992). These include a constantly moving target, mismatches between functions, lack of product distinctiveness, problem solving delays and unresolved policy issues.

In response to overcoming these symptoms, recent initiatives in the management of product development projects have included initiatives with concurrent engineering, the use of cross-functional teams and the development of product platforms. These attempt to enhance the performance of development projects through the solving these pitfalls directly. On the surface, the logic behind these initiatives seems apparent. In some cases, there have been significant improvements to the adoption of these new methods (Merrils, 1991; Nevins and Whitney, 1989). However, aggregated results have been fairly mixed (Iansiti, 1993; Clark and Wheelwright, 1993; Dean and Susman, 1991). Researchers have attributed this contradiction to increased complexities and tightened constraints imposed by the interdependencies requiring higher levels of coordination (Ulrich and Eppinger, 1994; Malone and Crowston, 1990).

Before we can understand why these initiatives have provided mixed results in actual implementations, it is first necessary to evaluate what the value proposition intended to be.

2.4.2 Recent initiatives in product development projects

The traditional paradigm of project management has its limitations and has begun to cause problems in this day and age where product life cycles are becoming shorter due to competitive pressures. Clark and Fujimoto (1991) describe the traditional paradigm as appropriate "...when markets were relatively stable, product life cycles were long, and customers concerned most with technical performance". The recent market and technology changes and the limitations of traditional project management methods have led to the development of a new image of effective product development (Nevins and Whitney, 1989; Hayes et al., 1988). Although this new image is still emerging, its central features have been articulated by researchers and applied by industry. The new paradigm fundamentally alters both the product development process and organization. Researchers currently envision product development as a collection of highly coupled activities that are performed iteratively and simultaneously by cross-functional product development teams (Ulrich and Eppinger, 1994; Wheelwright and Clark, 1992; Womack et al., 1990). The dominant change in the product development process from the traditional to the new paradigm is from sequential activities to concurrent activities (concurrent development). The dominant change in the development organization from the traditional to the new paradigm is from functional departments to cross-functional development teams. The dominant change in the product system architecture is from the development of single products to the development of a family of products (product platforms).

2.4.2.1 Concurrent development

Concurrent development's primary purpose is cycle time reduction. Concurrent development improves cycle times by planning, facilitating and executing multiple development tasks simultaneously instead of sequentially as in the traditional development paradigm. This requires breaking each development activity into smaller tasks and starting downstream tasks as soon as all prerequisites are met. Figure 2-7 illustrates the fundamental difference between the traditional ("phased approach") development process and concurrent design ("overlapping approach").

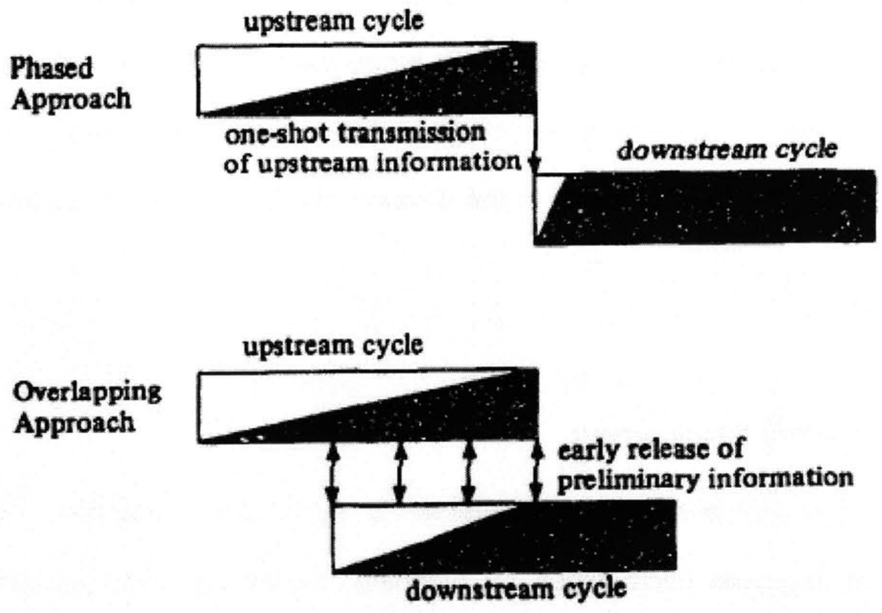


Figure 2-7 Sequential and Concurrent Product Development Processes
(Source: Hayes et al., 1988)

Large reductions in cycle time can be realized by applying concurrent development (Wheelwright and Clark, 1992; Womack et al., 1990; Nevins and Whitney, 1989). This cycle time reduction, however, is gained at the expense of increased complexity (Smith and Eppinger, 1995; Wetherbe, 1995; Osborne, 1993; Ford and Sterman, 1997; Ford, 1995).

2.4.2.2 Cross-functional development teams

A primary purpose of cross-functional teams is improved quality and effectiveness through improved coordination. Cross-functional development teams are groups of development specialists from different functional domains who work together on a single development project. The formation of cross-functional development teams is an extension of the move away from functional-based groups to the matrix structures used in the traditional development paradigm. Hayes et al. (1988) describe and Wheelwright and Clark (1992) later refine a more detailed model of this shift with intermediate steps defined by the level of influence of project managers. Restructuring product development organizations away from function-based groups and toward cross-functional development teams has also become a widely used approach reducing cycle time (Clark and Fujimoto, 1991).

However, several researchers (Bacon et al., 1994; Clark and Fujimoto, 1991; Dean and Susman, 1991; Takeuchi and Nonaka, 1991) and many firms (e.g. see Clark and Fujimoto, 1991 pg 105) have realized that the formation of cross-functional teams alone does not improve cycle time. They identify overextended managers as a contributing factor in cross-functional team failures. Reasons cited by other researchers vary. Dean and Susman (1991) found friction between members of the team from design and manufacturing. Wheelwright and Sasser (1991) cite a lack of planning due to a lack of information. Nevins et al. (1991) identified a lack of cross-functional skills in team members and no one taking responsibility for coordination. Clark and Fujimoto found an automobile development team consisting of only liaison people and no developers. The team failed because it was ignored by those developing the product.

Other researchers (Klein, 1994; Ancona and Caldwell, 1991; Wheelwright and Clark, 1992) have also attributed less than expected improvements on product cycle time to the increased complexity in managing cross-functional teams. Factors such as diversity increasing chances of internal conflict yet increasing the team's ability to communicate with the outside world are ironies that contribute to such complexities (Ancona and Caldwell, 1991). There is also the dilemma that having a broader base of expertise within a team compromises the functional depth in that team as well as the complication that expertise within the team changes over time. These factors increase the complexity of management as there needs to be assurance that there is a fit between individual, team and organizational needs (Klein, 1994).

Cross-functional work can be classified into two categories (Wheelwright and Clark, 1992). Systems to systems cross-functions are when there are interactions between different sub-systems of the total system. The focus is on system integration and on the interfaces between sub-systems. Systems to process cross functions, on the other hand, are when there are interactions between the upstream processes and downstream processes (e.g. design and manufacturing). With these cross-functional interactions, the intention is to reduce coupling in the system architecture but unfortunately increase coupling within the organization. Wheelwright and Clark (1992) point out that it is not enough just to have members from each functional group, there needs to be cross-element teams and communications. The complexity is so prevalent that Ancona and Caldwell (1991) point out that there is no clear evidence that cross-functional teams actually improve productivity.

Though there is not much research of the impact of system architecture on cross-functional teams. The impact on development cycles is quite apparent. It does therefore become beneficial to break the product down into systems that reduce the number of cross-chunk interactions (Pimmler and Eppinger, 1994; Suh.1990). This suggests that a more holistic approach is necessary and that the arrangement of subsystems is actually quite relevant.

2.4.2.3 Product Platforms

Meyer and Lehnerd (1997) define product platforms as "a set of subsystems and interfaces that form a common structure from which a stream of derivative products can be efficiently developed and produced. The assumption is that resources spent on executing development of single products is better off spent on executing the development of a platform of which a host of derivative products could be effectively and efficiently created. Some examples of successful product platforms include Black and Decker's power tools, Hewlett Packard's Ink Jet Printers and the Sony Walkman which produced more than 160 products when they were introduced between 1980 to 1990 (Sanderson and Uzumeri, 1996).

In response to the market's increasing competitiveness to develop products cheaper, faster and of higher quality, it becomes more essential for companies to generate a continuous stream of products. Product platforms are planned so that a number of derivative products can be efficiently created from the foundation common core technology. In many ways, the use of product platforms does not deviate from the two earlier concepts of concurrent engineering and cross-functional teams. Developing a product platform is in fact concurrent engineering in developing multiple products at the same time. In the implementation, it is necessary also to use multidisciplinary teams which is very much inline with the concepts of cross-functional teams.

Besides concurrency and multidisciplinary teams, the architecture of product platforms also has a role to play in more effective development projects. Herbert Simon (Simon, 1969) shows that the evolution of complex systems from simple elements depends on the numbers and distributions of potential intermediate stable forms. Hierarchical systems with stable subsystems (as opposed to integrated systems) will develop faster and of better quality. The classic analogy of two Swiss watchmakers is given as an example. While both watchmakers made fine expensive watches and were in equal demand, one prospered while the other struggled. One watchmaker assembled watches individually bit by bit and would need to put down the partly assembled watch and develop from scratch each time there was an interruption. The other watchmaker made watches by constructing subassemblies of about ten components, each of which held together as an independent unit. Ten of the subassemblies could then be fitted together in a subsystem of higher order and ten of those constituted the whole watch. If an interruption happened, instead of starting all over again, the watchmaker merely had to reassemble that particular subassembly. Similarly, product platforms are about managing the development and redevelopment of the subsystems rather than develop the overall system from scratch with each new product. As a result, the overall architecture of the product including how the subsystems interact with each other is crucial in the design stage. In fact, the common product architecture can improve development cycle times by facilitating a more streamlined development process and more frequent model changes (Clark and Fujimoto, 1991c; Smith and Reinersten, 1991)

Meyer and Lehnerd (1997) propose a model for effectively managing the evolution of product platforms. Coined the "power tower", an integrative model for managing innovation, considers three essential elements of the enterprise including the market applications, the product platforms and the technical and organizational building blocks. Figure 2-8 shows how these three components fit together.

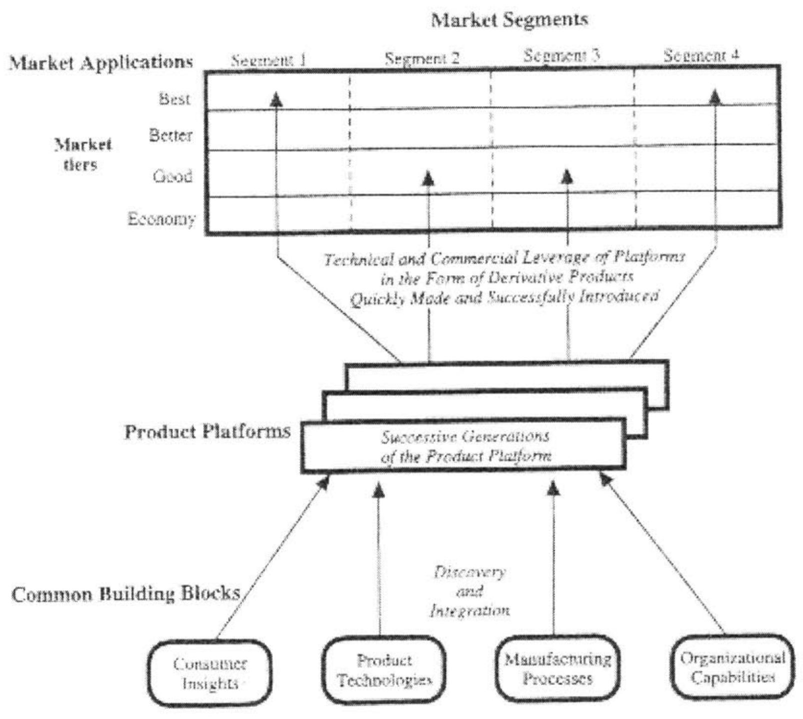

Market Segments
Market Applications
Segment 1
Segment 2
Segment 3
Segment 4
Market tiers
Best
Better
Good
Economy
Technical and Commercial Leverage of Platforms in the Form of Derivative Products Quickly Made and Successfully Introduced
Product Platforms
Successive Generations of the Product Platform
Common Building Blocks
Discovery and Integration
Consumer Insights
Product Technologies
Manufacturing Processes
Organizational Capabilities

Figure 2-8 The Power Tower
(Source: Meyer and Lehnerd, 1997)

As can be seen, the technical capabilities and organizational capabilities work are both crucial to successful product platform management. How organizational differences affect product development cycles, seem to be somewhat lacking in the literature. Meyer and Lehnerd (1997) also propose composite teams encompassing the essential skills to create a robust platform. This is much in-line with the literature on cross-functional teams.

2.4.3 Literature Review on System Architecture Issues

Under traditional product development processes, the primary measures for success have been time, cost and quality. As described in previous sections, there are several levers that can be used to increase product development project performances. Traditionally, these have been organization and sequencing. The use of cross-functional teams and concurrent engineering initiatives are in fact using these levers. The use of product platforms, a more recent approach seems to be using yet another lever which, in the opinion of this author, does not seem fully exploited. This lever is the product's system architecture.

System architecture is basically the arrangement of the functional elements into physical blocks, (Ulrich and Eppinger, 1994). An alternative definition describes it as the embodiment of concept, and the allocation of functionality and definition of interfaces among the elements (Crawley, 1999). Yet another definition describes it as the structure, arrangements or configuration of system elements and their relationships necessary to satisfy constraints and requirements, (Boppe – Crawley, 1999). In simpler terms, the system architecture defines the system's elements (sub-systems or components) and their relationship with each other (e.g. how they would interface).

System architecting includes not only improving the performance of the product, have implications on its production cost but also impacts the performance of the product development project (Rechtin and Maier, 1997). The system architecture would eventually not only have impacts on the projects sequencing, it will also have impacts on the project's organization which is why the system architecture is such an important lever in managing the product development project performance.

It has been suggested that system architecture uses, manages and balances the three themes in architecture. These themes are basically simplifying complexity, resolving ambiguity and focusing creativity (Crawley, 1999). More specifically, system architects are involved with:

- Removing/reducing/resolving ambiguity at the interface with the upstream process.
- Manage complexity and reduce apparent complexity
    - Manage through abstraction, decomposition and hierarchies
    - In a poorly architected system, complexity will rise down the downstream process.
- Focusing creativity

Based on these themes, the level of ambiguity, complexity and creativity vary over the life cycle of the product development project as shown in Figure 2-9. This concept is relatively similar to the knowledge evolution paradigm proposed by this research as levels ambiguity, uncertainty, complexity and creativity all change as knowledge of the system evolves and is developed into reality.

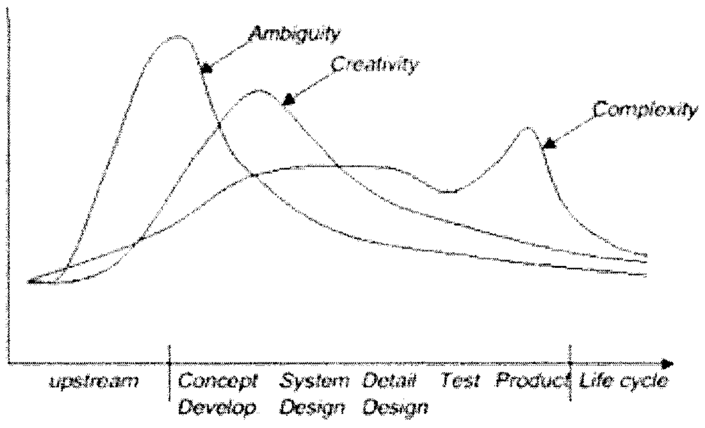


Figure 2-9 Progress of product development projects
(Source: Crawley, 1999)

2.4.4 Summary of product development process literature review

The existing literature describes and documents recent fundamental changes in product development processes and the related organizational structure from sequential, functional approach to a concurrent, cross-functional approach. Even though there are cases of significant improvements in product development projects, the aggregated results have proven to be fairly mixed (Isanti, 1993; Clark and Wheelwright, 1993; Dean and Susman, 1991). This suggests that even though such practices are moving in the right direction, there seems to be a gap of knowledge using current methodologies. Researchers have attributed this contradiction to increased complexities and tightened constraints imposed by the interdependencies requiring higher levels of coordination (Ulrich and Eppinger, 1994; Malone and Crowstone, 1990). In order to understand how these complexities and interdependencies interact to cause these results, this research proposes to analyze this product development process from a knowledge evolutionary paradigm as opposed to the traditional task based paradigm used by most project management tools today.

2.5 Literature Review on Project Management Frameworks and Tools

2.5.1 Critical Path Method and PERT

One of the most commonly used project management tools is probably the CPM (critical path method). It is a task-based tool that disaggregates the development process into activities or tasks that are related through their dependencies. Although initially developed for schedule control, it has been expanded to manage resources (and therefore costs)(Moder et al., 1984; Willis, 1986; Mueller, 1986) and is based on the traditional paradigm of development. Under CPM, each activity is treated as a monolithic block of work described only by its duration. The temporal dependencies describe the constraints which earlier (upstream) activities impose on later (downstream) activities. The constraints are described with relationships between the beginnings and completions of activities. The logic of the schedule can be represented in a network diagram. A simple example of a network diagram is shown in Figure 2-10.

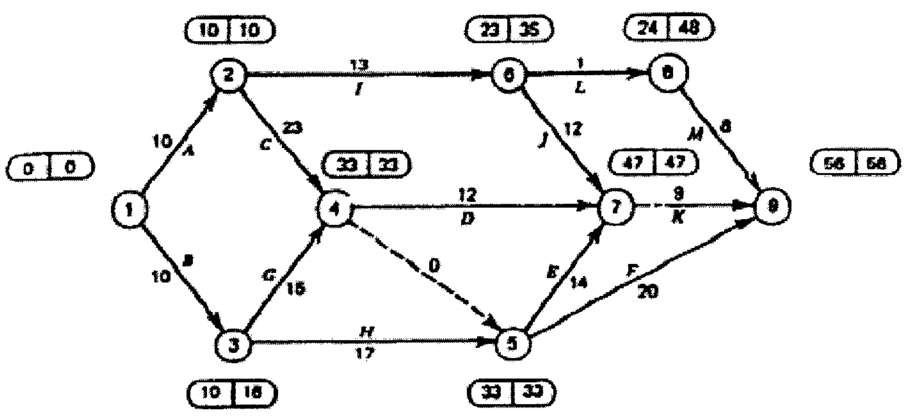


Figure 2-10 A network diagram for CPM schedule management
(Source: Halpin and Woodhead, 1980)

The fundamental basis of CPM is to identify a project's critical path, which is the sequence of tasks whose combined durations define the minimum possible completion time for the entire set of tasks. Earliest and latest possible start and finish dates of all activities within a schedule determined by the critical path can be calculated, as can the available slack (sometimes called float) times. The results of this planning and analysis can be presented for broader communications with a Gantt chart. An example of a Gantt chart from Moder et al. (1983) is shown in Figure 2-11.

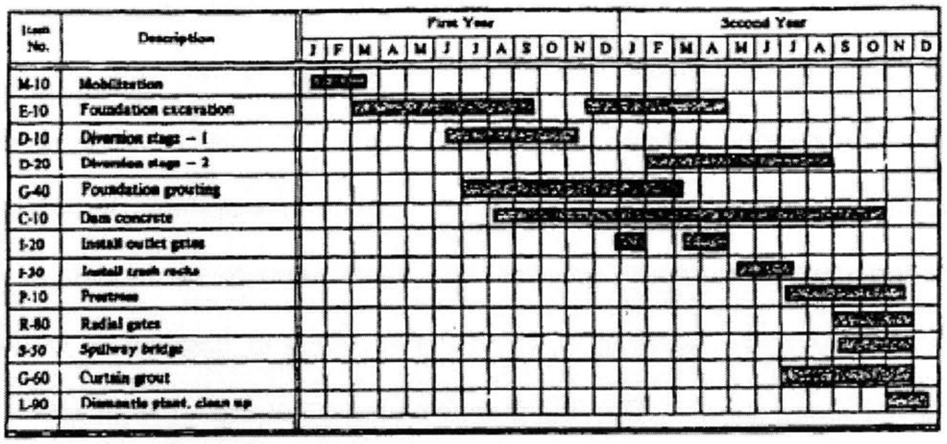


Figure 12-1 Bar chart for concrete gravity-arch dam.

Figure 2-11 A Gantt Chart Representation of a CPM Schedule

(Source: Moder et al., 1983)

Recent additions to the CPM provide several tools for trading away good performance in one measure in another. For example, resource constraints may be put in to the project model and the critical path based on satisfying these constraints can be recalculated. Thus, the durations of activities along the critical path can be shortened by adding more resources (Ulrich and Eppinger, 1994; Wheelwright and Clark, 1992; Moder et al., 1983). The CPM also provides a time-cost tradeoff method for analyzing the effectiveness of accelerating alternative activities. The effects of altering activity dependencies among activities to shorten the critical path can be investigated (Barrie and Paulson, 1984; Moder et al., 1983).

The CPM's ease of use and application is one of its key advantages. It provides a set of fundamental tools for characterizing and managing a development project in temporal terms. However, the method has critical limitations. The method assumes no rework of errors which are undiscovered when the phase is "completed" and that the rework of errors discovered within a phase's duration is incorporated into the phase duration estimate. The method cannot explicitly represent bilaterally coupled activities and therefore cannot describe loops, feedback or iteration in a system. In order to model iteration and rework, separate activities and tasks need to be modeled. Until the activities actually finish, however, the number of times of iteration or where rework occurs is not really known. One other weakness is that the CPM assumes that the product development project remains unchanged over time. This prevents the method from modeling time-varying and endogenous factors such as developer skill, training and coordination issues. Therefore the CPM is unable to model the highly coupled aspects and dynamic nature of the product development process. Finally, the CPM cannot describe the rationale that underlies the structure description and therefore lacks depth of information content.

PERT (Project Evaluation and Review Technique) uses an approach to schedule management which is similar to CPM. This method was developed for processes such as product development (Moder et al., 1983). PERT addresses one of the limitations of the CPM by incorporating the uncertainty inherent in the estimates of the durations of development activities into a scheduling tool. Three estimates (pessimistic, likely and optimistic) of project duration are used for each activity to model the variability of durations. The PERT method calculates the probabilities of a project meeting specific objectives. PERT incorporation of duration uncertainty makes it more valuable in managing less certain processes such as product development. However, PERT requires lots of data and is limited in accuracy by the estimates of variability of activity durations. Like CPM, PERT cannot explicitly represent coupled loops or feedback, assumes the project is static, and cannot model causes of process behavior.

2.5.2 Design Structure Matrix

In addressing the issues of concurrency and the iterative nature of couple processes, the design matrix was developed (Steward, 1981; Eppinger et al., 1990). The design structure matrix is a square matrix with the full set of development activities as both row and column labels. Each cell within the matrix represents a unidirectional dependency between two activities. Design structure matrices have been used to map (Smith and Eppinger, 1991) and predict (Morelli and Eppinger, 1993) information flows among activities. The matrix can be used to identify information flows as sequential, parallel or coupled and for the efficient ordering of development activities. Chao (1993) applied the design structure matrix to study the use of iterations in making time/quality tradeoff decisions. The focus was a portion of product development at a large semiconductor firm (DEC). Two strategies for making time/quality decisions (faster iteration and higher quality iteration) were proposed and tested.

Osborne (1993) applied iteration maps and the design structure matrix to describe product development at a leading semiconductor firm (Intel) in terms of cycle time. Osborne investigated variability in cycle times. His conclusions about the major factors of iteration on cycle time include iteration and project complexity. His work demonstrates the need for the additional investigation of the impacts of dependencies among development activities on cycle time. The design structure matrix is a potentially useful tool in describing and investigating information transfer and iteration for cycle time reduction. But the design structure matrix cannot directly model the structure of a development process over time. Like the CPM, the design structure matrix assumes that the dependencies between phases are fixed or that the distribution is fixed. Osborne's research supports other work which suggests that iteration in product development is a primary cause of the dynamic nature of product development process (Cooper, 1993a,b, c, 1994; Ford et al., 1993; Seville and Kim, 1993; Kim, 1988). Iteration is therefore suspected to be a primary driver of cycle time performance as well as a measure of quality.

2.5.3 Principles of Axiomatic Design

The traditional product development process is a general problem-solving paradigm that divides the decision making process into a series of stages, and these stages are typically outlined as follows: problem identification, formalization of goals, generation of solution concepts, evaluation of concepts and selection of the best solution and implementation. Implementation of this approach relies on hierarchical decomposition of a problem into sub-problems, and the generate-and-evaluate process is invoked iteratively at each level of the hierarchy (Mar and Palmer, 1989). Thus a system concept is initially identified to satisfy an overall set of requirements. This initial concept is then partitioned into sub-systems, components and interfaces that must be configured so as to satisfy the original set of requirements plus any sub-requirements related to serviceability, ease of manufacture and maintenance.

Though this provides a logical framework for design, the traditional process does not provide a prescriptive methodology for selecting an optimal solution. The principles of axiomatic design can be seen to formalize the concepts and provide prescriptive methodologies for selecting optimal solutions. The resulting design methodology is a consistent framework for attacking performance-based design problems at various levels of abstraction.

Axiomatic design is a formal strategy for partitioning and evaluating solution alternatives, without which it is difficult to advance directed, comprehensive solutions because there is no mechanism to prevent the designer from trying to force-fit traditional concepts. Axiomatic design (Suh, 1990) also defines the design process as a hierarchical activity where the solutions are based on a sequence of stages pertaining to problem definition, solution synthesis, solution evaluation and implementation. However, the advantage of axiomatic design versus traditional systems engineering is the fact that the decomposition and decision-making processes are made explicit by the following key concepts:

- Existence of different design domains.
- Definition of the decomposition process in terms of a zigzagging between design domains.
- Requirements that the best solution satisfies certain design axioms.

Firstly, the transition from an abstract statement of need to a physical entity is not linear but cyclical, depicted as a helix (Hubka, 1982; Suh, 1990). Successful transition requires continuous exchange of knowledge between and within different design domains. These domains are shown in Figure 2-12. Problem identification corresponds to a customer's statement of needs and occurs in the client domain. In the development of single-item products, such as a building, the client or his agent takes an active role: a) initiate the order for the project, b) express desired performance requirements, and c) establish project cost constraints. In the manufacture of mass-produced products, the client domain refers to a market of consumers, and their active input is often provided by proxy in the sales or product planning sections of the manufacturing organization. In either case, designers also use the client domain as a buffer for gaining an understanding of their particular tasks and for reasoning about the functional requirements for subsequent design stages or subordinate design disciplines.

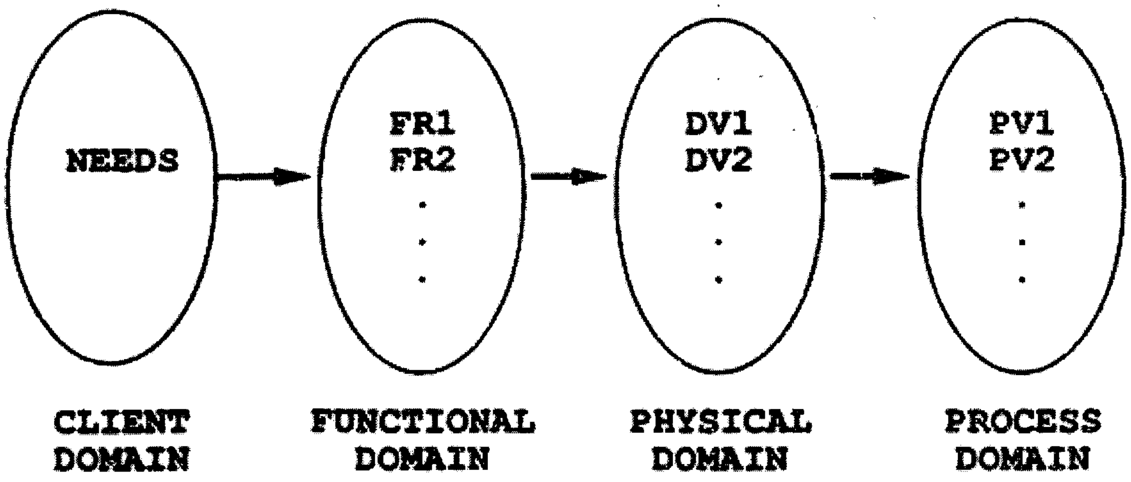


Figure 2-12 Design Domains (Axiomatic Design)

The output from the client domain is then translated in the functional domain into a combination of objective and subjective goals or functional requirements (FRs). In an office development project, for example, leasable space is often specified as an objective requirement related to economic feasibility and aesthetic properties are subjective criteria to promote tenant interest. Starting with the stated FRs, the design engineer generates a solution by defining design variables (DVs) so as to satisfy the FRs and describe the solution's physical make-up. This process of synthesizing a design solution constitutes a mapping between the functional and physical domains. In turn, the DVs generated in the physical domain are interpreted as a set of requirements for implementation in the process domain. The engineer responsible for designing a construction plan or a manufacturing process plan maps these requirements to a sequence of actions or process variables (PVs) to affect a result.

Secondly, the concept of decomposition refers to the fact that the output of each design domain evolves from abstract to concrete ideas in a top-down hierarchical manner. The FRs for a design project are decomposed into a hierarchical structure of sub-objectives. The total design description at any level of the hierarchy consists of all components, attributes and relationships needed to satisfy the stated sub-objectives. Various sources (Hubka, 1982; Hubka and Eder, 1988; Pahl and Beitz, 1988) recognize that the mapping process must take place over a number of levels of abstraction so as to define sufficient design data to support construction or manufacturing activities. For example, Hubka (1982) recommends typically six levels of design detail while Pahl and Beitz (1988) identify three. In some systemic approaches to design, the conceptual design stage involves decomposition of the overall functional requirement(s) into an appropriate set of sub-functions followed by a one-to-many mapping between each sub-function and feasible function carriers. Solution concepts are obtained from the synthesis of various function carriers into composite structures or layouts. The feasible solutions are then detailed during the succeeding stages of design refinement in order to support the original sub-functions and to satisfy other design requirements, such as durability, aesthetics, and ease of production.

Axiomatic design also recognizes that the mapping between functional requirements and design variables must take place over a number of levels. Functional decomposition is continuous and precedes the generation of design variables at all levels of abstraction. (Suh, 1990). Thus the reasons for concept refinement during the later design stages are made explicit. In addition, design for ease of construction or ease of manufacture is considered concurrently because the hierarchical decomposition in one domain cannot be performed independent of the evolving hierarchies in the other domains (Suh, 1990). As a result, prior to decomposing a parent set of FRs, the FRs must be mapped successfully to a set of DVs and these DVs must be mapped, in turn, to an appropriate set of PVs. The decomposition between the functional, physical and process domains are shown in Figure 2-13.

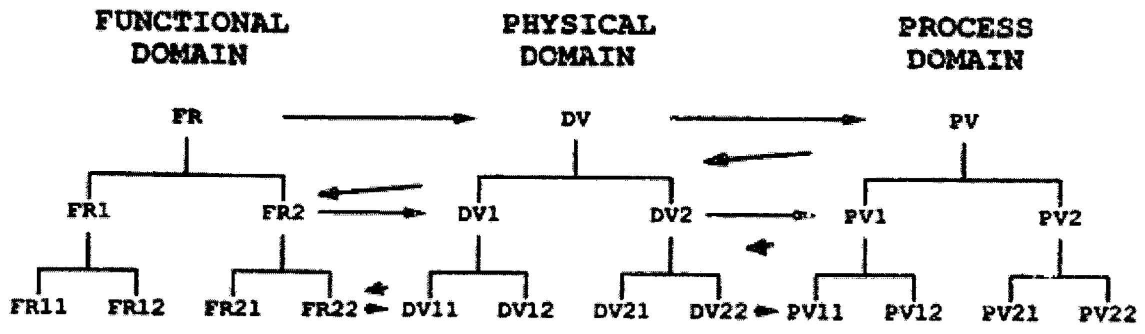


Figure 2-13 Hierarchical structure and Zigzagging

Continuous zigzagging through the process or construction domain is a robust strategy for advancement of manufacturable solutions. It allows concurrent development of design and construction plans. In addition, the manufacturability of the product can be reviewed in terms of the required systems and activity sequences, rather than relying on the detailed calculation of the man-hour and material expenditures necessary for constructing the discrete building elements. The latter approach to improved manufacturability is a sub-optimization scheme, and empirical studies (Griffith, 1984) have indicated that reliance on such elemental approaches does not significantly reduce total costs.

The most important concept in axiomatic design is the existence of design axioms which provide a rational basis for the evaluation of proposed solution alternatives and the subsequent selection of the best solution. The intent of the design axioms are to allow explicit objective reasoning about the merit of each proposed design solution and the identification of the best solution from among many alternatives.

The first axiom focuses on the mapping between "what is required" (FRs or DVs) and "how do we achieve it" (DV's or PVs). The second axiom establishes information content as a relative measure for comparing alternative solutions. The axioms may be stated in the following procedural form (Suh, 1990):

AXIOM 1 The independence axiom

In an acceptable design, the mapping between the FRs and DVs is such that each FR can be satisfied without affecting any other FRs.

AXIOM 2 The information axiom

Among all proposed solutions that satisfy axiom 1, the most optimum design has the minimum information content.

2.5.4 Systems Dynamics

2.5.4.1 Dynamic Modeling of Projects

Whilst some research approaches (such as the Design Structure Matrix) identify some dynamic consequences of different project structures on project performance, a description of the process structure in the form of the causal relationships that generate project behavior is needed to investigate how project processes drive performance. The system dynamics methodology, developed by Jay Forrester (1961) at the Massachusetts Institute of Technology captures such causal relationships between variables and their impacts on each other in an ever changing environment.

2.5.4.2 System Dynamics Models of Product Development Projects

Several researchers have built system dynamics models of product development. Roberts (1974) built a relatively small (30 equations) project model which investigated the management of R&D projects. The primary material which flows through the product development process is generic "job units". Completion of job units is based upon perceived progress, which includes both actual progress and perceptual errors. These decisions include manpower changes, which directly impacts the job unit completion rate.

Cooper (1980) and Reichelt (1990) described the construction and use of large system dynamics models by Pugh-Roberts Associates of large scale shipbuilding operations for claims settlement. Cooper's model included product development in each of the two projects modeled. This model focused upon rework caused by customer changes. Manpower was primary cost driver and therefore key to the model structure. The model simulates the major phases of shipbuilding operation. Cooper modeled (1980) and subsequently modeled (1994, 1993a,b, c) the impacts of rework in projects on cycle time. The model simulates the initial completion of development tasks (basework) and corrective action (rework). A delay in discovering defects slows the completion of unflawed development tasks. The process structure propagates change across project phases with interdependent schedules and disruptions which reduce quality and require rework. Reichelt (1990) describes the dependence of downstream project activities on the completion of upstream activities in a two-stage process. Cooper and Reichelt's work adds three valuable concepts:

1. the ability of customers to influence cycle time and increase coordination needs in product development processes;
2. the distinction between direct (first-order) and indirect (higher-order) impacts and
3. competition for resources among product development activities.

Richardson and Pugh (1981) developed and explained in detail a model focusing upon the management of single R&D projects. Richardson and Pugh use rework to expand on the resource effectiveness portion of the fundamental structure used by Roberts. They distinguish between work performed satisfactorily and tasks requiring rework. Both satisfactory tasks and rework are modeled explicitly. This allows the incorporation of new and potentially important influences on cycle time and coordination such as the error rate in development activities and the rate at which errors are discovered. Richardson and Pugh identify the impractical nature of some cycle time reduction policies such as assuming no rework or assuming a constant project scope. They use their model to illustrate the effects of different assumptions and policies on cycle time.

Abdel-Hamid (1984) modeled software development to better understand project management in light of cost overruns, late deliveries and user dissatisfaction. The model simulated software production as influenced by human resources management, planning, and controlling. In this model schedule pressure influences resource quality through the prediction of work force size necessary to complete the project on schedule. The model's schedule pressure influences resource effectiveness through productivity, the error generation rate and worker allocation to quality assurance.

Jessen (1990) investigated the behavior and impacts of project manager motivations with a model based upon resources, rework, targets and resource strategy. This model focused on the roles of goal seeking (balancing) feedback loops in projects. It expands the description of the motivational structures in projects.

Homer et al. (1993) modeled project process structure explicitly by introducing "gate functions" to describe the constraints on work progress imposed by both preceding phases and the work within phases. This model uses graphical table functions to describe these precedence relationships in more detail than possible with the Critical Path or PERT methods. For example, two phases can be described as very tightly coupled with the upstream phase limiting the work available throughout the duration of the downstream phase, not just limiting the start or finish of work as in the critical path methods. Homer et al.'s model uses both available work and resources to constrain progress. The information prerequisites in the model described in this work has its foundation in the Homer et al. model.

Seville and Kim (1993) built a model based on Kim's earlier model (1988) of product development at a computer hardware company. These models simulated the flow of product development tasks through two stages: product design and process design. Seville and Kim use different levels of coordination between product and process engineers with an exogenous coordination fraction decision. Seville and Kim contributed an explicit structure for modeling the coordination effects on resource quantity and effectiveness. They also used a two-stage ageing structure and modeled the impacts of factors such as productivity to each stage separately.

Ford et al. (1993) studied the interface between two product development groups within an electronic entertainment equipment manufacturer (Ford and Paynting, 1995). The model focused on the relationships among coordination, schedule and quality. They explicitly modeled required rework due to errors due to errors and optional iteration to meet a quality goal. This allowed them to incorporate the influences of schedule pressure on decisions about iteration for quality. This illustrates modeling cycle time reduction as a tradeoff between time and project quality. Ford et al. add a distinction between required and voluntary iteration in product development.

Whilst these researchers have covered the three domains (monitoring and control, rework and human resources) as identified by Rodrigues and Bowers (1996), it is Ford and Sterman (1998) that tie these together with the process structure. Ford and Sterman (1998) develop a product development model which explicitly models all four performance drivers – process structure, resources, targets and scope. They have demonstrated the importance of distinguishing process dynamics from other project dynamics and have added development process to the three domains identified by Rodrigues and Bowers (1996). More specifically, Ford and Sterman (1998) have identified process concurrence relationships as an important aspect impacting process, resource, scope and targets on project performance.

2.5.4.3 Significant System Dynamics structures

In the various system dynamics models cited above, several underlying structures are found to cover the four performance drivers mentioned. As in general system dynamics models, although individual cycles do not describe the behavior of the system as a whole, the impact of the individual cycles are profound. These cycles are described below:

- The labor structure

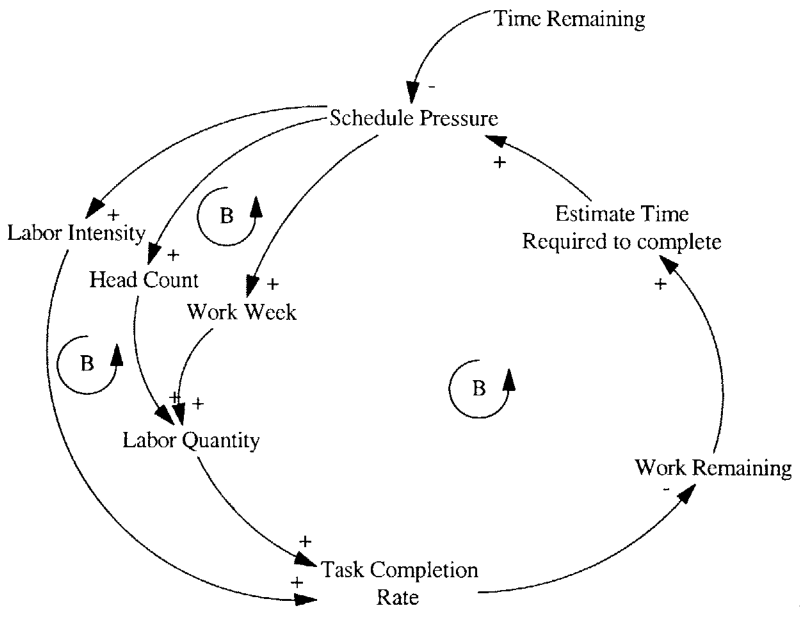


Figure 2-14 The Labor Structure

The labor structure includes three balancing feedback loops which increase labor effort as a response to schedule pressure. The response can adjust the amount of effort (Labor Quantity), how hard people work (Labor Intensity) as shown in Figure 2-14. Although existing models generally assume that labor is the dominant resource in product development projects, development resources other than labor can be modeled with the same structure (e.g. Cooper, 1980)

- The schedule structure

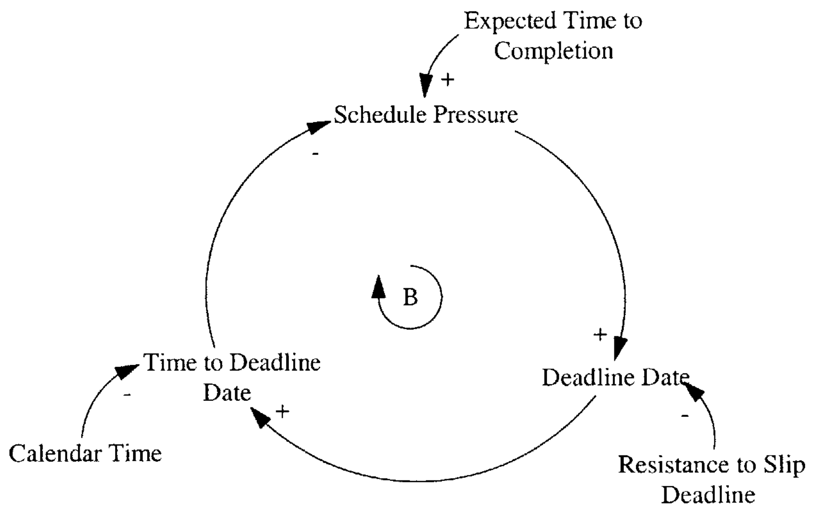


Figure 2-15 The Schedule Structure

The schedule structure as shown in Figure 2-15 describes another common project management tool, slipping the deadline in response to schedule pressure. Many models include this feedback loop (Roberts, 1974; Richardson and Pugh, 1981; Abdel-Hamid, 1991).

- The rework structure

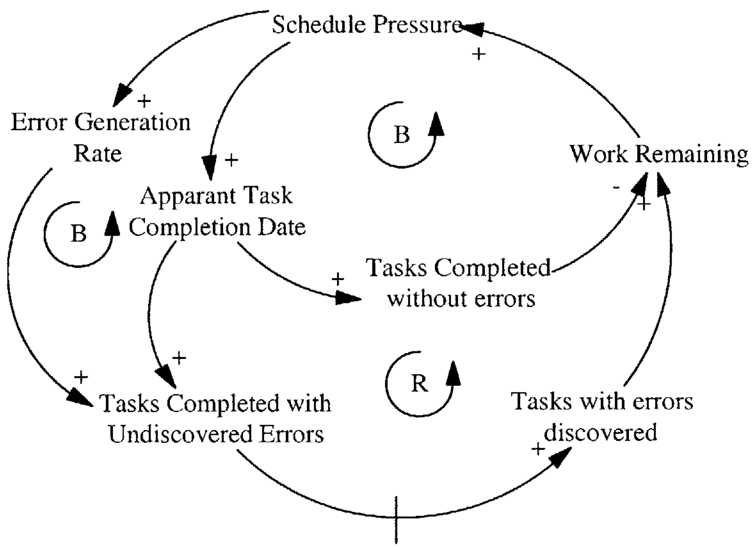

Schedule Pressure
+
+
Error Generation Rate
+
B
B
Work Remaining
Apparant Task Completion Date
-
+
+
Tasks Completed without errors
+
+
R
Tasks Completed with Undiscovered Errors
Tasks with errors discovered
+

Figure 2-16 The rework Structure

The rework structure describes the process where work perceived to be completed is not actually complete due to errors. It is distinguished from quality by the fact that rework must be done whereas quality work is optional. Although not solely responsible for, rework explains many symptoms we see in projects. For example, the 90% syndrome which reflects why a project is persistently 90% complete is explained through the rework structure as 90% is always perceived to complete but little "real" progress is made due to rework (Ford and Sterman, 1999).

- The available work structure

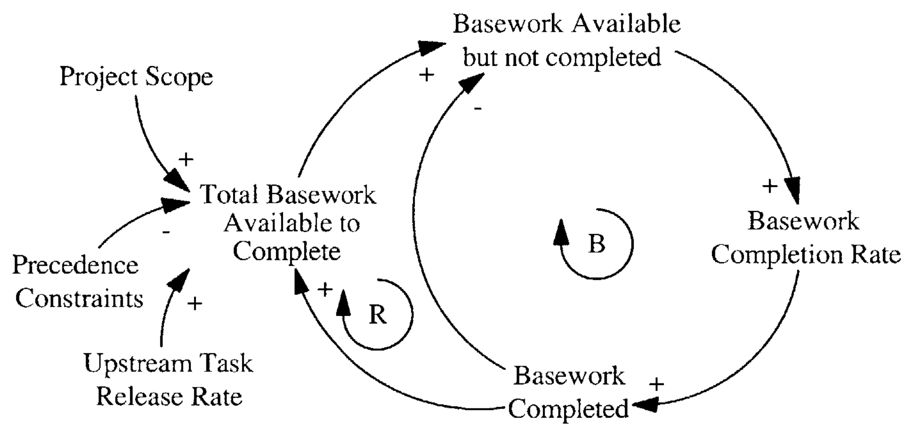

Project Scope
Basework Available but not completed
Total Basework Available to Complete
Precedence Constraints
Upstream Task Release Rate
Basework Completion Rate
Basework Completed
R
B
+
-

Figure 2-17 The Available Work Structure

The performance of projects can be constrained by the availability of work. This structure explains the critical path in the CPM/PERT methodology where work cannot proceed if precedence constraints are not met. The structure shown in Figure 2-17 shows that work can only proceed based on the resources available (basework completion rate). However, if basework is not available (due to precedence constraints not met), only available basework is executed.

- The quality structure

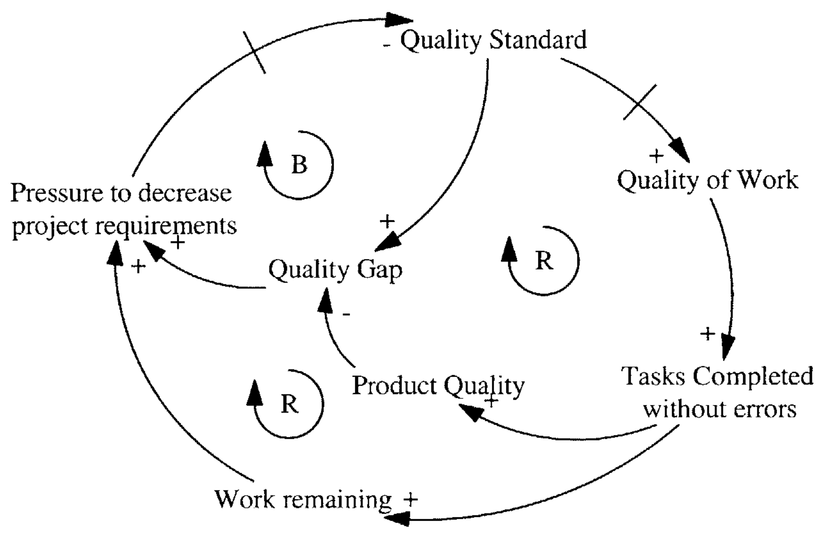


Figure 2-18 The quality structure

The quality structure represents the repetition of tasks undertaken to meet quality requirements (Quality Standard). This repetition is distinguished from those in the rework structure where the repetition is mandatory and required for basic product functionality. Repetition in the quality structure is "optional" and can be adjusted by reducing project requirements. This is particularly influential as project managers have major impact on number of iterations through the setting of quality targets.

- The scope structure

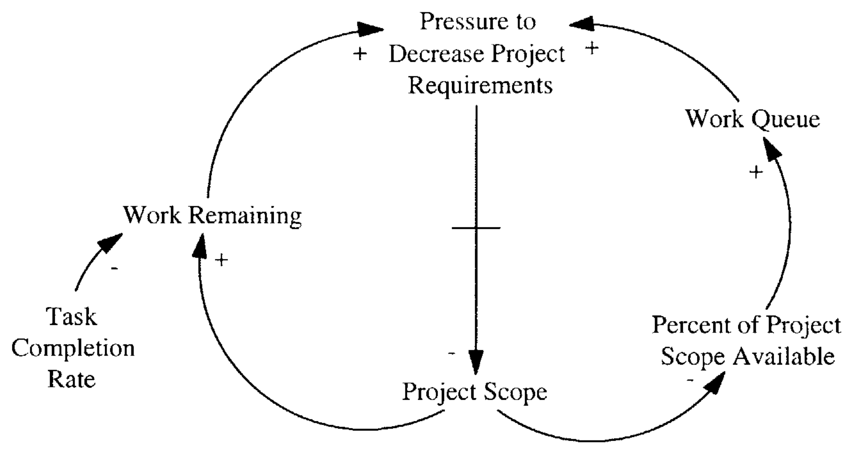

Pressure to Decrease Project Requirements
Work Queue
Work Remaining
Task Completion Rate
Percent of Project Scope Available
Project Scope
+
+
+
+
+
-
-
-

Figure 2-19 The Scope Structure

The scope structure as shown in Figure 2-19 represents the adjustment of project size.

2.5.4.4 Evaluation of System Dynamics Models of Product Development Projects

Although current research ties together the four performance drivers – process structure, resources, targets and scope with the six major project structures. The impacts on project performance due to organizational structure and system architecture have not really been studied. This research intends to develop a framework that ties in the organizational structure and system architecture to the project structure. This would be achieved through changing the model structures from a viewpoint of tasks processed to a viewpoint of knowledge transfer between team participants.

2.6 Summary of Literature Review

This chapter has covered some of the recent research in the fields of knowledge management, product development, organizational theory and project management tools. Recent initiatives in product development projects, particularly concurrent engineering, cross-functional teams and product platforms have been used to improve product development projects. Results, however, have been fairly mixed. This indicates that there may be some missing ingredient to improving product development project performances.

The principle of knowledge being embodied within artifacts provides the underlying philosophy of being able to track the collective knowledge in a product development project as it evolves over time. Although not normally viewed as such, the recent initiatives in product development processes can also be viewed upon as knowledge management initiatives by ensuring knowledge within the project organization is easily accessed and disseminated quickly once further knowledge is generated.

The review of project management frameworks and tools has shown that most of these tend to be task-based (i.e. representing the project as a breakdown of tasks). As a result, these frameworks fail to capture the iterative nature of these projects. Although recent use of dynamic simulation tools does model rework, this tends to be primarily rework due to mistakes, error or failure to meet quality standards. It cannot be imagined, however, that iteration will not happen in the absence of mistakes, error or low quality. In fact, any basic text in product design will indicate that the process is an iterative one.

The iteration process in product development project is one that is not captured or represented in current research and project management methods.

2.6.1 Research Needs

The lack of accounting for iteration and rework in projects accounts for a big chunk of the underperformance of product development projects. The common saying that an accurate estimate of project duration is to take your best guess and multiply by two holds true in many cases where the additional factor is used to account for rework, change in scope and other forms of iteration.

As traditional product development project management techniques are task based, the only means of modeling change in scope, rework and iteration is to place in additional tasks or activities. However, one seldom understands how much iteration needs to be modeled for. It is also impossible to estimate the duration of these tasks. Further, rework and iteration themselves are repetitive in nature and placing in additional tasks and activities in exactly the same format, does not really add value to the project model. If there is a change in scope, it becomes even more difficult to model the breakdown of tasks and activities since one does not know ahead of time how the scope will be changed and what type of tasks are needed.

Just as a basic determinant for organizational performance is the grouping of tasks, there is no doubt that organizational performance is a key determinant of project performance. It is, unfortunately not the only determinant. Given a pre-determined set of tasks, organizational structure is a very important factor in the project management process. Rarely, however, do product development projects come prescribed with a pre-determined set of tasks. How the project performs also depends greatly on how the project is broken down into tasks. Very often, tasks are broken down in accordance with the organizational structure. It can therefore be appreciated that product development projects are basically coupled problems where tasks need to be broken down according the organizational structure and where the organizational structure is designed according to the task breakdown at the same time.

To circumvent this problem, one possibility is to have product development project models that model the information flow and knowledge generated. Such a framework would also need to account for the iterative nature of projects as well as the coupled nature between system architecture, task breakdown and project organizational structure.

This research proposes such a framework. By tracing the evolution of information and knowledge about the product and systems in product development projects, we seem to be able to account for most of the issues mentioned above.

2.6.2 Proposed Knowledge Evolution Framework

Under the proposed knowledge evolution framework, the state of information and knowledge over the duration of projects is tracked. The information and knowledge intended to be tracked encompasses all information and knowledge generated including those embodied in documents, prototypes, and partially developed products and in the processes that produce products.

In this framework, the knowledge of the product evolves over time as it is transformed from conception in a few individuals to detailed engineering knowledge in more individuals to knowledge embodied in physically into the products/prototypes. The process by which knowledge evolves is one where individuals constantly create and transfer the knowledge to other members in the project team. The other members will make use of the existing knowledge, assimilate with other sources of knowledge and create new knowledge before codifying and transferring to other members. Since most knowledge is rooted in tacit knowledge, the infrastructure for transfer of knowledge and information needs to suit the purpose of acquisition, creation, transfer and utilization of information and knowledge needed the product development process.

At any state, there is always uncertainty about the knowledge that has been generated up till that time. This uncertainty does vary over time as well. As the knowledge evolution framework suggests, this uncertainty is the underlying trigger for iteration. Its evolution is just as important and is one of the underlying themes of this research.

Chapter 3 Model Description

3.1 Introduction

This chapter develops both the knowledge evolution framework as well as the model used for the analysis of projects. The knowledge evolution framework presents a model of a product development project from the viewpoint of how knowledge pertaining to the project is developed and evolves. This is opposed to traditional project management frameworks that are task-based and need explicit task definition to model the project. The knowledge evolution framework presents a fresh vantage point and provides further insight into product development projects.

The background theory of the knowledge evolution framework that is based upon other researchers' work is presented in the first part of this chapter. The second part of the chapter explains some of the specific features of this knowledge evolution model that is then used to model the Delta Design process in the subsequent chapter.

3.2 Theoretical Background

3.2.1 Communication

In the recent initiatives of the product development process as described in the preceding chapters, the one common recurring theme is communication. Information processing capacity forms one of the major bottlenecks in the product development process (Galbraith, 1977) and project organizations should be designed to either maximize the capacity or to reduce the need for communication as much as possible. While concurrent engineering is based on utilizing as much of the information processing capacity of a project organization as possible, concurrent development teams minimizes the extent to which information needs to be moved around the project organization.

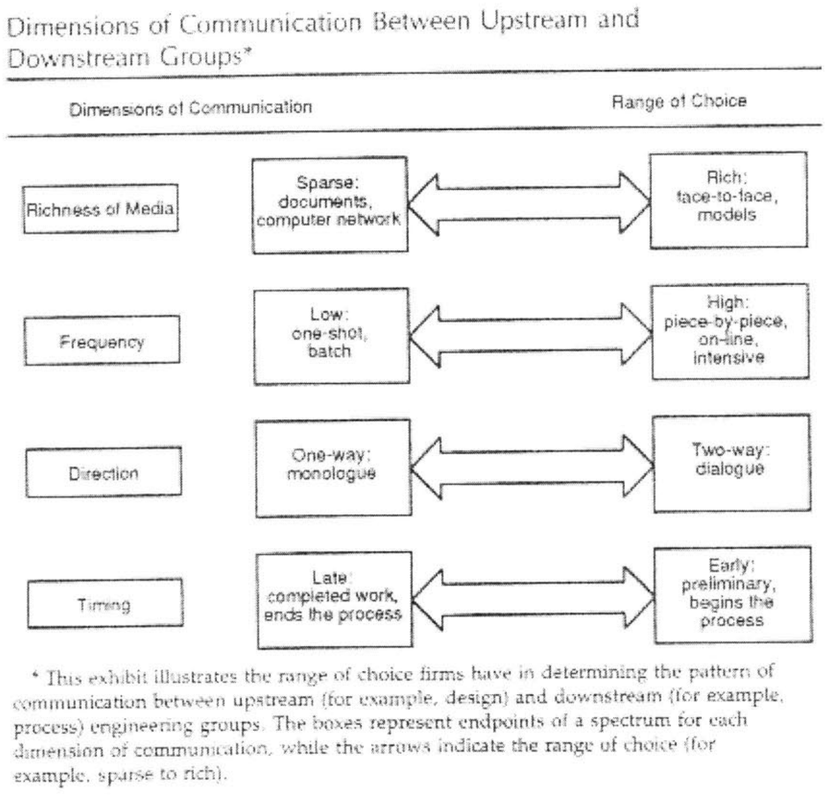
There are several measurements when we look at communication efficiency in project organizations. These can be listed as follows:

- Dimensions of communication
- Mode of communication
- Dispersion (or distance) of organization
- Diversity

These measurements are discussed briefly below.

3.2.1.1 Dimensions of Communication

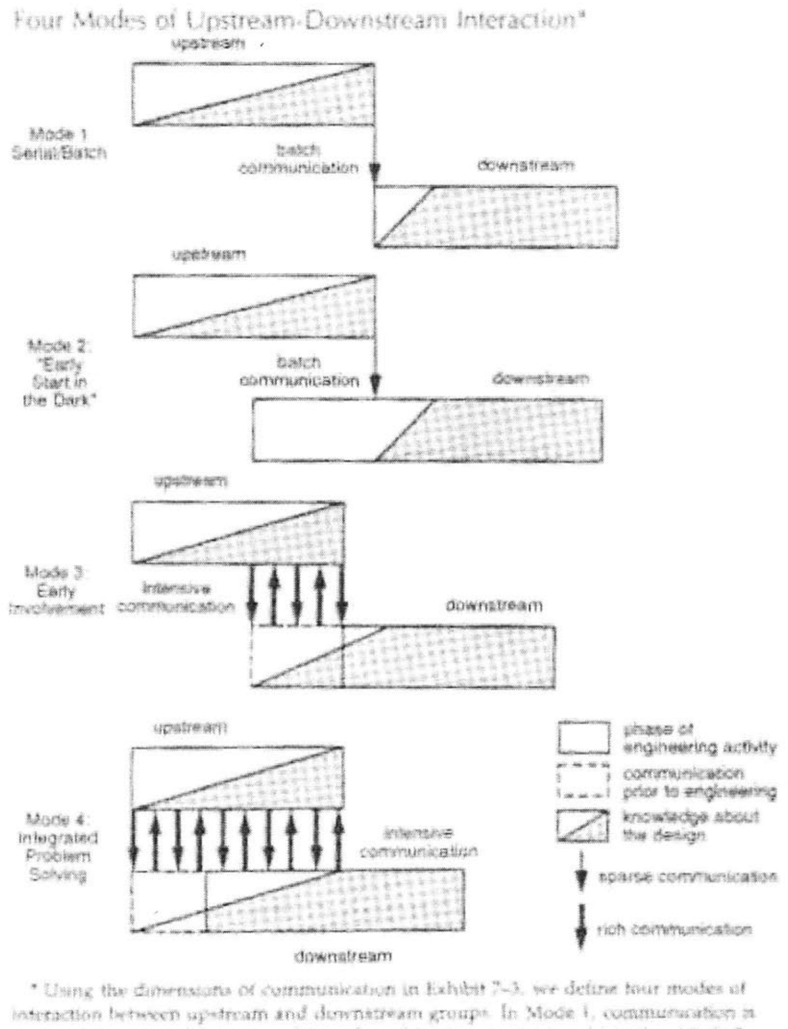
A critical element affecting the performance of the project team is the dimension of communication (Wheelwright and Clark, 1992). Four dimensions of communication pattern – richness, frequency, direction and timing – jointly determine its quality and effectiveness. Figure 3-1 presents the dimensions and the spectra through which communications can exist.


Dimensions of Communication Between Upstream and Downstream Groups*
Dimensions of Communication
Range of Choice
Richness of Media
Sparse: documents, computer network
Rich: face-to-face, models
Frequency
Low: one-shot, batch
High: piece-by-piece, on-line, intensive
Direction
One-way: monologue
Two-way: dialogue
Timing
Late: completed work, ends the process
Early: preliminary, begins the process

* This exhibit illustrates the range of choice firms have in determining the pattern of communication between upstream (for example, design) and downstream (for example, process) engineering groups. The boxes represent endpoints of a spectrum for each dimension of communication, while the arrows indicate the range of choice (for example, sparse to rich).

Figure 3-1 Dimensions of communications
(Source: Wheelwright and Clark, 1992)

3.2.1.2 Mode of communication


Four Modes of Upstream-Downstream Interaction*
upstream
Mode 1 Serial/Batch
batch communication
downstream
upstream
Mode 2: "Early Start in the Dark"
batch communication
downstream
upstream
Mode 3: Early Involvement
intensive communication
downstream
upstream
Mode 4: Integrated Problem Solving
intensive communication
downstream
phase of engineering activity
communication prior to engineering
knowledge about the design
sparse communication
rich communication

* Using the dimensions of communication in Exhibit 7-3, we define four modes of interaction between upstream and downstream groups. In Mode 1, communication is sparse, infrequent, one-way, and late; the information is serial and lengthy. Mode 2 maintains the pattern of communication, but moves up the starting point. Modes 3 and 4 make communication richer and more frequent (Mode 3) and starting much earlier (Mode 4).

Figure 3-2 Four modes of upstream-downstream interaction

(Source: Wheelwright and Clark, 1992)

Besides the dimensions of communication pattern, there is also the mode of communication. These are illustrated in Figure 3-2 and also affect the performance of the project organization, particularly where there is concurrency involved.

Concurrent engineering attempts to integrate upstream and downstream processes as much as possible. Based on task-based management tools such as CPM and PERT, integrating the processes is possible as long as pre-requisites for the downstream processes have been met. This is also accounted through the "Work Availability Structure" in the use of system dynamics in project management. Increasing project performance is therefore about ensuring that work is always available and thus optimizing project resources.

This research, however, will show that integration brings on added complexity through rework and iteration. So, even though pre-requisites are met, the quality and knowledge segregation implies less than ideal circumstances for an earlier start. This may ultimately lead to longer durations through more rework or iteration.

3.2.1.3 Dispersion of Organization

In a study of communications patterns of R&D laboratory workers, the most successful R&D groups include individuals who spend as much time communicating outside as inside the company. Such individuals search for new technologies, participate in the definition of industry standards and bring to the lab a wealth of competitive information (Allen, 1977). In the same study, Allen also noted that the probability that two people will communicate decreases dramatically as the distance that separates them increases. This is illustrated in Figure 3-3, which represents a regression from data of seven R&D organizations.

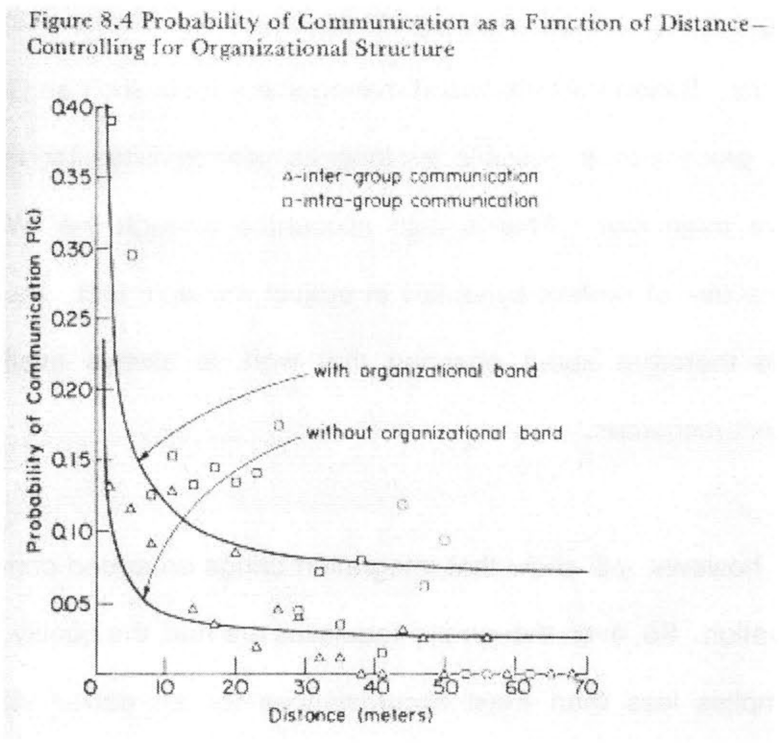


Figure 3-3 Communication and Distance

(Source: Allen, 1977)

Extending the concept of reduced communications when distance between participants are increased, it can be envisioned that any other similar "barriers" to communication would have a similar impact. Some of these "barriers" are listed as follows:

- Geographic distance
- Organizational boundaries
- Functional boundaries

These boundaries or “barriers” can be compressed into a single measurement which we coin organizational dispersion. Organizational dispersion not only measures the physical distance between participants in the project organization but also takes into account whether the participants are working cross organizationally and across boundaries. Each of these “barriers” represents increasing reduction in communication frequency, probability and volume.

Unlike R&D organizations, however, product development project organizations have pre-determined communication processes. Because of project goals, certain groups of participants, for example, will be required to hold regular meetings for communications. What increased dispersion means, however, would allow for increased formal communications and decreased informal communications. This has great impact on the impact of iteration and rework.

3.2.1.4 Diversity of organization

Based on the dispersion concept mentioned above, having uniform teams (in terms of functionality and organizations) makes more sense. This is, in fact, contrary to the use of cross-functional teams where organizational and functionality dispersion is increased but the geographic distance between participants is reduced.

Ancona and Caldwell (1989, 1990) have, in fact, found that diversity both helps and harms in product development project performance. Further, their results are consistent with other researchers (O'Reilly and Flatt, 1989, Dougherty, 1987). The reason for this paradox is the need to differentiate between communication within the team (internal) and communication outside the team (external) as discussed in the preceding chapter.

With this research, one can begin to understand why previous implementations of cross-functional teams have yielded mixed performances. Without fully understanding the causal effects of the team makeup, the results are truly not predictable. Unfortunately, it is not possible to model the organizational changes in traditional task-based models. The closest possibility is to reduce time to individual tasks due to closer collaboration and quicker comment throughput. This, however, does not reflect the whole picture. To understand the nuances behind the use of cross-functional teams, an information or knowledge evolution based view of the project must be used.

3.2.2 System Architecture

3.2.2.1 A coupled system at the root level

The literature review has suggested that system architecture has some bearings on both the project organization as well as the task definition for the project. Likewise, very often system architecture is determined by the skill sets that are available within the project organization (or anticipated project organization).

Figure 3-4 shows a simple diagram with the different causal relationships between the elements of the project structure.

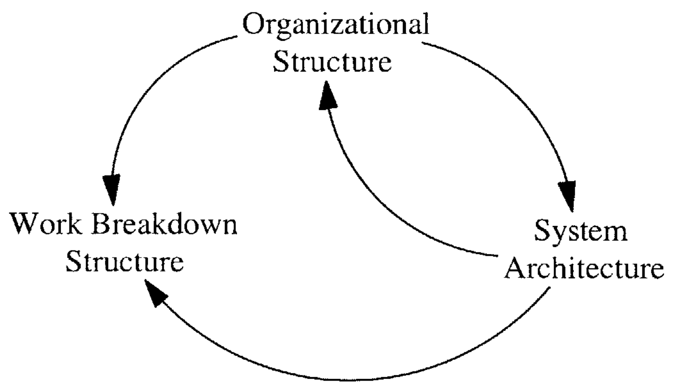


Figure 3-4 Elements of Project Structure

Indeed, many researchers propose that it will be more efficient if the process is decoupled (Steward, 1981; Suh, 1990). This is, however, difficult as the problem is coupled at the root level as indicated by Figure 3-4 above.

Over time, many types of product development projects have established traditions, standards and codes of practice. These are established to reduce some iteration by using acceptable industry norms. These industry “norms” dictate the way project organizations are formed since every role has its place and project procedures are used since “everyone is used to this way of working”. In an economy where project performance inefficiencies can be lived with, these norms provide structure without trial and error and unnecessary iterations. However, with the commercial pressures of today, any competitive advantage needs to be used. Unfortunately, in certain industries where such norms have become so entrenched, a different approach could yield less ideal results.

Many industrialists will recognize that often the system architecture and the organizational structure match each other in terms of breakdown and scope of work. Traditionally, organizational structure and system architecture reflect each other quite well. But does organizational structure drive system architecture or vice versa? The solution seems to lie in tradition. For example, in the design and construction of buildings, the system architecture of buildings have driven the specialization of trades leading to current industry relationships between architects, engineers and contractors. This "natural" progression seems to be a result of efforts to ease management through the division of labor and specialization of trade as discussed in Section 2.3 above. As artifacts and the development processes increase in complexity, it is observed that such "natural" project structures may no longer be optimal and "experimentation" with techniques such as concurrent engineering, cross-functional teams and knowledge re-use. The underlying common philosophy this research assumes is that the move towards the optimization of information and knowledge transfer as a means of increasing project performance.

It is important to distinguish between information flow and knowledge flow. Having efficient means of transferring information is no guarantee of better project performance. The information that is transferred needs to be relevant, generated in a timely fashion and delivered and made available to the right persons. Having efficient and effective knowledge management is thus imperative to increasing project performance.

Understanding not only how knowledge is created, utilized and transferred but what type of underlying organizational structure and system architecture are needed to optimize the knowledge activities is the philosophy of the underlying knowledge evolution framework.

3.2.2.2 Stable subsystems

Herbert Simon's watchmaker analogy (2.4.2.3 Product Platforms; Simon, 1969) demonstrates that hierarchical systems with smaller subsystems evolve quicker than large non-hierarchical systems. The speculations on speed of evolution were first suggested through an information theory estimating the time required for biological evolution (Jacobsen, 1995). Jacobsen's model suggests that the expected time required to reach a particular state is inversely proportional to the probability of the state – hence it increases exponentially with the amount of information of the state. Simon's model introduces the concept of levels and subassemblies, which produces much smaller estimates.

A system dynamics model is created based on Simon's model. In this model, the total number of parts in each product is 1000. The number of parts per assembly (ppa) represents the number of parts (on average) there are in each assembly. The number of parts per product divided by the number of parts per assembly therefore represents the number of sub-assemblies there are in each product. Figure 3-5 shows the model that was created with the equations in Appendix 1.

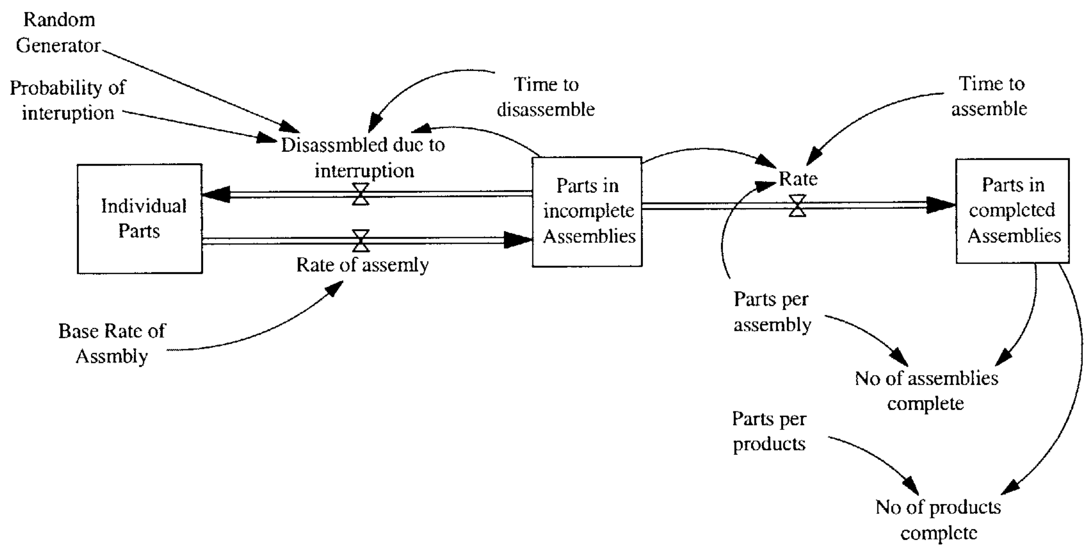


Figure 3-5 System Dynamics model of Herbert Simon's watchmaker analogy

Figure 3-6 shows the different rates of evolution for the different number of sub-assemblies. Even though the base rate of assembly of 10 parts/hour is held constant, the impact to the final rate of production varies quite significantly due to loss of work due to interruptions.

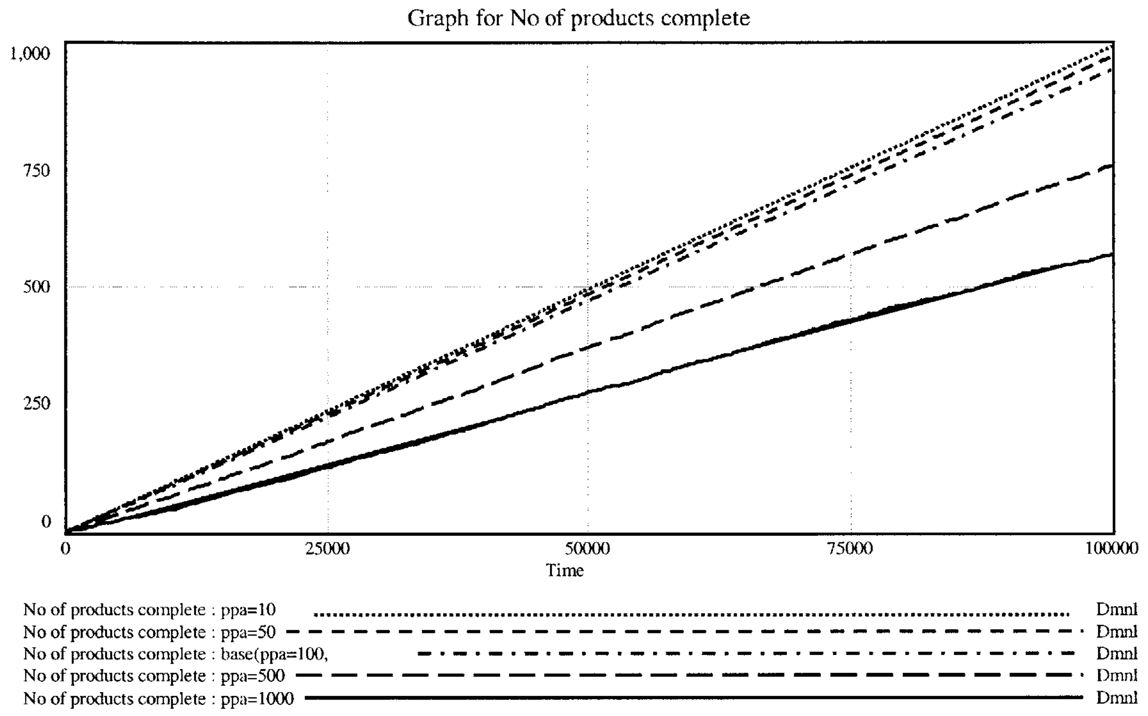


Figure 3-6 Results from Simon's Watchmaker Analogy

These results show that the lowest number of parts per assembly would yield the quickest rate. This is indeed similar to a rework cycle. In this case, rework is due to the inability to achieve a stable state prior to interruption. There have been other system dynamics models (Ford, 1995) that have indicated the impact of rework in the product development project duration. Though not exactly the same, the stable subsystem model indicated by Simon's watchmaker analogy further enforces this point.

It is this type of behavior that explains the 90% syndrome (Ford and Sterman, 1999). A scenario where the project reaches 90% completion pretty much in accordance to schedule and the remaining 10% of the work takes about the same time as the first 90% to complete. This results in project overruns caused the second half of the project working on both the final 10% of work as well as the rework to the initial 90%.

3.2.3 The Rework Structure and its impact on overlapping

There are further implications resulting from this rework structure. As discussed in Section 2.5.4.3 above, the rework structure explains many discrepancies we see between project estimates and actual project results. For example, Li (1999), Peña-Mora and Li (2000) and Eppinger (1994) describe a scenario where overlapping upstream tasks and downstream tasks do not necessarily result in a shorter overall duration. This phenomenon is described with the rework structure and its impact on the "quality of work done". This is illustrated through the system dynamics model shown in Figure 3-7 below (the equations are provided in Appendix 2)

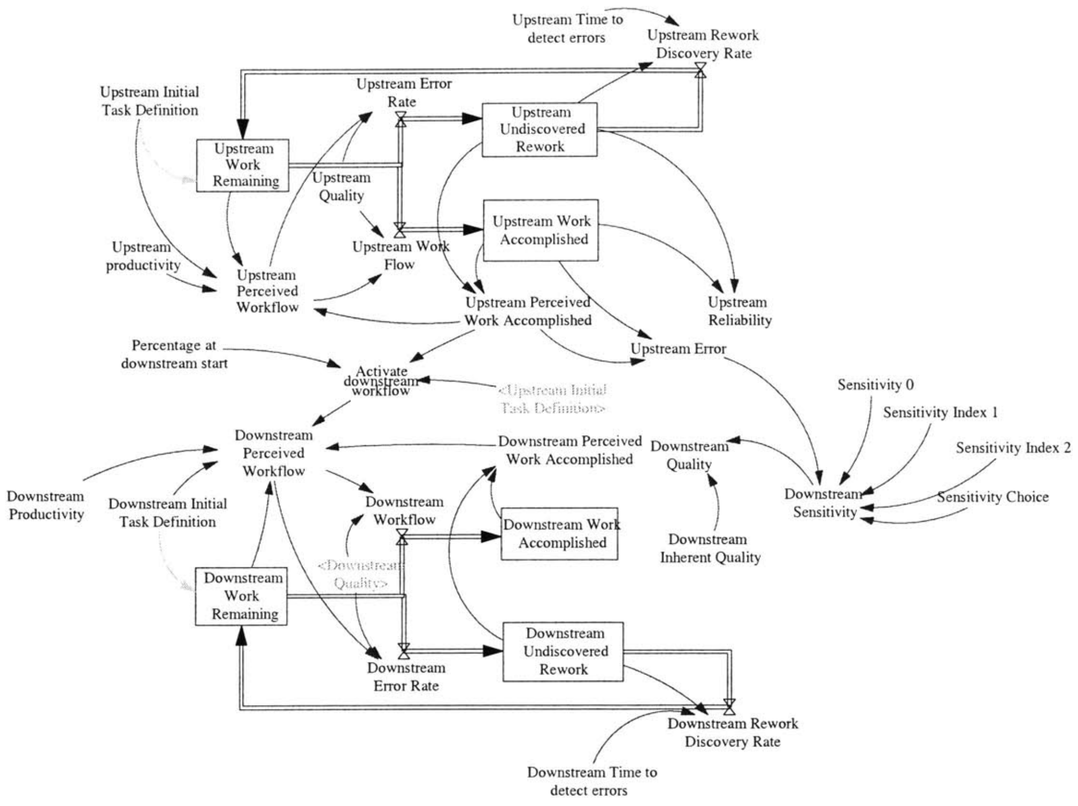

Upstream Time to detect errors
Upstream Rework Discovery Rate
Upstream Initial Task Definition
Upstream Error Rate
Upstream Undiscovered Rework
Upstream Work Remaining
Upstream Quality
Upstream Work Accomplished
Upstream productivity
Upstream Work Flow
Upstream Perceived Workflow
Upstream Perceived Work Accomplished
Upstream Reliability
Percentage at downstream start
Upstream Error
Activate downstream workflow
<Upstream Initial Task Definition>
Sensitivity 0
Sensitivity Index 1
Sensitivity Index 2
Downstream Perceived Workflow
Downstream Perceived Work Accomplished
Downstream Quality
Downstream Productivity
Downstream Initial Task Definition
Downstream Workflow
Downstream Sensitivity
Sensitivity Choice
Downstream Work Accomplished
Downstream Inherent Quality
<Downstream Quality>
Downstream Work Remaining
Downstream Undiscovered Rework
Downstream Error Rate
Downstream Rework Discovery Rate
Downstream Time to detect errors

Figure 3-7 Rework and Overlap

The model takes two simple tasks one upstream and one downstream. Work accomplished is differentiated between work actually accomplished and work that is deemed accomplished but is actually not. The distribution of work would depend on the work quality, which represents a percentage of work complete that is done without requirements for change. The behavior of the upstream task is shown in Figure 3-8 below.

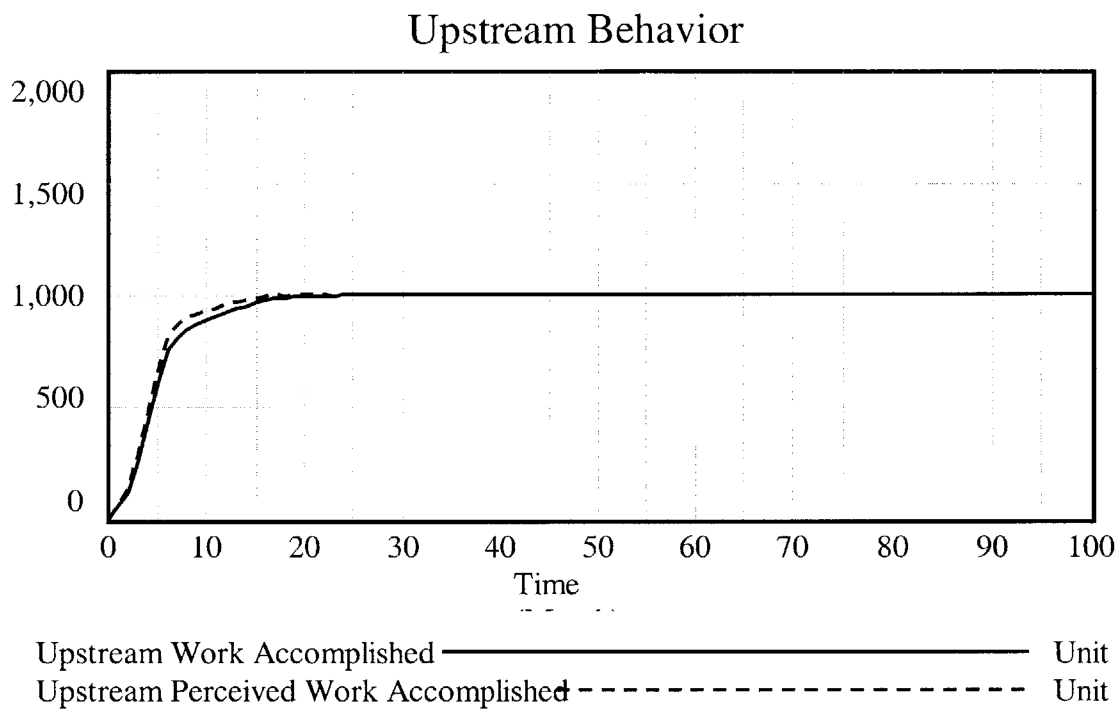


Figure 3-8 Behavior of upstream task in Rework and Overlap model

The downstream task starts at a point where the upstream task is perceived to have completed a certain amount of work. This is indicative of practice in industry where overlapping is implemented. Using the terminology of CPM, this represents a predecessor/successor relationship of start-to-start with a lag period. The lag, in this case, uses a percentage of perceived work accomplished rather than a fixed time period but basically achieves more or less the same intention.

Most project managers will, traditionally, use a lag period that is defined by two parameters. The first is that work is available for the downstream task. The second is that work will remain available for the downstream task. Since the downstream task cannot complete before the upstream task, there is no point in starting too early (by introducing more overlapping) only to find that you need to stop the downstream task at some point to wait for more work to become available. There are, of course, other considerations that need to be considered, such as resource availability.

What is not realized, however, is that based on the perceived work accomplished (since actual work accomplished cannot be determined), there are inherent errors introduced into the downstream task. This affects the quality of the downstream task significantly. Obviously, different types of processes impact the downstream quality differently. This is captured through what it termed the sensitivity. Basically, highly sensitive downstream tasks are affected tremendously by errors triggering off high impacts to quality (lower figures) with relatively low error rates. Figure 3-9 shows a set of fairly sensitive relationships that are used in this model indicated by the variables Sensitivity 0, Sensitivity Choice 1 and Sensitivity Choice 2. These are in increasing sensitivities. Insensitive relationships would result in lines above the Sensitivity 0 line.

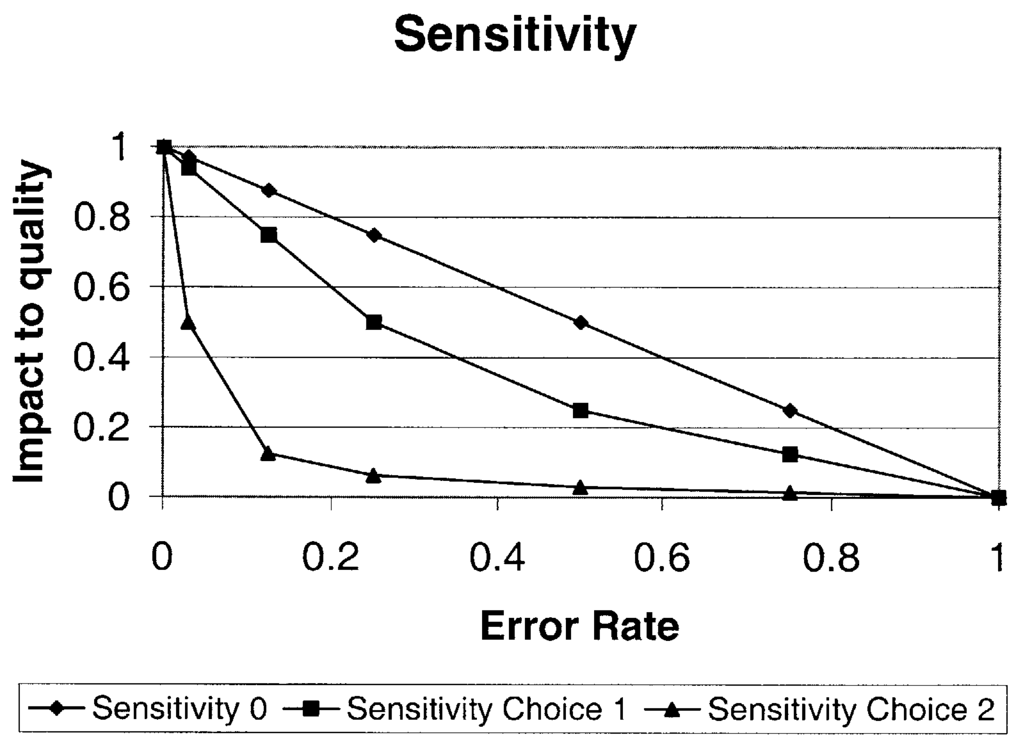


Figure 3-9 Relationship between error rate and impact to quality

The downstream process, like the upstream has a quality factor that indicates the percentage of work that is done right the first time round. This factor has two components. Firstly, the inherent quality which represents quality inherent to the task itself. That is to say, that in the absence of errors from the upstream process, certain errors will be generated inherently due to task considerations like quality control, people, processes etc. The second component is the sensitivity to upstream errors. Errors that are generated upstream do affect the downstream task and when they are spotted, they contribute to rework on the downstream task as well.

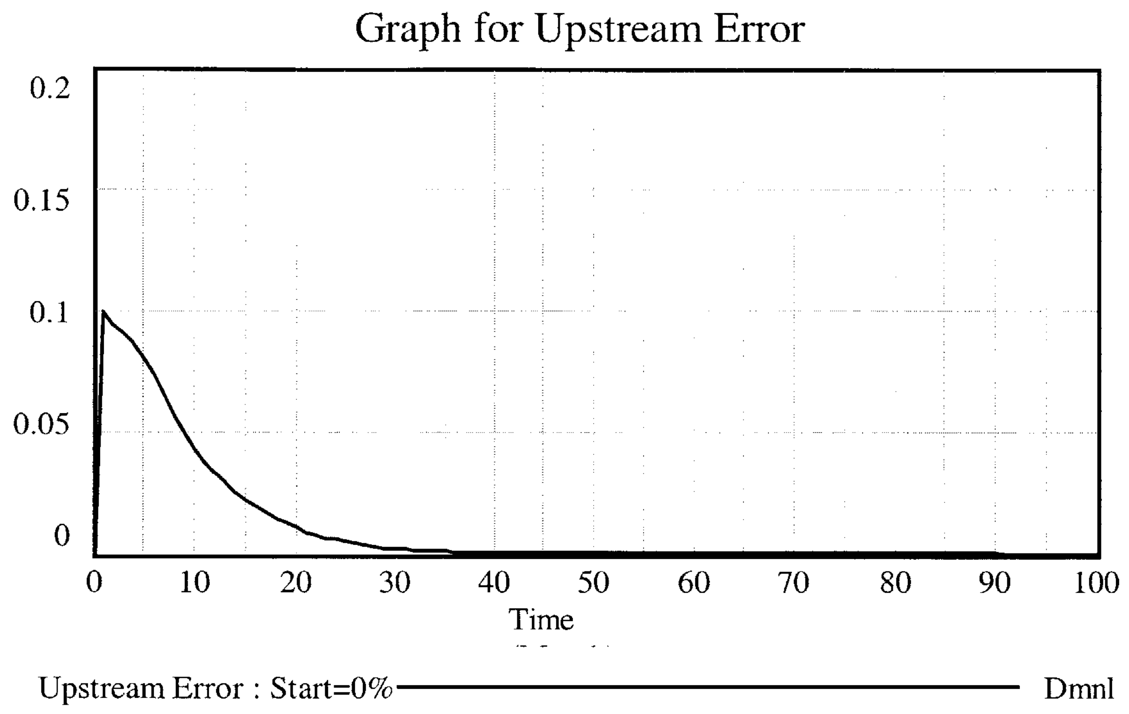


Figure 3-10 Upstream Error in Rework and Overlap Model

Figure 3-10 shows the errors that are generated in the upstream task. There is initially a sharp increase followed by a gradual decrease in errors. The decrease in errors results in the decay caused by an information delay in both detecting and amending the errors.

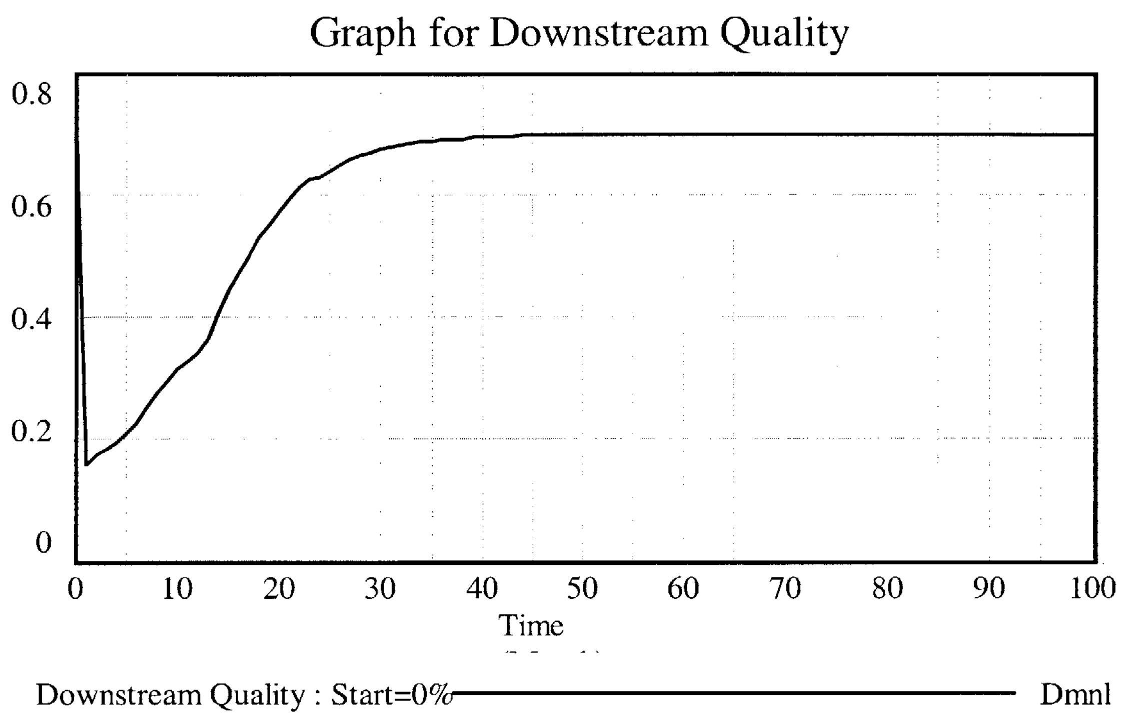


Figure 3-11 Downstream quality in Rework and Overlap Model

Figure 3-11 shows the impact of this upstream error on the downstream quality. As can be seen, there is a sharp decrease in quality corresponding to the sharp increase in errors and the gradual increase in quality corresponding to the gradual decrease in errors.

So, as the quality of the downstream task varies over time, when the downstream process actually starts affect the downstream error generation rate and thus the duration of the downstream task.

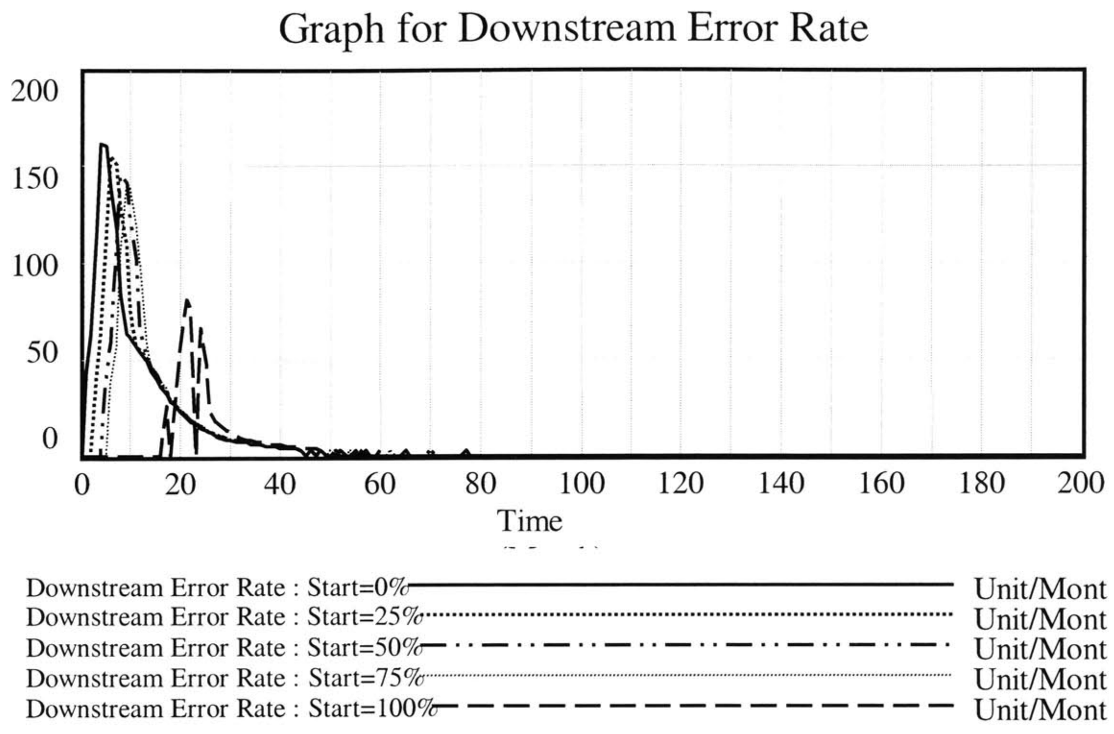


Figure 3-12 Downstream error rate in Rework and Overlap Model

Figure 3-12 shows the error generation rate for different scenarios as to when the downstream task actually starts. Start = x % indicates that the downstream task starts when the upstream task is perceived to be x% complete. Earlier starts indicate higher error generation rates whilst later starts indicated lower error generation rates. The error generation rates generally determine the total duration of the downstream task. So it is conceivable that in certain instances, a later start could result in an overall (upstream and downstream) duration. This is in fact demonstrated in Figure 3-13 and Figure 3-14 where the downstream task completes earlier for a start at 25% completion rather than at 0% or 50% completion of the upstream task.

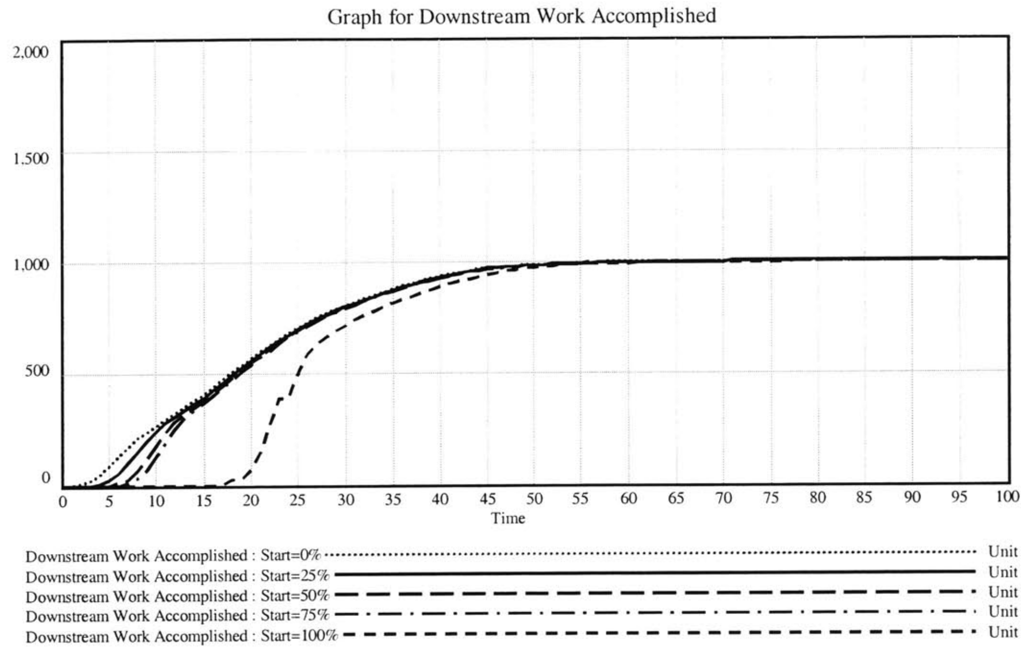


Figure 3-13 Downstream Work Accomplished – Overall View

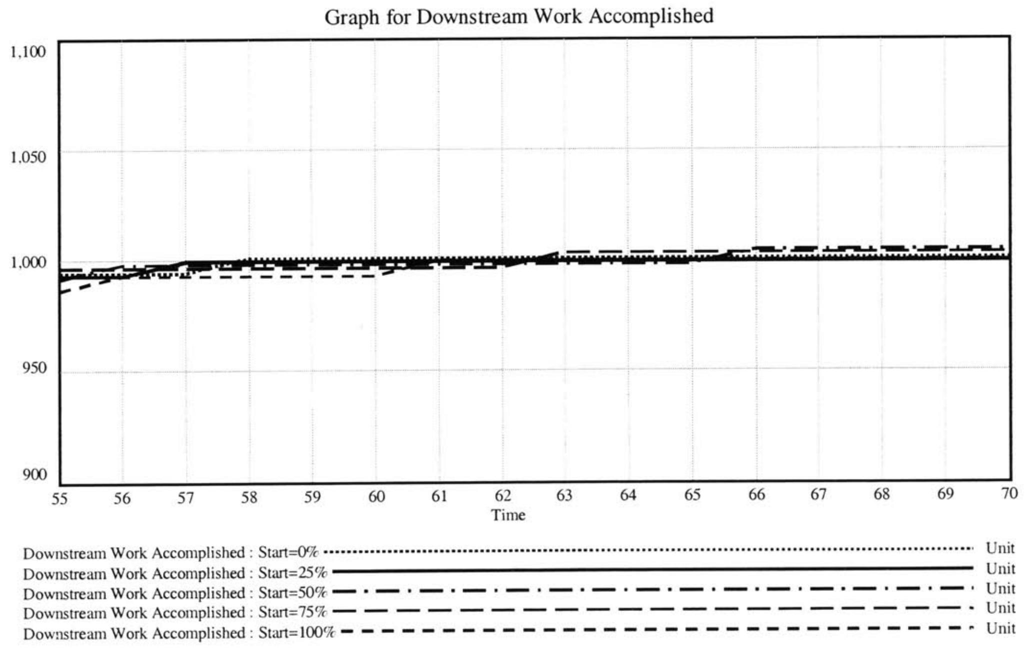


Figure 3-14 Downstream Work Accomplished - Exploded View

Basically, because of the inherent quality of the upstream task, there is a perceived amount of work complete that is less than the actual work complete due to errors. The downstream tasks starts based on those errors and basically cause further errors to the downstream tasks. What actually happens is that errors in the upstream tasks affect the quality of the downstream tasks and cause more rework to be done in the downstream tasks. So the downstream tasks actually take a longer time to execute if executed too early (as the upstream errors are not fixed yet). In some cases, as shown in Figure 3-13 and Figure 3-14, the savings in time by overlapping is insufficient to compensate the additional time it takes to execute the downstream tasks; thus causing the phenomenon described here. One question that begs answering though is under what circumstances this phenomenon exists and whether there is an optimal point as to the amount of overlap should be used (Peña-Mora and Park, 2001a). Peña-Mora and Park (2001b) use a concept called reliability buffering and identify "windows of opportunities" where this phenomenon exists.

3.3 Building the model

3.3.1 Project Structure and the basis for knowledge evolution

Traditional project management models such as CPM and PERT have concentrated on optimizing the project performance by the sequencing of tasks. These tasks are based on a work breakdown structure which is obtained from breaking down complex problems into sub-problems and parts that are easier to address alone than in the context of the whole. This concept of breaking complexity into simpler elements has been extensively used within established theories. The concept is to breakdown a complex system into simpler sub-systems and to breakdown the sub-systems into even simpler sub-sub-systems, resulting in a system hierarchy. This system hierarchy will then consist of components each of which represents a sequence of tasks that need to be performed. This sequence of task thus reflects the system hierarchy quite closely and is known as the work breakdown structure.

However, as described in Chapter 2 above, these models do not adequately model the dynamic complexities of product development projects. Product development processes are highly iterative and some form of dynamic modeling is needed to model the dynamic complexities. Tools such as the Design Structure Matrix (DSM) and Axiomatic Design address some of these dynamic issues but neglect to describe or model the underlying process which drives cycle time. In order to adequately describe the process structure in the form of the causal relationships that generate project behavior, we select system dynamics as a modeling tool. Section 2.5.4 above describes some of the recent research using system dynamics to model the product development process. More notably, Ford and Sterman (1998) has developed a model that ties in four principal performance drivers – process structure, resources, targets and scope. Along with the recent movement in the use of concurrent engineering, product platforms and cross-functional teams to reduce cycle time, one realizes that organizational structure, system architecture and knowledge re-use are also fundamental drivers in development project performance. In the past, where there was little pressure to reduce development cycle time, issues such as organizational structure and system architecture have not been considered levers of project performance. Our literature review, however, shows that they can be quite important drivers and it is the purpose of this research to develop a framework that allows us to study this.

The knowledge evolution framework is a more holistic representation of the project and views the project as processes generating knowledge and information rather than completing predetermined tasks that may change over time.

3.3.2 Knowledge Evolution

Figure 3-15 shows an overview of the knowledge evolution model that was built using the system dynamics methodology. As in most system dynamics model, although the model works together as a whole where small changes in certain parameters anywhere in the model can affect behavior in globally, the model does consist of distinct separate features that are described in the sections that follow. These distinct features include:

- The knowledge repository
- Knowledge processes
- Information prerequisites
- Probabilities
- Reduction factors and learning rates
- Iterations
- Dispersion between knowledge processes

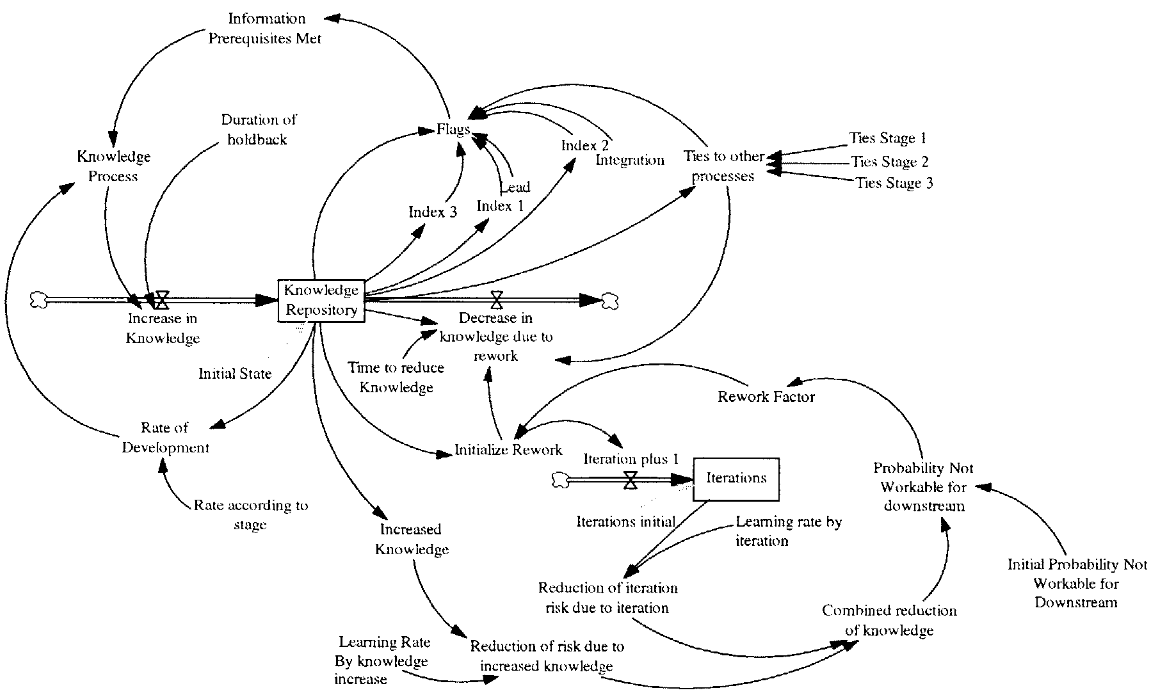


Figure 3-15 Overview of Knowledge Evolution Model

3.3.2.1 The Knowledge Repository and the stages of knowledge

At the heart of the model is the knowledge repository which indicates the state of knowledge with the project at any point in time. Since the knowledge evolution framework intends to map the progress of collective knowledge across a project's organization, it is necessary to somehow measure the state of knowledge of any particular portion of the project.

A means for such a measurement is proposed by Bohn (1994) as discussed in Section 2.2.3 above. Bohn applied various states of "knowing" to the various processes. Similarly, it is possible to apply Bohn's (1994) various stages of knowledge of technical processes to the various parts of the system hierarchy. Considering a typical component of a system hierarchy, that component will have its sub-systems, a parent system (of which the component is a sub-system) and sibling components that the component will need to interface with. That specific component has a knowledge cycle that proceeds through various stages of knowledge are affected by the stage of knowledge of its parent system, sibling systems and sub-systems.

Figure 3-16 shows the phases of product development that a system goes through. It may be said that each of the systems, subsystems, sub-subsystems and components go through a similar process. Phases 4 & 5 provided a basis for the sub-system design and incorporation which means it could be iterative (if proposed subsystems don't work). These are also the phases in which the subsystems and their subsystems or components are specified. The interactions with the sub-system processes are two fold.

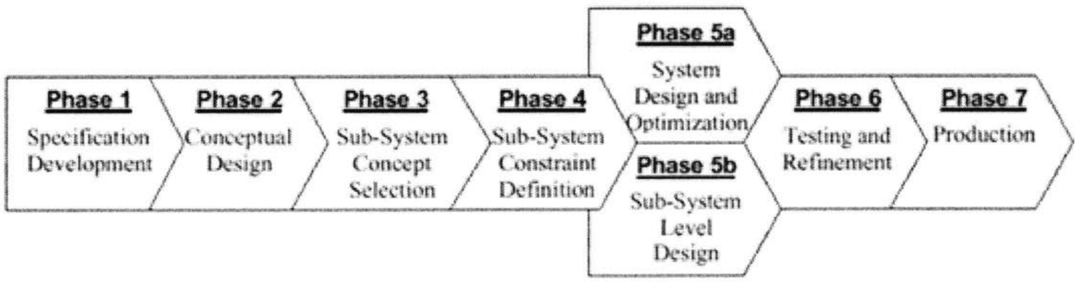


Figure 3-16 Phases of Product Development

There is first a top-down approach to the knowledge evolution of the system as it is broken down into subsystems. There is then a bottom up approach to finalize details of the subsystems so knowledge about the sub-systems are built-up as knowledge of the system itself can then be finalized. This is very similar to Ford's Product Development Process (FPDS) methodology as shown in Figure 3-17 where there is a top-down cascade of targets followed by a bottom-up approach for optimization and verification. (Lavine, 1999).

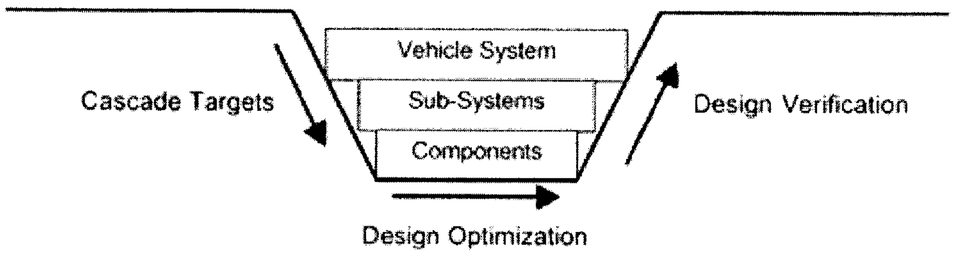


Figure 3-17 Ford's FPDS Systems Engineering Approach
(Source: Lavine, 1999)

Also coined a "V" shaped process, the left part of the "V" represents the initial top-down approach where the system is primarily architected, the right part of the "V" represents the bottom-up approach where details are filled in. Based on this design methodology, we would expect knowledge stages to progress down a hierarchy with higher stages of knowledge on the top of the hierarchy initially. At a certain stage, the progress reverses; the higher stages of knowledge are at the bottom of the hierarchy. Referring to Figure 3-16 above, Phase 5 is both top-down (5b) and bottom-up (5a)

As different components and subcomponents of the hierarchy advances through knowledge evolution at different rates, it becomes necessary to keep track of them separately. The stage of knowledge of individual components are kept track in the knowledge repository.

Just as Bohn's stages of knowledge represented more certainty in the control of the technical processes, advancing to a further stage in knowledge evolution represents more certainty of the information generated. This is discussed in slightly further details under probabilities.

3.3.2.2 Knowledge processes

What causes the collective knowledge of the project to evolve as the knowledge repository advances through the different stages of knowledge are the knowledge processes. Knowledge processes represent the teams and people that act to evolve the collective knowledge. This concept of knowledge processes is similar to the various domains in axiomatic design where knowledge needs to be transferred between the domains in a zig-zagging process.

There are some primary relationships between knowledge processes that affect their activity and hence the rate of evolution. These include the dispersion between the knowledge processes as well as the prerequisite requirements of the knowledge processes. By the very nature of there being distinct knowledge processes indicates that there is segregation of knowledge. Knowledge segregation, in this case, does not simply mean that particular knowledge processes do not have the skill or expertise to make informal decisions. It also includes situations where the processes have not been given the mandate to make those decisions and hence are not aware of the outcome of those decisions.

So one of the key impacts of knowledge segregation is that it leads to risk of having to iterate. Other relationships that affect the activity of knowledge processes and hence the rates of evolution are the dispersion and the prerequisite requirements.

3.3.2.3 Prerequisite requirements of knowledge processes

Understanding prerequisites has always been one of the keys in managing product development projects. The CPM/PERT method involves identifying predecessors to activities building a network of activities but identifying which activity is performed first and in the case of CPM, a critical path is calculated identifying the longest path of critical activities that would determine the duration of the project. The DSM method is sometimes used to identify couplings (through the identification of information of tasks completed) of processes and optimized to produce an optimal sequence of work. Even the system dynamics models on project management involve this concept of work availability as mentioned in Section 2.5.4.2 and shown in Figure 2-17.

In the knowledge evolution framework, these perquisite requirements are information prerequisites. In knowledge evolution, knowledge processes are active only when information prerequisites are met. In other words, further knowledge or information about a component can be generated only when the information prerequisites are met. For example, the foundation of a building cannot be designed unless the loads from of the building are known. Wheelwright and Clark (1992) indicate four modes of upstream and downstream interaction as shown in Figure 3-2. In the knowledge evolution model, as the knowledge repository represents the knowledge already known about the system, the upstream portion is read from the knowledge repository. The downstream portion is, of course, the knowledge process itself. For a system with multiple processes and multiple repositories, this mapping of information prerequisites occur in a matrix such as the one shown in Figure 3-18 below.

	Knowledge Repository				
Knowledge Process					
	X				
	X	X			
	X		X		
		X		X	

Figure 3-18 Information prerequisites

This mapping closely resembles the Design Structure Matrix (DSM) where the "X" denotes communication between the processes. Unlike the DSM, however, "X" represents one of the modes of upstream-downstream interaction in accordance to Table 3-1below.

Value of "X"	Representation
0	Process is independent of other processes.
1	Tightly integrated problem solving where the difference in stage is within a specified amount.
2	Integrated problem solving where both processes must remain in same stage.
3	Early involvement. The upstream processes must be ahead of the downstream process
4	Early start in the dark. Batched Processed, where the upstream process must be a certain amount ahead of the downstream process.
5	Serial/Batch. Where the upstream process must complete its whole stage before the downstream process can proceed.

Table 3-1 Types of information prerequisites

3.3.2.4 Knowledge segregation and probabilities that work requires iteration

The phenomenon of rework and overlap as described in Section 3.2.3 above is of significant importance in the knowledge evolution model as well. Before it is possible to elaborate this, it is useful to identify sources of iteration. Iteration can be largely categorized into four broad categories.

There is firstly iteration that is represented by rework due to errors or failure to meet quality requirements. This is the type of iteration that prior research (Roberts, 1974; Richardson and Pugh, 1981; Ford, 1995; Ford and Sterman, 1999; Li 1999) has been primarily concerned with. Rework of this nature tends to be due to the process itself and can be minimized through better management of the process.

The second category of iteration is caused by a change in scope. This change in scope may be caused by requirements change initiated by the client or due to circumstances as more information becomes available.

The third and fourth category of iteration is due either to insufficient information or sufficient information but not made available to the right persons. Though the symptoms and behavior are very different, they have different implications on remedies.

The differentiation on the causes of iteration is important as they identify the sources of the problem and generally dictate what remedies are available. As rework due to mistakes are caused by the processes internally and any solution will need to address that. Rework due to information misplacement indicates inefficiencies in the information distribution system and solutions such as communication enhancement through organization or technology may be possible. Iteration due to insufficient information and information change, however, indicate implementation strategy issues.

Insufficient information is the most immediate outcome of knowledge segregation and is the focus of this research. The question on whether to start the ball rolling on processes when information is lacking then becomes significant. Like the phenomenon described in the Rework and Overlap model, starting a process earlier does not necessarily imply ending it earlier due to more rework if the process is started earlier.

Iteration due to insufficient information occurs when certain assumptions that are made about parameters are no longer true. In the product development process with segregation of knowledge, there are probabilities that certain assumptions made do not hold true. As the knowledge is segregated, such assumptions also do not get challenged until sufficient information is generated. When these assumptions are generated, work done that was based on these assumptions may no longer be valid. If such is the case, iteration becomes necessary. As will be demonstrated, this type of iteration is inherent by the fact that the design process is performed by multiple participants and hence the information and knowledge is segregated.

Iteration due to information change occurs when upstream processes change their information. This could be due to mistakes or perhaps information changes on processes even further upstream.

Iteration due to information misplacement occurs when the required information exists but is deemed to be unavailable due to inadequacies in the information distribution system. This can exist as it takes time for information to get distributed or some persons may be unintentionally "left out of the loop". Incidentally, if it does exist, this mode of iteration can be minimized through proper knowledge management.

The basis of this research is that due to the segregation of knowledge and information, product development project participants do not have full knowledge to perform their tasks without iteration. That being stated, there is neither certainty nor uncertainty that at any particular stage work will need to be iterated. The relationship is therefore probabilistic.

The probabilities in the spreadsheet input represent the probabilities that a process would need to go through at least one iteration in the process of the cycle. As the system dynamics software uses discrete systems, the probability for each time step needs to be estimated. This is done as follows:

Suppose that P represents the probability of at least one iteration taking place, and n representing the number of time steps. If the probability at each time step is uniform and is represented by p, then

$$P = 1 - (1-p)^n.$$

Therefore,

$$p = 1-(1-P)^{1/n}.$$

In actual cases, p may not be constant. In many cases, there is a higher chance of p occurring at a certain time. In the absence of evidential data, however, a uniform distribution is a relatively accurate first order approximation. If the model were to be refined, and if more information about the timing of iteration probability were available, the model could be modified accordingly.

In deriving the probabilities to be used in a model, several factors need to be considered. The following represents a list of some of these factors:

- *Criticality.* This represents how critical the knowledge is.
- *Sensitivity.* How sensitive the knowledge processes is to changes in information.
- *Location of knowledge.* This has impacts on how the knowledge would be communicated and whether it would be explicitly or tacitly transferred.
- *Dispersion.* How the organization of the firm would facilitate the transfer of required knowledge.

These factors all affect the probabilities and thus would directly impact project duration and performance.

3.3.2.5 Reduction of probabilities

Since the probabilities of iteration are due to knowledge segregation, the probabilities reduce as knowledge and information is available to other participants in the product development process. In other words, the probabilities reduce as more is learnt about the system. It is necessary to differentiate between two types of learning. Firstly, learning by knowledge increase. This is the general type of learning that occurs as more information and knowledge is generated about the system. The second type of learning is the learning by iteration. Even though iteration occurs, there is much we can learn about the system. At the very minimum, the project has learnt that what does not work.

Generally, as more information and knowledge is generated by the project organization, the probability that iteration will happen is reduced. In the knowledge evolution model, useful information and knowledge is indicated by an increase in the stage of knowledge repositories. Possible reduction factors in probabilities relating to this increase is shown in Figure 3-19.

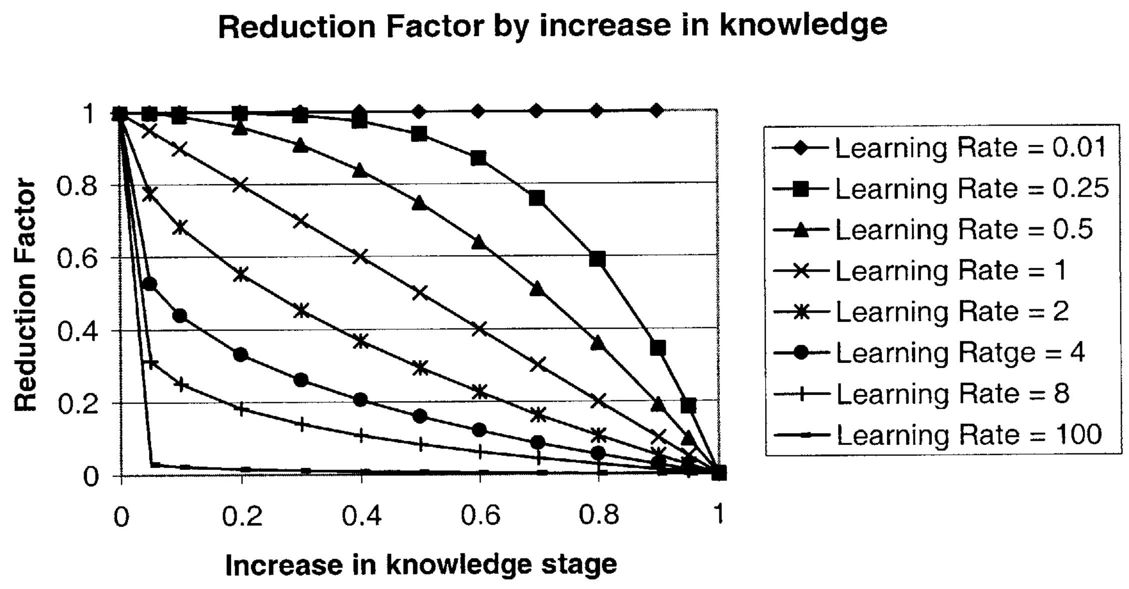


Figure 3-19 Reduction Factor through increase in knowledge

Though the relationship is not precisely known, we know that the space is defined by a reduction factor between 0 and 1 and since each stage of knowledge is unity, the increase in knowledge is between 0 and 1.

The equation

$$\textit{Reduction factor} = 1 - (\Delta \textit{Knowledge})^{1/\textit{LearningRate}}$$

Represents a family of curves as shown in Figure 3-19 that suit this description and represents a good estimate to understanding the relationship.

Once the process goes through iteration, even though the knowledge repository does not advance (it in fact decreases), some knowledge about the system is still generated. At the very least, knowledge of what does not work and why it does not work is now known. So even if the stage of knowledge actually decreases, the knowledge gained in the iteration process reduces the probability of iteration.

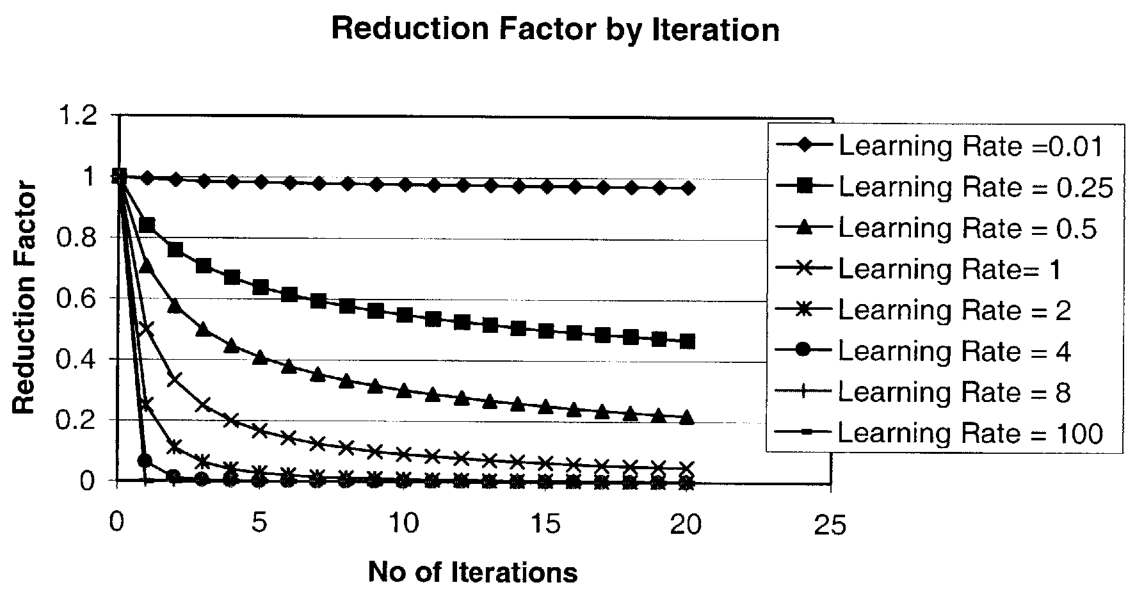


Figure 3-20 Reduction Factor through iterations

The equation

$$\textit{Reduction factor} = 1/(1+ \textit{No of iterations})^{LearningRate}$$

represents the family of curves that is shown in the Figure 3-20 which describes a probable relationship between the number of iterations and the reduction factor.

As there is a difference in the types of learning, there must also be the differentiation in the learning rates. In both types of learning, a higher learning rate represents a quicker reduction of probability of iteration occurring as either knowledge increases or number of iterations increase.

3.3.2.6 Iteration

Iteration in the knowledge evolution model is represented by a reduction in the stages of knowledge for the relevant processes. Some may comment that this is a "loss" in knowledge. This is not the case since knowledge cannot really be lost. Some of the knowledge may be rendered irrelevant but the knowledge and information generated still exists. In fact, as described in the previous sections, that knowledge and information, though not directly relevant is still of value to the product development project as a whole since knowledge of what does not work reduces the probabilities of iteration.

Iteration also does not mean that the process starts over again since much of the knowledge gained and information generated can be reused. In the knowledge evolution framework, some knowledge process triggers off iteration, as the knowledge generated in some previous process is unacceptable. At this point the stage of knowledge is reduced by a certain pre-determined amount for the previous process as well as all the immediate downstream processes. This pre-determined amount varies depending on the process but can be very little (e.g. for tightly integrated processes where there are checks along the way), or very large.

There is a stock for iteration that basically keeps track of how many times a knowledge repository has been iterated.

3.3.2.7 Dispersion between knowledge processes

Keeping in mind that the knowledge creating process includes both the transfer of tacit and explicit knowledge, how effectively that knowledge/information is transferred has an impact on the performance of the project.

As discussed in Section 3.2.1.3 above, the probability of communication amongst team members reduces as the distance between members increase (Allen, 1977). Whilst Allen's study observed the impact of physical distance, there are other forms of "distances" that effectively reduce the transfer of knowledge and information. These "distances" could include geographic, organizational and functional boundaries. A more generic term to "distance" would be dispersion.

For the case of simplicity, we combine all these factors of geography, organization and function into the single term, dispersion. Based on the similarity with Allen's (1977) study, the greater the dispersion between two knowledge processes, the less likely communication will take place. Unlike the R&D environment, however, communication is enforced in product development as it becomes necessary to coordinate. What dispersion is likely to do is to make the communication more formal and less frequent.

For example, in a building project, the communication between an architect that works for an architectural firm and a structural engineer that works for a structural engineering firm would communicate mostly through drawings, perhaps weekly meetings. There is already a functional and organizational dispersion. If the geographic dispersion were to increase (say the firms were in different cities), the communication would rely more on drawings and the frequency of meetings would reduce drastically. This would reduce the opportunity for knowledge to be transferred tacitly as there would be less face-to-face meetings.

It then becomes important to understand characteristics of formal communications such as papers, letters, documentation and informal communications such as brainstorming sessions, face-to-face discussion and phone conversations. As discussed above, the four dimensions of communications affect its quality and effectiveness. These four dimensions are richness, frequency, direction and timing as shown in Figure 3-1. This research assumes the following relationships:

- As dispersion increases, frequency of communication decreases, the communication becomes more formal and less informal.
- Formal modes of communications occur at a slight lag and occur slower than informal modes of communications. However, formal modes of communications are more accurate (as more thought had been put in) and less prone to mistakes.

Formal modes of communication also occur in larger batches compared to informal modes of communications.

Though dispersion affects multiple variables in the model, a first order approximation by varying the holdback duration is used in this research.

3.4 Model Description Summary

The knowledge evolution framework represents a model of the product development project via its information and knowledge evolution over time as opposed to the traditional models of tasks sequencing and progress over time.

Basically, stages of knowledge in what is termed the knowledge repository represent how much knowledge is generated by the individual knowledge processes. Knowledge processes represent individuals or teams that are part of the product development project organization. Multiple knowledge processes and repositories can be modeled in the knowledge evolution framework. These knowledge processes are constrained (much like the predecessor relationship in CPM) by other knowledge processes. There are 5 types of constraints ranging from independence to tightly constrained (integrated mode). These constraints, termed information prerequisites are entered as a matrix which closely resembles the Design Structure Matrix.

Each of the knowledge processes may be required to go through an iteration processes. This will result in some loss in the stage of knowledge. The possibility of this iteration process happening is probabilistic according some pre-set probability and generally reducing as the stage of knowledge or number of iterations increase.

Chapter 4 The Delta Design Game

In order to demonstrate the usefulness of the knowledge evolution framework, the model is used to simulate the design project of a Deltan Habitat as outlined by the Delta Design Game (Bucciarelli, 1991). The Delta Design game was created as an abstraction of the engineering design process intending to demonstrate designing as a social process of negotiation amongst participants. Though this research is not purely concerned with the social interactions amongst participants, the design of a Deltan habitat does provide a useful design problem on a simplified level (the game is designed to be complete within 3 to 4 hours), yet provides a relatively accurate picture of a product development project. The Deltan habitat project was chosen in this research precisely because it is an abstraction of the design process and limits the study to the issues at hand without the complications of large projects. The simple and short project with multiple design roles provides the stability that large and long projects cannot as there changes in project organization structure to inevitably happen during these projects. Other complications such as changing requirements are also absent in the Deltan habitat project since the scope and specifications are clearly spelt out without the risk of client change.

Although the abstractness of the task disconnects it from the real world (the design is for a habitat on the fictional deltoid plane), the open form of the design problem provides a very real experience where there is no single right answer. And although the process will allow participants to reach a conclusion within a couple of hours, the specifications and constraints have all the elements of a real project.

This chapter describes the use of the knowledge evolution framework on the Deltan design process. Firstly, the rules of the game and some brief design specifications are described. The model with customized parameters is then used to simulate the process. The results of these simulations are then compared with six separate projects that were undertaken by first time participants.

4.1 The Delta Design Game

The Delta Design game is a team exercise intended to demonstrate the social interactions amongst the participants of engineering. Participants learn that designing in teams is as much a social process as it is a technical process and that excellence in design requires attention to both. The design task is an abstract one of designing a habitat on an imaginary deltoid plane where not only the units of measure are different but the basic fundamentals of mechanics are vastly different as well.

The game is played in teams of four participants each performing different roles in the design of a Deltan habitat. The games consist of 4 phases, the first taking about an hour where participants are each given a different set of materials where they understand the role they play in the design process as well as the deep functional knowledge that the roles need to perform their function.

This is then followed by another hour reserved solely for the design. Which includes time both for coming up with a suitable situation as well as tabulating the performance measures in preparation for a 5 minute presentation.

The third phase is the presentation of the various groups and the final phase is an open discussion.

The 4 roles in the game are the architect, the project manager, the structural engineer and the thermal engineer. The architect provides the layout design ensuring the habitat satisfies certain criteria such as entrance location, overall color coordination and aesthetics. The project manager makes sure the cost is within budget as well as within schedule. The structural engineer specifies key support locations as well as checking to ensure the segments are structurally sufficient. The thermal engineer calculates the stable temperature of the habitat as well as ensuring that the temperature of each delta is within an acceptable range.

The following information is provided more to understand the information that needs to be generated and is required from the various participants as well as constraints that the various participants face.

4.1.1 Representation of the design

Representation of the design consists of an arrangement of blue and red equilateral triangles placed on a grid of “diamonds”. Each triangle represent a delta which are the building blocks of the habitat. A sample of the design representation is shown in Figure 4-1.

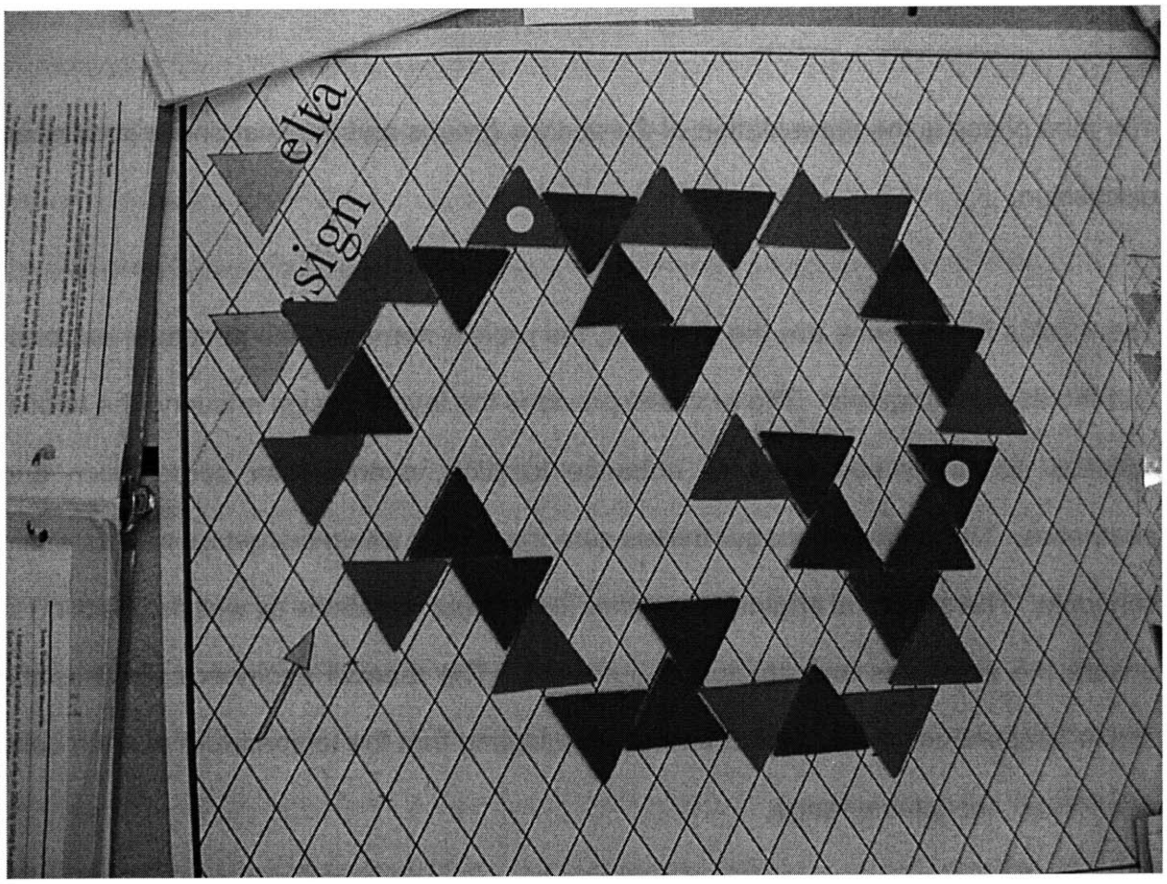


Figure 4-1 Sample design

4.1.2 The task

4.1.2.1 Definition of terms

Before the design task can be properly specified, it is first necessary to define some terms. This is necessary since in order to moot any discrepancy in technical experience, the project exists on an imaginary deltoid plane where mechanics are slightly different from the way we know it to be.

Table 4-1 shows the key measurements and the units used in Delta design:

Measurement	Unit of Measurement	Symbol
Time	Wex	Wx
Distance	Lyn	Ln
Area	Quarter-Delta	Qd
Heat	Deltan Thermal Unit	DTU
Temperature	Degrees Nin	^{o}Nn
Force	Din	Dn
Moment	Lyn-Din	LD
Currency	Zwig	!

Table 4-1 Measurements used in Delta Design

Although Deltan space appears to us as two-dimensional, it does have three distinct directions that occur 60° to each other. The grids mentioned in the design representation, above, are 1 lyn by 1 lyn. Deltas, the fundamental building blocks come in the form of equilateral triangles with sides that measure 2 lyns each. Each delta consists of 4 quarter-deltas as shown in Figure 4-2below.

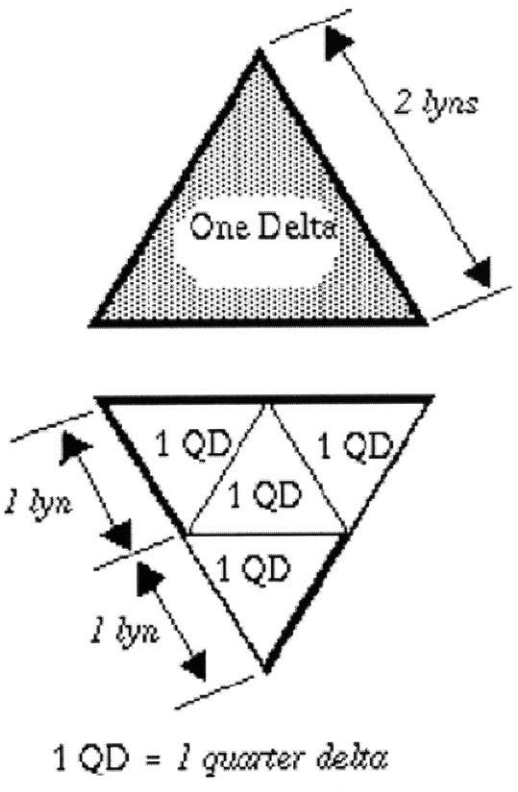


Figure 4-2 Of Deltas, lyns and quarter-deltas

Deltas basically come in two different types of colors, red and blue. Each type has different thermal properties as well as different costs. Further, the cost of joining the deltas together is different depending on whether the joint is red-red, red-blue or blue-blue. Red deltas produce heat, blue deltas do not. There are further properties about deltas which are spelt out under the specific roles.

4.1.2.2 Specifications

The design of the habitat is to satisfy the following conditions:

Minimum Area	100qd
Maximum Cool Deltas (% Total)	60-70%
Average Internal Temperature Range	55-65 ^{0}Nn
Individual Delta Temperature Range	20-85 ^{0}Nn
Maximum Load at Anchor Points	20 Dn
Maximum Internal Moment	40LD for 2lyn joint 20LD for 1lyn joint
Overhead Factor -K	(varies)
Total Budget	!1,400.00
Aesthetics	subjective

Table 4-2 Summary of Design Specifications

In order to understand the above design specifications, it is also necessary to understand the details some of the details that are outlined in the various roles participants play. As the Delta Design game was designed specifically to demonstrate the social interactions between participants in design, participants will find quickly that their design proposal will impact others. For example, the architectural requires no more than 60 to 70% of blue deltas as a color requirement but this provides a constraint to the thermal engineers who have temperature requirements both for the habitat as well as individual deltas.

4.1.3 The roles

The architect

The architect's role is spelt out in the "Architect Primer". Firstly there is the issue filling out an enclosure to satisfy the minimal area requirement. There is then the issue of blues versus reds. Several parameters include the percentage of blue deltas to total (not more than 60-70%) and the dispersion of the blue deltas (minimize the number of joints with two blues). Finally there is the issue of aesthetics. This is a "soft" requirement as it is in vogue to have smooth externals and angular interiors. The performance measure is therefore to maximize interior length to exterior length.

The project manager

The project manager's responsibility is to ensure that cost is kept within the budget and that the construction time is kept to a minimum. The levers that affect the cost include the choice of colors for deltas and how they are laid out.

The structural engineer

The structural engineer's responsibility is to ensure that the habitat will hold together under the prescribed loading conditions. The habitat will require 2 and only 2 anchors which can each only support 20 din. The strength of the individual connections between the deltas also need to be designed for. These depend on the contact length of the joint and is 20 LD/lyn of contact.

The thermal engineer

The thermal engineer's responsibility is to ensure the temperature of the interior as well as the individual deltas is within the specified temperature range.

The above responsibilities with the respective measurements and specifications are summarized in Table 4-3 below.

Role	Measure	Specification
Architect	Internal Area	> 100 qd
	Blueness	<60-70% blue deltas
	Blue dispersion	Maximize %tage of non-blue joints
	Aesthetics	Maximize internal length to external length
Project Manager	Cost	< ! 1,400 Total Cost = K (delta cost + cement cost + module cost)
	Time to build	Minimize Time to build
Structural Engineer	Anchor reactions	Each reaction < 20 dins
	Internal Moments	Moment < Strength of joint where strength of joint = 20LD per lyn of contact
	Gravity Waves	Possibility of gravity changing directions. Up to designer how much to account for.
Thermal Engineer	Delta Temperatures	20 – 85 ^{o}Nn
	Interior Temperature	55 – 65 ^{o}Nn

Table 4-3 Summary of Design Responsibilities for Different Roles

4.2 Relevance of the Delta Design Game

In order to test the use of the knowledge evolution framework and its efficiency in testing policies for true existing projects, it is first necessary to test the robustness of the concepts. Data coming from complex projects, however, is often "tainted" and not objective. Firstly, product development projects are often very long projects, which means data needs to be tracked over long periods of time. To make things more difficult, very often, organizational changes do occur over the course of the project. This leads to inconsistencies in the way data is recorded over the course of the project. In the case of this research, as organizational structure is itself a crucial variable, changes in project organization makes it difficult to make meaningful observations. Further, true projects are not bound as the scope and goals may shift over time.

Based on these shortcomings of using complex projects for testing the knowledge evolution framework, a short-term development project that has a fixed scope and captures the essence of multiple individuals working towards that goal would seem ideal. Though not a true product development project, the Delta Design Game satisfies most of these requirements. By nature of its intended use (to demonstrate social interaction between participants), there are many interdependencies on each of the participants for information. Yet as the design process is simple enough, it provides a boundary that keeps observations and analysis simple so that the crucial issues can be effectively captured.

Finally, complex product development projects are typically one-off. This makes it difficult to compare results across projects as there may be basic differences in the projects that result in different project performances. As described in previous chapters, whether the design is modular or integrated and how the project organization is structured are factors that contribute to changes in project development. The Deltan habitat design project, because of its simplicity and short time required to execute, makes it easy to repeat the process and to compare results across multiple projects.

4.3 The design process for a Deltan habitat

4.3.1 Intended Design Process (The ideal case)

With most project planning tools, iteration is not explicitly planned for. Very frequently, it is either planned for as a review activity with a guesstimated duration or built into the actual activities themselves with some "extra" time for iteration. These estimates for "extra" time are derived from estimates of the ideal case. So, similar to other project planning, an "ideal case" estimate is required.

Although there is no preset procedure for the design of a Deltan habitat, the roles and limitations in the system architecture lead to some preset paths for the design process. This is due particularly as the knowledge base and the information available to a particular role is limited. As discussed above, the design parameters that each role has control over also affects other design parameters. Even though ideal, there is probably no one unique methodology.

One such possible anticipated process can be described as follows:

"Before the project manager, thermal engineer or structural engineer can contribute in any fashion, there must be some form of base design that is made available for them to work on. The architect, therefore, has the initial role in providing the base design. What this base design actually consists of would depend on the "experience" of the team. An experienced team would provide minimal information so to provide for the other participants for feedback before providing further details. This is done to primarily minimize rework due to subsequent tasks. Once this minimal base design is provided to the various other participants, the project manager performs an initial check, the structural engineer performs his initial design and the thermal engineer performs his role. Once the various participants "OK-ed" the base design, the architect would need specifics from the thermal engineer on suggested red and blue assignments before checking the dispersion requirements". In the meantime, the various other participants could go ahead with their designs. Once the thermal engineer makes his design decisions, the architect then finalizes his adjustments and informs the other participants. All participants now start to optimize their design inputs."

4.3.2 More realistic scenarios

The above process described is ideal. In reality, however, as designers do not have full knowledge of what is acceptable or not outside their domain, designs are not guaranteed to be implementable the first time round.

The following description is typical of a process of a fairly experienced team:

"Initially, the architect provided a layout of deltas satisfying the internal area requirement as well as the aesthetics requirement. Feeling that the thermal engineer would have much to say about the choice of colors for the deltas and not too particular about specific locations of red or blue deltas, the architect did not specify the colors of the individual deltas but rather only mentioned that of the 20 deltas used, a maximum of 13 could be blue.

Once the basic layout was given, the project manager, structural engineer and thermal engineers could get to work. The project manager could not pin point the exact cost but could perform conservative estimates as to project cost. It seemed as if there was a good chance that the design could be built within budget so re-design on the architect's part was not necessary. The structural engineer could provide a design for the location of anchors and confirm that the joints between deltas all had sufficient contact points. In this particular case, there were a couple of joints that were not sufficient and the structural engineer had two options. One was to change the location of the anchors, the other was to increase the contact length of those particular joints. The structural engineer brought this up to the architect who seemed willing to change the contact length.

The change was implemented and the other roles (the project manager and thermal engineer) were informed of the change. This, of course, affected some of the calculations that had been made up to that point in time based on the architect's original layout. The project manager did not really mind, since the estimate was still relatively preliminary, but the thermal engineer had more specific rework since he had started specifying which deltas were red and which were blue. Based on temperature requirements and the layout, the thermal engineer was able to estimate that he needed 10.3 to 12.2 red deltas. This would clearly satisfy the blueness requirements that the architect was concerned about but the thermal engineer felt that, based on past experiences there would be a problem keeping delta temperatures below the melt-down range. He recommended one of the alternate routes, to add deltas to the habitat therefore increasing the percentage of blue deltas, or reduce the contact lengths of the joints. In any case, since it was just a hunch, he could do the calculations and confirm whether it was really an issue. The other roles decided to wait for the calculations to check. Seeing that either of the alternatives would mean rework for all the roles, all involved decided that the thermal engineer confirmed it would pose a problem before deciding to make any changes.

The thermal engineer performed the check only to find his suspicion confirmed. Having performed the necessary calculations, he proposed to add a few deltas which would increase the loads on the system (for the structural engineers to check) as well as the cost of the habitat (for the project managers to check). After making the additions, the design went back to the structural engineers for re-design & check, the project manager for a more detailed costing and the architect to ensure the blueness and dispersion criteria were satisfied. There were a few more minor tweaks in the design meant to optimize certain parameters before the design was finalized."

Even with an experienced team, iteration is necessary as participants are not fully equipped with the tools necessary to perform other roles. The experience gained, however, has caused participants to be cautious and aware of previously made mistakes. This is captured in the knowledge evolution framework as lower probability of iteration and resulting in a lower number of cycles. First time participants would yield a higher number of iterations.

4.4 Project Execution

As part of this research, six projects of first time participants were executed concurrently. In order to demonstrate certain features of the knowledge evolution framework, 3 teams (Teams 1, 2 and 3) were asked to follow the game as per instructions. The other 3 teams (Teams 4, 5 and 6) were forced to work with the slight handicap of having to work asynchronously. The diagrams labeled Figure 4-3 through Figure 4-8 show the results of the six projects.

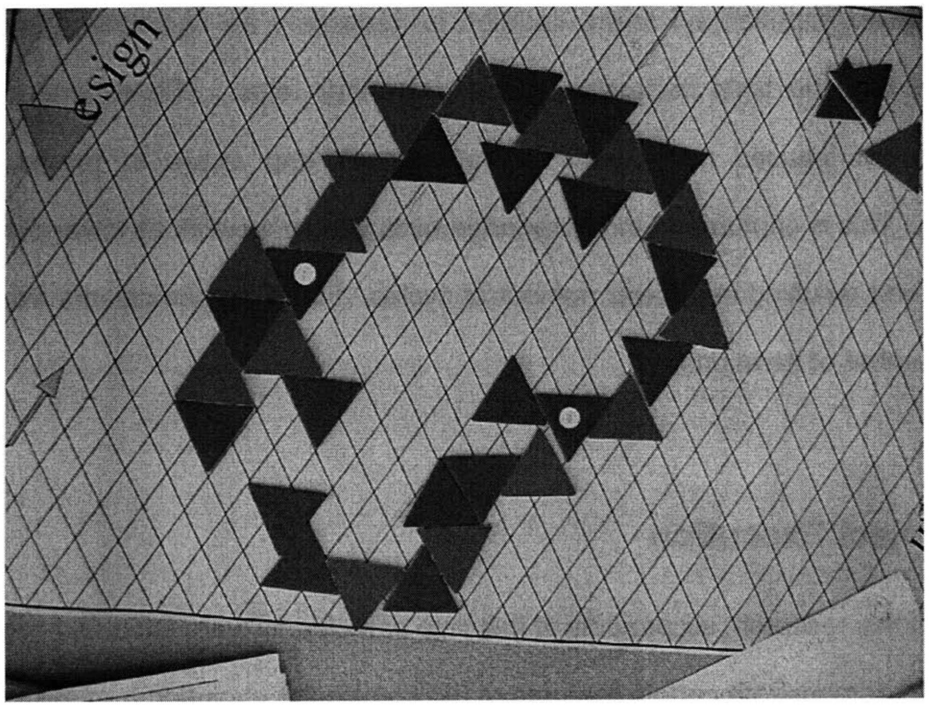


Figure 4-3 Delta Design by Team 1

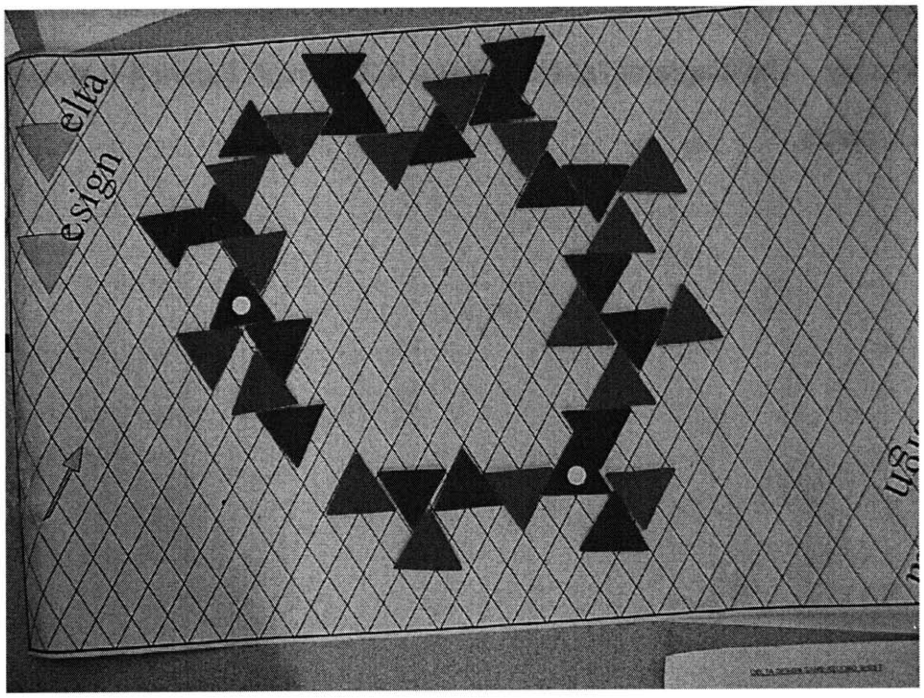


Figure 4-4 Delta Design by Team 2

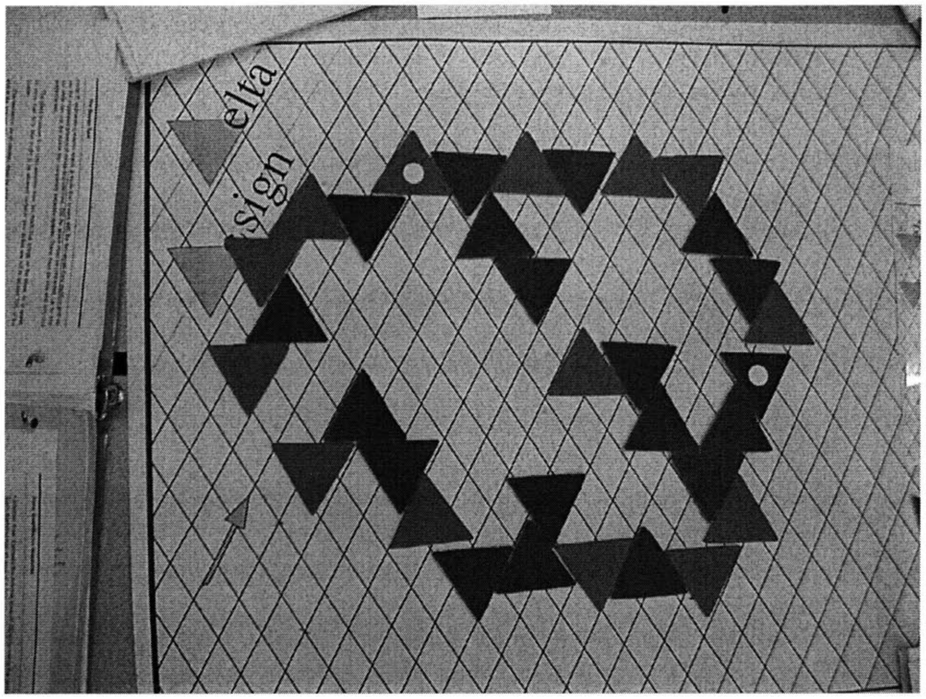


Figure 4-5 Delta Design by Team 3

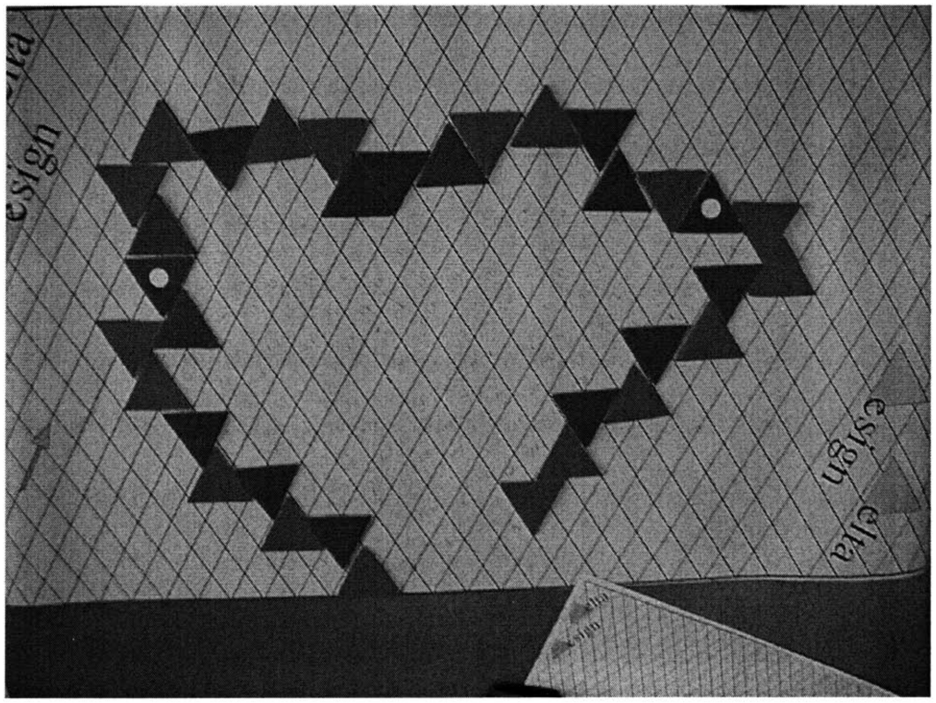


Figure 4-6 Delta Design by Team 4

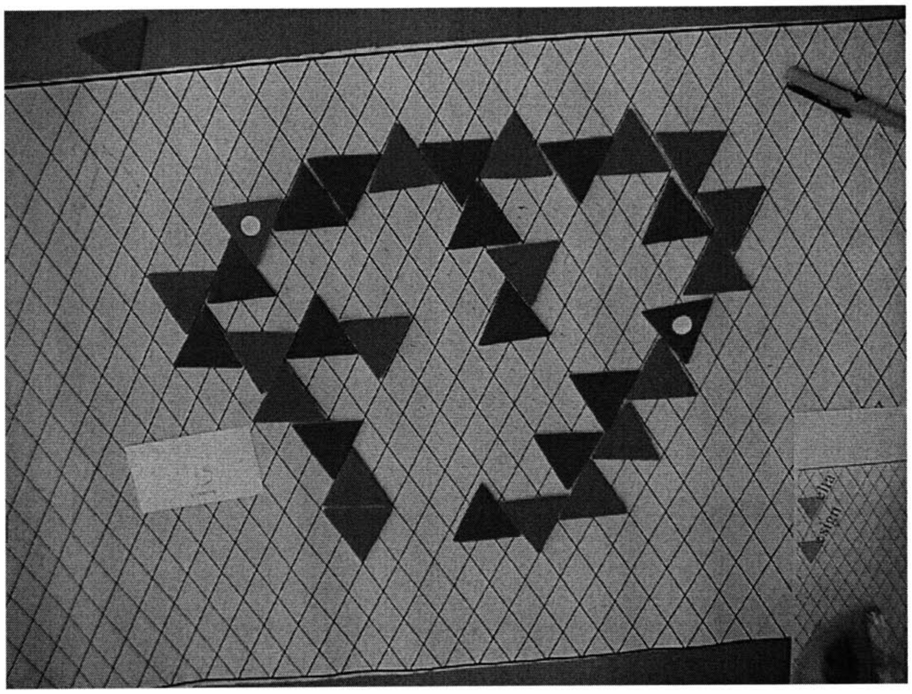


Figure 4-7 Delta Design by Team 5

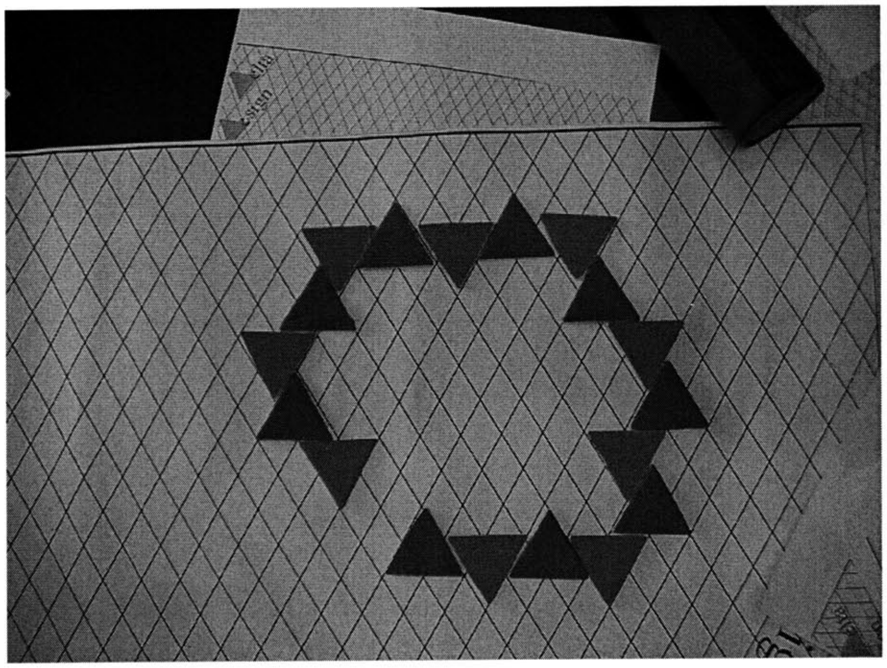


Figure 4-8 Delta Design by Team 6

	Synchronous Mode			Asynchronous Mode		
	Team 1	Team 2	Team 3	Team 4	Team 5	Team 6
Project Duration	70 mins	70 mins	70 mins	65 mins	65 mins	65 mins
Total No of Iterations	2	5	6	3	4	2

Table 4-4 Results of the Delta Design Projects

Table 4-4 summarizes the results of the six projects describing the total duration of the project and the number of iterations the process goes through. The results also tabulate when the iterations happened and which participant had contributed to the iteration.

4.5 Observations from the design process

Monitoring the above processes, several interesting observations that affected the duration of the design process were made. The first of these was the need for iteration. Prior research has modeled iteration as rework that is resulting from mistakes and lack of quality control. From the delta design process above, it can be seen that iteration was unavoidable even though no mistakes were made. Based on the information that was available to individuals, they came up with workable solutions (or commented on unworkable ones) that were limited to their domain. As they did not have access to information in others' domains, it seemed that iteration was unavoidable. Upon closer observation, one realized that the fact that the system architecture and design organization were coupled also had a significant role. This meant that a particular designer's design changed drastically the parameters of others' designs. In some cases, these effects would be severe enough such that other participant's are not able to satisfy the design constraints. This would result in the design already performed to be iterated. If this were not the case, the need for iteration is greatly reduced.

The second interesting observation was the conservatism of design that participants used. Participants tried to give themselves some slack or "safety margin" when they made design decisions or recommendations. To the individual designer, this made quite a bit of sense. This "safety margin" would allow them the flexibility of slight changes from other designers without actually having to perform rework. On the global scale, however, this probably provides more constraint on the other designers and hence probably causing more rework to occur. This dilemma is similar to the one described by Goldratt (Goldratt, 1997) where he recommends that the risk of schedule overrun be managed collectively rather than individual processes in the manufacturing chain build in their own slack time. At the same time, the reverse (where some participants were demanding iteration without evidence that it was needed) was also true. In a particular project, the project manager did not like the using too many modules and demanded an iteration to occur. This was done even though he had no idea at this stage how much the habitat would cost.

The third interesting observation was that even though certain participants could progress they did not. These participants had no lack of information from other participants but were unwilling to proceed as they understood that other participants could cause significant rework on their part and they wanted an understanding before they wasted efforts in further development. Now during this exercise, that seemed like a waste of resources (since idle time is wasted time anyways). It would make sense therefore to proceed with the design since there is a chance of there being no rework. In reality, however, when participants are typically assigned to more than one project, perhaps even more than one role, the participants would rarely proceed with the next stage of design knowing there would be a good chance of rework. They would typically get some form of sign-off on their preliminary work before heading to the next stage of development.

The fourth interesting observation was the similarity in performance between the synchronous mode and the asynchronous mode. Even with significant handicap in communications, the teams that had to communicate asynchronously did not suffer from significant performance loss. Performance in this case being measured by the duration of the project. With the exception of team 1, teams working in a synchronous mode had more iterations than teams without.

Based on these observations, and desirous to improve the design process, we actually identify some issues where it is unclear what the proper improvements are to be made. Issues such as whether safety margins should be managed at individual levels or globally, whether a cross-functional team makes sense and even whether to wait for feedback or develop concepts further are some examples. Projects managers today focus on improving the productivity of individual participants as well as interpersonal communication through the use of innovative methods such as information technology. Although there are efforts promoting the use of cross-functional teams, results have been fairly mixed. This research hopes that understanding the knowledge evolution of the project may provide some insight to these ambiguities.

4.6 Knowledge evolution of the project

Figure 4-9 shows the knowledge evolution model for the Delta Design process. For the purposes of the delta design project, it becomes necessary to calibrate certain parameters in the model to suit the specific project.

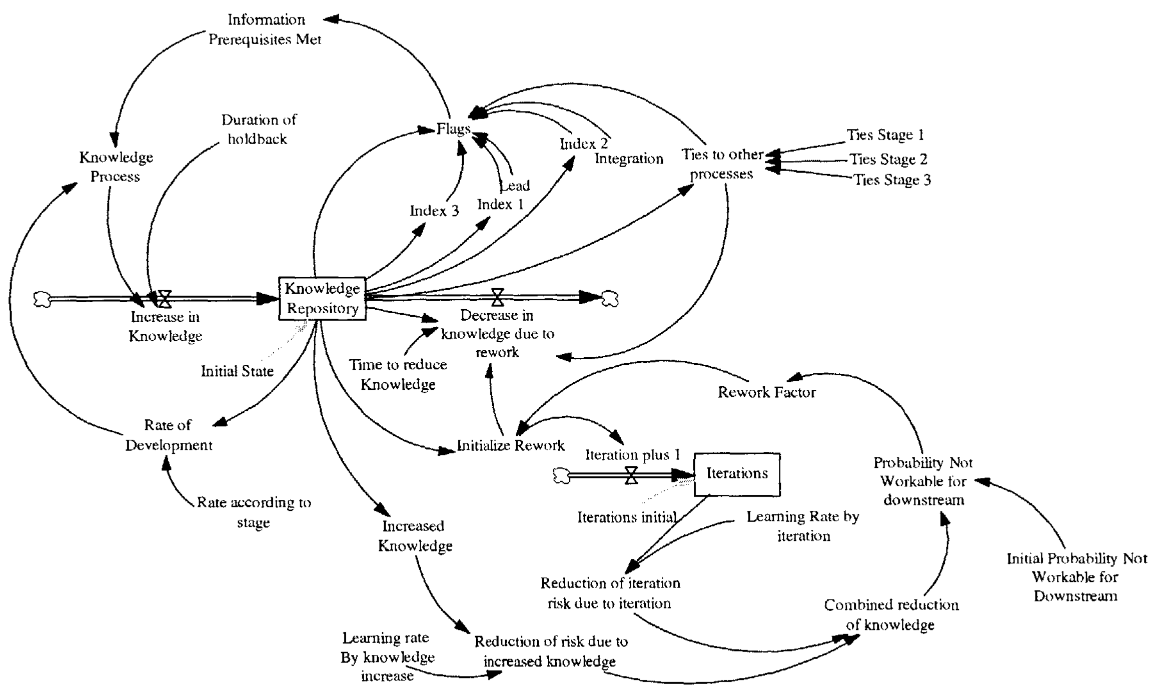


Figure 4-9 Knowledge Evolution of the Delta Design Process

In particular, the following are parameters that need to be defined and calibrated:

- Knowledge Processes and Knowledge Repository
- Rates of development
- Ties to other processes
- Probabilities not workable for downstream
- Holdback period

These parameters are described in more detailed below and the equations used for the model can be found in Appendix 3.

4.6.1 Knowledge Processes and Knowledge Repositories

Firstly, it will be necessary to define what the knowledge processes and repositories are. The knowledge processes are relatively straightforward. Each of the roles represents one knowledge process as shown in Table 2-1.

Knowledge Processes
Architect
Project Manager
Structural Engineer
Thermal Engineer

Table 4-5 Knowledge Process for Delta Design

The system architecture is relatively simple. As the entire habitat is built from combinations of red and blue deltas, the system does not consist of any subsystems. It is an integrated system with red and blue components. The system is integrated as there are numerous possible interfaces between the components to the system and that they perform multiple tasks including thermal control, structural integrity and enclosure. It is also possible to view the system architecture as comprising these different subsystems i.e. thermal, structural and enclosure but with these subsystems tightly integrated or coupled with each other. This architecture is shown in Figure 4-10. For the purpose of mapping out the knowledge evolution framework, this seems more suitable as there is a one-to-one mapping with the knowledge processes.

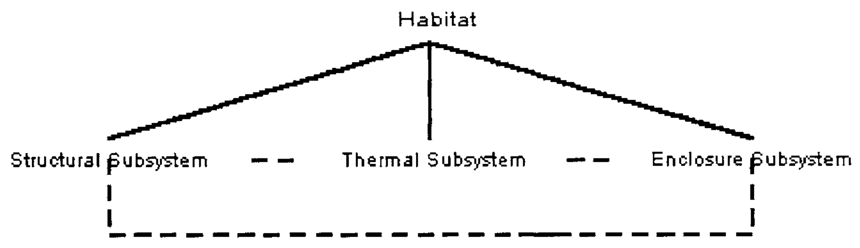


Figure 4-10 System Architecture of Habitat

Once the knowledge repositories are set, it is useful to identify the various stages of knowledge. These stages are selected based on the milestones where information is transferred between the knowledge processes. As construction does not fall under the scope of this project, not the full range of knowledge is used. For simplicity, we represent the stages in need as stages 1 thru 3. Table 4-6 shows the representation of the various stages.

	1	2	3
Architect	Initial layout without finalizing colors	Finalizing color suggestions	Minor changes to optimize
Project Manager	Initial cost estimate	Detailed costing	Minor changes to optimize
Structural Engineer	Structural check	Detailed structural calculations	Minor changes to optimize
Thermal Engineer	Thermal check with color suggestions	Detailed thermal calculations	Minor changes to optimize

Table 4-6 Various Stages of Knowledge for Repositories

4.6.2 Rates of Development

Based on projections on the amount of work, the rates of developments are estimated as follows:

	Stage 1	Stage 2	Stage 3
Architect	0.1	0.2	0.2
Project Manager	0.2	0.1	0.2
Structural Engineer	0.08	0.2	0.2
Thermal Engineer	0.07	0.2	0.2

Table 4-7 Rates of Development for the Knowledge Processes

The next stage is to understand the information pre-requisites for the individual processes and build a matrix of information prerequisites as well as identify the type of pre-requisite.

4.6.3 Information Prerequisites

Similar to placing predecessor activities in the CPM method, information pre-requisites are placed by analyzing which pieces of information are necessary for subsequent stages to proceed. Based on the Delta Design process, the project manager, structural engineer and thermal engineer are unable to proceed with any design until the Architect has completed stage 1. The relationship between the Project Manager, Structural Engineer and Thermal Engineer with the Architect as shown in Table 3-1 is 5. Now, the Architect cannot move to the second stage of knowledge development until each of the other processes has achieved Stage 1. This represents each of them having performed initial checks on the proposed layout by the architect. The relationship is therefore relatively tightly integrated (2 according to Table 3-1).

Table 4-8 shown below summarizes the information prerequisites for the design process.

		Upstream (Stage 1)			
		Architect	Project Manager	Structural Engineer	Thermal Engineer
Downstream	Architect	0	2	2	2
	Project Manager	5	0	0	0
	Structural Engineer	5	0	0	0
	Thermal Engineer	5	0	0	0
		Stage 2			
		Architect	Project Manager	Structural Engineer	Thermal Engineer
Downstream	Architect	0	0	0	2
	Project Manager	0	0	0	0
	Structural Engineer	0	0	0	0
	Thermal Engineer	5	0	0	0
		Stage 3			
		Architect	Project Manager	Structural Engineer	Thermal Engineer
Downstream	Architect	0	0	0	2
	Project Manager	0	0	0	0
	Structural Engineer	0	0	0	0
	Thermal Engineer	5	0	0	0

Table 4-8 Table of Information Prerequisites

4.6.4 Probabilities upstream is not workable for downstream

Table 4-9 represents the probabilities that knowledge already developed needs to be change based on restrictions by subsequent processes. As whether the knowledge already developed needs to be reestablished depends on what was developed, one can imagine that the variable is probabilistic.

The values for probabilities used in our initial evaluation were determined by trial and error to represent what seemed like realistic figures from comparison of behavior of the model to what actually occurred.

		Upstream			
		Architect	Project Manager	Structural Engineer	Thermal Engineer
Downstream	Architect	0	0	0	0
	Project Manager	0.8	0	0	0
	Structural Engineer	0.8	0	0	0
	Thermal Engineer	0.8	0	0	0

Table 4-9 Table of probabilities not workable for downstream

Table 4-9 suggests that the Architect's proposed schematic has an initial chance of 80% needing rework at some point in the development of the project.

4.6.5 Holdback period

The holdback period represents a time where even though all the information prerequisites have been met knowledge processes remain inactive. Such times of lull activities can be either intentional (as in the case of our observations where participants are unwilling to proceed) or unintentional (as in the case of asynchronous activities). We use a holdback duration of 5 minutes to simulate the asynchronous teams.

4.7 Model behavior

4.7.1 Without iteration

Figure 4-11 shows the behavior of the knowledge evolution model when iteration is not considered in the model.

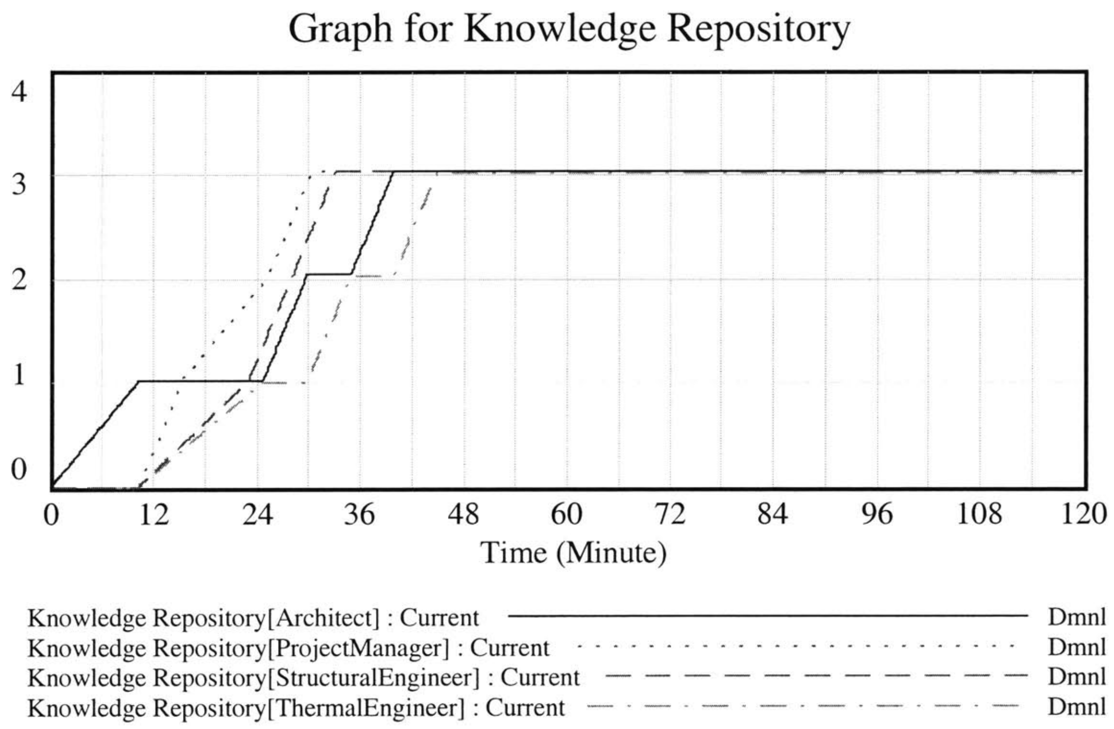


Figure 4-11 Knowledge Evolution of Deltan Habitat (without iteration)

4.7.2 With iteration

Figure 4-12 and Figure 4-13 show the behavior of the knowledge evolution model when probabilities for iteration are considered.

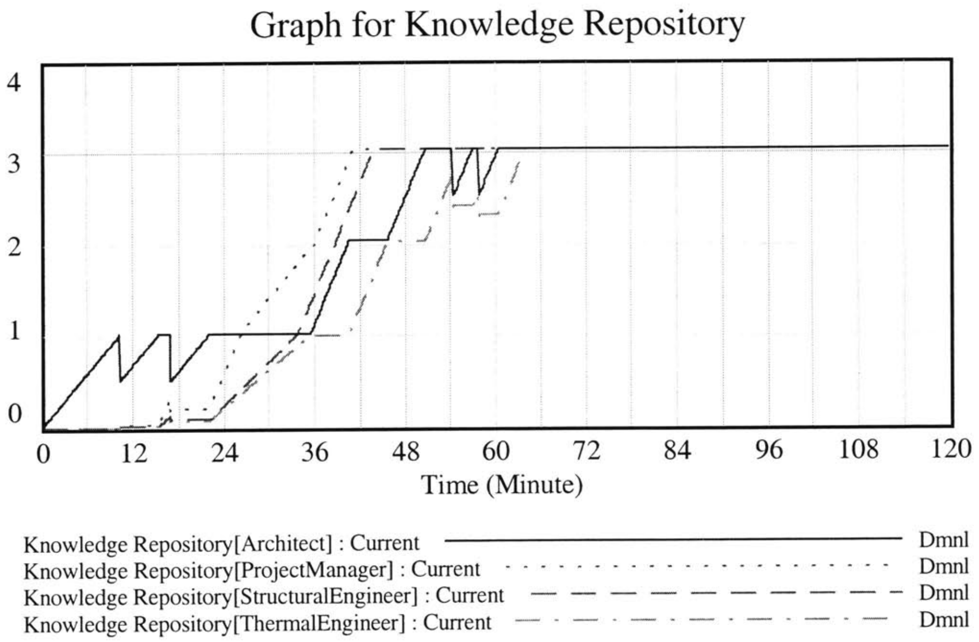


Figure 4-12 Knowledge Evolution of Deltan Habitat (with iteration) - synchronous

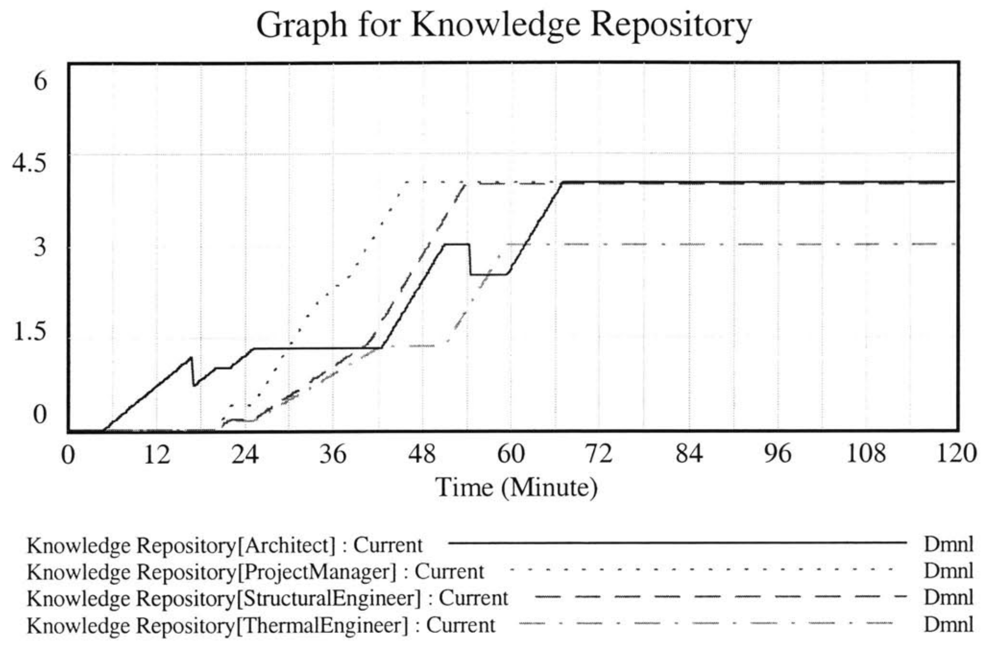


Figure 4-13 Knowledge Evolution of Deltan Habitat (with iteration) – asynchronous

Figure 4-12 shows the case with synchronous communication whilst Figure 4-13 shows the case with asynchronous communication.

4.8 Comparisons

The model simulation shows similarity to the results as tabulated in Section 4.5 above. In particular, the differences between the asynchronous teams and the synchronous teams with the calibration of the duration of holdback period variable. It is noted that the synchronous teams tend to go through more iterations that the asynchronous ones so even though the asynchronous teams were working with a significant handicap, the project performance was relatively similar. This concept of using the holdback period to simulate project organization structure.

4.9 Summary

The Delta design game represents a project with all the components of a typical project including multiple participants, multiple skill sets and thus segregated knowledge, a well-defined scope and definite project schedule. Yet, the project is simple enough without the complications of boundary conditions including the project scope and project organization changing. These features of the Delta Design project make it ideal for analyzing the segregation of knowledge in project organizations.

The knowledge evolution framework described in the preceding chapters are calibrated to suit the Deltan Habitat projects. The knowledge evolution simulations closely resembled the results from actual projects run by first time participants.

Six projects were run concurrently with 3 teams working synchronously and 3 teams working asynchronously. The project performances were not found to be significantly different though the synchronous teams went through more iterations than the asynchronous teams. These attributes were also captured by the knowledge evolution framework.

Chapter 5 Analysis & Observations

5.1 Introduction

The knowledge evolution framework can be used to provide insight into project complexities and assist us in implementing policies to improve product development project performances.

The previous chapter demonstrated that the knowledge evolution framework is capable of modeling and simulating projects as the results produced by the simulation model resembles the results of the six projects actually executed. This chapter evaluates the particular model developed for the Deltan habitat design project to gather further insights as to how specific variables affect the performance of product development projects. These variables include the following:

- Probabilities
- Reduction factors and learning rates
- Reduction in knowledge repository
- Duration of holdback period

These insights have implications on how specific strategies and policies intended to improve project performance will actually perform. Implementations of concurrent engineering and cross-functional teams that have not produced consistent performances as mentioned in previous chapters are examples of policies the framework could be used to avoid.

5.2 Sensitivity Analysis

5.2.1 Probabilities

Whilst iterations are affected by a few key factors including the number of iterations and the stage of knowledge within the knowledge processes, the probability that processes need to be iterated is one of the key factors responsible for iteration. Quite simply, high probabilities imply more iterations while lower probabilities imply less iterations. This trend is indeed confirmed by the simulations that are tabulated in Appendix 4-1.

One interesting observation that seems counter intuitive, seems to be the impact of multiple probabilities. Comparing the results between processes where only one process can be a cause for iteration and processes where multiple processes can be causes for iteration with equal probabilities, it seems that there is no significant increase in the number of iterations. For example, comparing the cases where Prob = [0,0,0.9,0] and Prob = [0,0.9,0.9,0.9] we see no significant rise in the number of iterations as illustrated in Figure 5-1 and Figure 5-2 respectively.

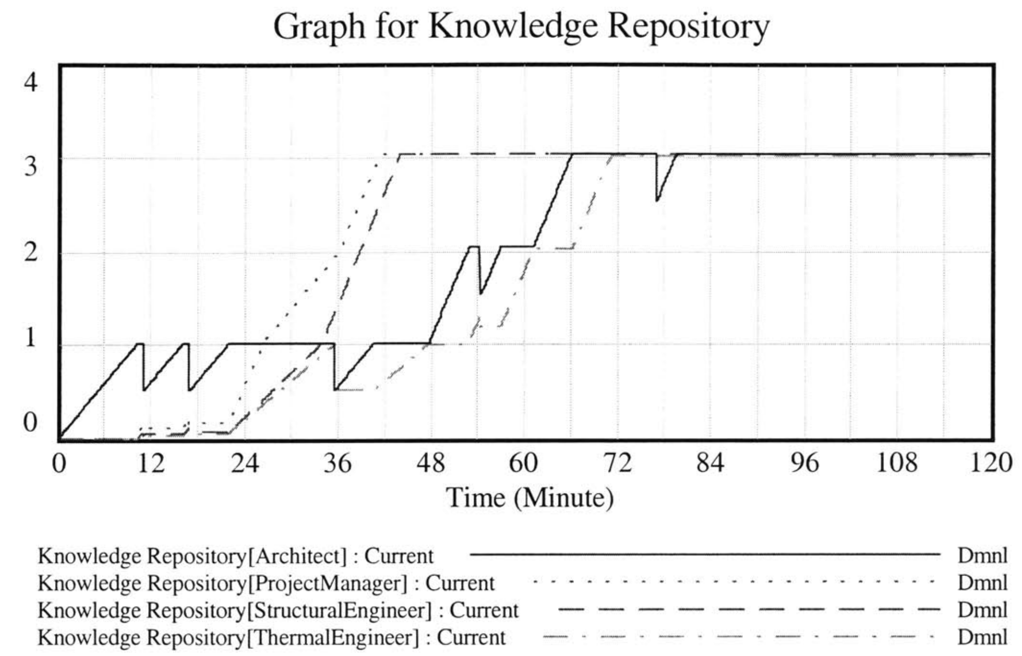


Figure 5-1 Knowledge Evolution (Prob = [0,0,0.9,0]])

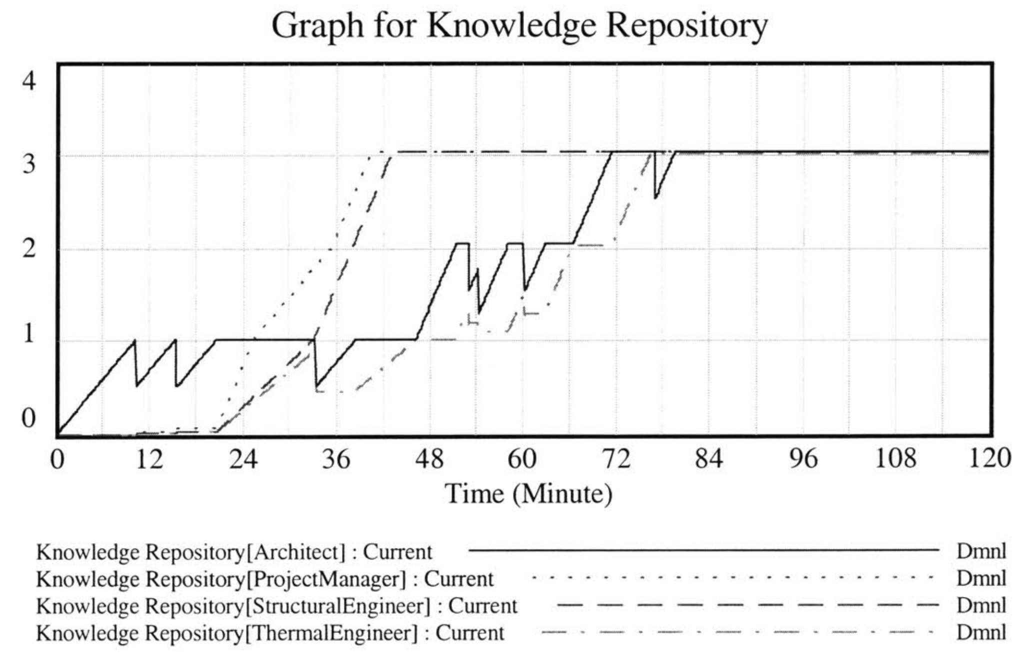


Figure 5-2 Knowledge Evolution (Prob = [0,0.9,0.9,0.9])

Intuition will tell us that the additional risk [0, 0.9, 0, 0.9] to the original [0, 0, 0.9, 0] will cause more iterations. With further analysis of the results, we find that though this is not untrue, the additional impact is minimal. If we compare with the other simulations for Prob = [0,0.9,0,0] and Prob = [0,0,0,0.9] we will find that these are less dominant (see Appendix 4-1). The probabilities of iteration, therefore seem to conform with the saying "that a chain is only as strong as its weakest link". This seems to be due to the following reasons:

- The iterations caused by the original [0, 0, 0.9, 0] actually dominate the process.
- Even though the probability from the various processes are the same, this is definitely possible since the reduction factors for iteration reduce probabilities after iteration and further suppresses the risk for iteration from the other processes. The impact from the reduction factors is evaluated in the following section.

5.2.2 Reduction factors and learning rates

Another key factor affecting iterations in the simulation models is the reduction factor for the probabilities. As discussed in 3.3.2.5 , there are two separate types for reduction factors; one of which is for the number of iterations and the other for increases in the stage of knowledge. In general, as the learning rate increases, the reduction factor reduces more quickly as shown in the family of curves in Figure 3-19 and Figure 3-20. The direct implication of this is, of course, that the quicker the reduction of reduction factor, the lower the iteration. This is more or less confirmed by the results that are tabulated in Appendix 4-2.

As mentioned in the previous section, the dominance of certain probabilities over others is verified as well. These results are also tabulated in Appendix 4-2. The behavior that the probability for one process dominates over the others is more significant for higher learning rates. Figure 5-3 shows the differences between Prob = [0,0.9,0.9,0.9] and Prob = [0,0,0.9,0]. For high learning rates, the evolution is more or less identical whilst for low learning rates, the case for Prob = [0,0,0.9,0] has significantly more iterations.

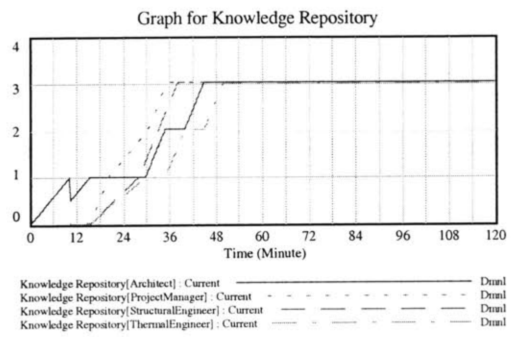
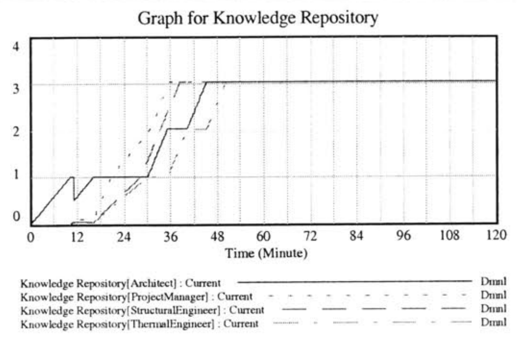
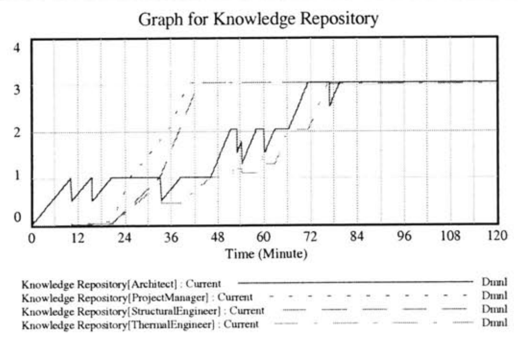
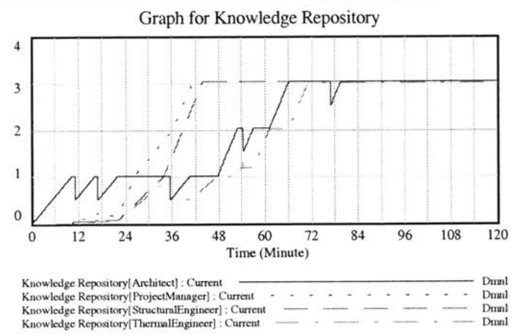
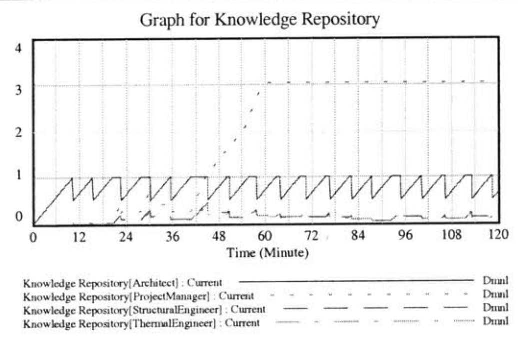
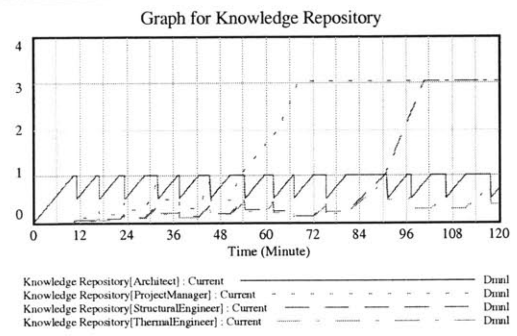
	Prob = [0,0.9,0.9,0.9]	**Prob = [0,0,0.9,0]**
High Learning Rate		
Medium Learning Rate		
Low Learning Rate		

Figure 5-3 Probability dominance and Learning Rates

Though not intuitive, this behavior is logical as the dominance of one process over another is determined by its effect to quickly bring down the reduction factor which causes lower probabilities for the other processes.

5.2.3 Reduction in knowledge repository

Under the knowledge evolution framework, iteration is represented by a reduction in the knowledge repository. This does not mean that knowledge is lost, but simply that the work it took to generate the knowledge needs to be redone. Obviously, the immediate question is how much reduction in the stage actually takes place. Also obvious, is that the more reduction per iteration, the longer the project duration.

Therefore, in the ideal situation, the lower the reduction per iteration, the better. If it were possible to minimize reduction in the knowledge repository However, low reductions in the knowledge repository typically implies a more integrated process with frequent checks and thus more iterations. This implies that there is a tradeoff between more iterations with lower reductions in the knowledge repository but more iterations and less iterations but greater reductions in the knowledge repository. Any policies that is intended to increase project performance must either be targeted at lowering the reduction per iteration without increasing the frequency of iterations or take this tradeoff into consideration.

5.2.4 Duration of holdback period

The duration of holdback period denotes the period of time a knowledge process after its information prerequisites are met and before it starts productive work towards the project. Whilst the existence of these periods may be either unintentional or intended as part of formal policies they do exist in projects.

In a previous chapter, the calibration between synchronous teams and asynchronous teams were performed by varying the holdback period. The figures suggest that asynchronous teams operate with some holdback. This holdback may not be intentional, rather, it occurs as a consequence of asynchronous teams information assimilation process and hearing from other parties before proceeding with work. This was, in fact, one of the observations from the Deltan habitat project. Even though certain participants had the necessary pre-requisite information, they preferred not to perform work until further information was available.

Intuition will tell us that the impact of holdback period is that its effect on the project duration will be negative only if the work to be performed is on the critical path. This is true in the absence of other factors. In fact, intuition has also told participants of the Delta habitat project to holdback in the fear of rework and iteration as we had observed. From a cost and resource usage angle, holdback therefore does make some sense. Our model, however, also shows us that with holdback (such as with asynchronous teams), there can be improvements in project duration. This is illustrated in Figure 5-4 and Figure 5-5.

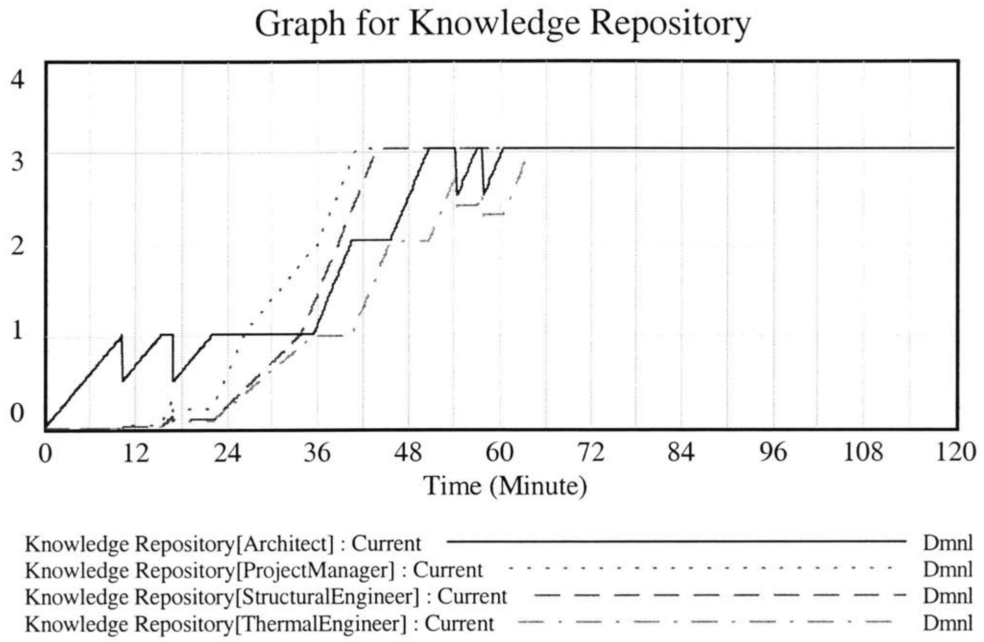


Figure 5-4 Knowledge Evolution without holdback

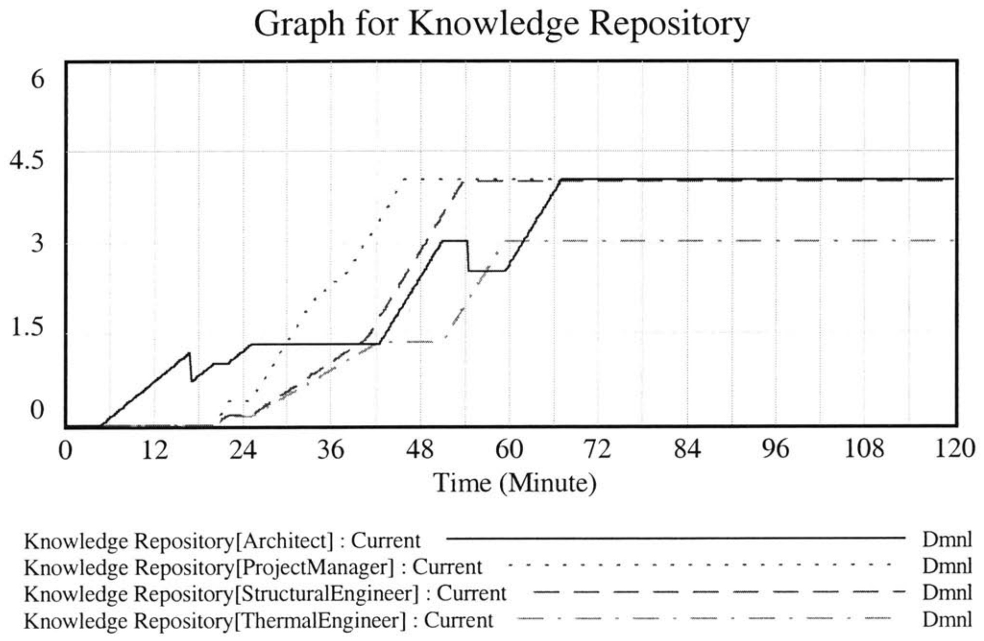


Figure 5-5 Knowledge Evolution with holdback

This counter-intuitive behavior is in fact caused by a lower number of iterations. With a certain amount of holdback, the reduction in probabilities lowers the probabilities sufficiently in the duration of the holdback. This lowering of probability reduces the risk of iterations and hence the number of iterations.

Through various simulations, some key levers have been identified that triggers this type of behavior. Firstly, it is the relative learning rates between knowledge increase and iterations rather than the absolute learning rate that is key. The larger the ratio, the more likely holdback will result positively towards project performance. This is due to the fact that if the learning rate by iteration is large as well, iteration will lower the probabilities of iteration just as much. Secondly, the reduction in stage of knowledge compared to the duration of holdback period with the rates of development is significant as well. This finding extends Peña-Mora and Park's (2001b) model on the reliability buffer that is concerned with primarily work quality identifying further causes for existence of a reliability buffer.

5.3 Insights

5.3.1 Policies for improving project performance

Based on the results from analyzing variations in key variables, in order to reduce project duration, the following generally hold true:

- Reducing probabilities that work is unacceptable as far as possible,

- Increasing the learning rates (both by iteration and by knowledge increase),
- Minimize the reduction in knowledge repository without increasing the frequency of iteration, and
- In the right settings, strategically holding back whilst waiting for important but not critical information.

There are, however, more important lessons to be drawn from the analysis. Firstly, for reducing probabilities, it was observed that the probability from one process could dominate the entire knowledge evolution. This process would therefore become the "bottleneck" in the entire evolution. Efforts to minimize probabilities should therefore be targeted at that particular process. In an effort to minimize this effect of the weakest link, not just the probabilities need to be reviewed but also other key variables such as reduction factors and reduction in stage.

Secondly, it is important to understand the learning rates for several reasons. The higher the learning rates, the more this behavior of dominance applies. Therefore, for scenarios with very low learning rates, applying this principle of dealing with the weakest link will see less dramatic improvements. Also, the ratio of learning rate by knowledge increase to learning rate by iterations is important since scenarios with higher ratios will see more improvements with holdback than otherwise.

Fourthly, it is generally not straightforward to reduce the reduction in stage of knowledge per iteration without increasing the number of iterations. Decreasing the reduction in stage of knowledge generally implies closer integration, more checks and therefore higher risks of iteration. As the reduction in stage of knowledge is typically limited to the stage, one means around this is to increase the number of stages. In the physical world, this means introducing more stable sub-forms and "locking in" the design at critical points. So similar to theories involving evolution and stable states (Simon, 1969; Jacobsen, 1995) more sub-forms decreases the time required (also see Simon's watchmaker analogy, Simon, 1969). One must note, however, that the mechanics of this artificially creates hold back scenarios which, again, may or may not be desirous in product development projects depending on the project circumstances.

5.3.2 Performance of concurrent engineering and cross-functional teams

In previous chapters we have stated that although there have been successful implementations of concurrent engineering and cross-functional teams, the aggregated results have been fairly mixed (Iansiti, 1993; Clark and Wheelwright, 1993; Dean and Susman, 1991). Researchers have attributed this contradiction to increased complexities and tightened constraints imposed by the interdependencies requiring higher levels of coordination (Ulrich and Eppinger, 1994; Malone and Crowston, 1990). This section attempts to identify further reasons for this contradiction.

The concept in concurrent engineering is to concurrently execute parallel activities and processes. Ultimately, once work becomes available, it should be executed. What is not accounted for, however, is the iterative process. Firstly, there is the iteration due to rework as discussed in the "Rework and overlap" model. Starting earlier, generally implies a lower quality and increases the amount or work that needs to be iterated. This leads to less than satisfactory results. This, rework, however, may be minimized through better quality controls and tighter scope controls. Besides iteration due to rework, this research has shown that iteration exists in the absence of poor quality. This iteration is due to lack of knowledge and information from certain participants and knowledge segregation. Like rework and quality, the later a knowledge process starts, the more information and knowledge is available and the risk of iteration reduced. Starting too early may imply starting with a high probability of iteration and leading to more iterations and probably longer durations.

Of course, this does not mean that concurrent engineering strategies do not work, they have proven to be successful in many instances. What needs to be evaluated prior to embarking on concurrent engineering is this risk of more iterations leading to longer durations. The results of this research therefore suggest that candidate projects suitable for concurrent engineering are projects with the following attributes:

- Strong organizational processes to handle increased complexity as well as ensure quality is maintained.
- Lower probabilities of iteration due to knowledge segregated, in other words, less coupled knowledge processes.
- In the event processes are coupled, the probabilities should be low. High learning rates is a plus but not necessary.
- Low reduction in knowledge repository per iteration is important as there will likely be more iterations in this type of environment.

- Low Learning rate by knowledge increase/Learning rate by iteration ratio (cross-functional teams imply less holdback periods)

Policies involving cross-functional teams, unlike concurrent engineering take on a different approach. The project organization is arranged such that participants with different functional background operate with each other closely (both in terms of physical distance and dispersion in general). This basically reduces the time and effort in communication between the functions. In the knowledge evolution framework, this can be represented by lowering the reduction in knowledge repository. However, as mentioned above, this typically implies more iteration. So we see immediately, that it is an issue of trade-off and deciding which alternative is more suitable for the case of the project.

Candidates for projects that are suitable for cross-functional teams are projects that have the following attributes:

- Low probability of iteration
- High learning rates
- Low learning rate by knowledge increase/learning rate by iteration ratio (cross-functional teams also imply less holdback periods)

In these analysis, the holdback period seems relatively crucial and is used to simulate the relative dispersion of the project organization. It seems that firms have inherent holdback periods which can be described as the period of time between when a process can theoretically proceed to when it can practically proceed given that it takes time for information to be communicated through meetings, drawings and so on. Both concurrent engineering and cross-functional teams are intended to reduce that time. However, as found in this research, such reduction of holdback period does not necessarily reduce the project duration.

5.4 Summary

This chapter analyzes how variations to the various key variables affect the project duration. Whilst much of the behavior is intuitive, some interesting results seemed counter-intuitive. The following results were rather intuitive:

- Project duration is shortened with the following trends:-
  - Lower probabilities of iteration
  - Higher learning rates
  - Lower reduction in knowledge repository per iteration
  - Minimum holdback period

These results tend to hold true except in special circumstances. These special circumstances lead to results that are counter-intuitive such as:

- Probabilities of iteration from a process can be dominant over other processes provided the learning rate by iteration is sufficiently high.
- A longer holdback period can shorten project duration provided the learning rate by knowledge increase is sufficiently high and the learning rate by iteration is sufficiently low.

- There exists a tradeoff between lowering the reduction in knowledge repository per iteration and the number of iterations that needs to be carefully considered.

These results have implications on the implementation of policies or strategies to shorten the project duration. The knowledge evolution framework is used to describe circumstances that would cause shortening or lengthening of project durations. In particular, the use of concurrent engineering and cross-functional teams have been analyzed using the framework and an explanation why these methodologies may or may not work have been provided.

Chapter 6 Conclusions

6.1 Summary

In response to commercial pressures to shorten the product development life cycle, many product development teams in different industries have implemented new techniques such as concurrent engineering, cross-functional teams and product platforms. Though there have been success stories, the aggregate results have been fairly mixed. This contradiction to the intended results seems to suggest there may be some missing ingredient.

It is a sad fact that almost all projects come in late and over budget. With rules of thumb for estimation of project duration as "take your best guess and multiply it by two", why do projects still come in late? Some researchers consider iteration as the major cause. Iteration, for example, explains the 90% syndrome. A common scenario in complex projects where 90% of the work gets executed pretty much in accordance to the schedule and the remaining 10% of the work takes more than the original 90% to complete. Past researchers have looked into this problem and have considered iteration to be caused mostly by rework due to mistakes, or failure to meet quality problems.

Many design texts, however, start off by stating that design, which is part of the product development process, is an iterative process. It is quite apparent that these texts are not referring to mistakes and errors.

This research has looked at the inherent nature of the iteration and has identified that knowledge and information segregation lead to rework. In order to understand the implications of this segregation, this research proposes a knowledge evolution framework, a framework that captures the product development project as an evolutionary one where the collective knowledge of the project contained in the minds of the participants, in the documents generated and even in physical artifacts such as prototypes, parts and even products increases over time. This concept is borrowed from practitioners in knowledge management who have considered physical artifacts to be embodiments of knowledge.

The knowledge evolution framework lies in the intersection of various fields including knowledge management, product development, project management tools and organizational theory. Relevant theory and literature from these fields were analyzed and concepts required in the knowledge evolution framework were expended upon. Much of the relevant literature lies in the use of system dynamics in studying the dynamic nature of projects. This research is no different as the knowledge evolution framework was developed using the system dynamics methodology.

The knowledge evolution framework was developed as a system dynamics model representing the knowledge as stages of knowledge in a knowledge repository and representing the people and teams by knowledge processes. Each knowledge process has probabilities that it will need to iterate as a result of downstream processes and has information prerequisites that need to be met before the process can proceed.

In order to test this framework, simple projects were executed involving the design of habitats on the fictitious Deltoid plane. The project is a design exercise used as an instruction tool for the demonstration that design is indeed a social process. These projects were simple enough to execute that the research ran six concurrent projects. The results were quite similar to that predicted by the simulation model. One interesting result that arose from the six projects was that even though three projects operated in an asynchronous mode at a considerable handicap, the project performance was comparable. This was due to the fact that the asynchronous projects had, on average, less number of iterations.

The knowledge evolution framework model for the Deltan habitat design project was also used to analyze the effect of some of the key variables on project performance. In particular, the project duration. While it was found that generally project durations shorten when probabilities are low, learning rates are high and the reduction of knowledge repository per iteration is low, optimizing these factors is not sufficient. The analysis also yielded some interesting results.

One such result was that probabilities of iteration from a process can be dominant over other processes provided learning rate by iteration is sufficiently high. The implication of this result is that any policies targeting reduction of probabilities must be targeted at "the weakest link". Especially if learning rate by iteration is high.

Another interesting result was that longer holdback periods can potentially shorten project durations. This is provided the learning rate by knowledge increase is sufficiently high and the learning rate by iteration is sufficiently low. This result partially answers the question as to why initiatives such as concurrent engineering and cross-functional teams yield mixed results. Both these initiatives act to shorten the holdback periods reducing either the time between processes or dispersion in the project organization. While these are commonly regarded to shorten project duration, they can also lengthen project durations if the environmental factors are correct. This is caused by the fact that earlier actions result in higher probabilities of iteration.

The final interesting result was the understanding of the tradeoff between increased iteration and reduction in knowledge repository. Many policies that act to improve on side also balances the other thereby canceling the benefits. This understanding allows us to understand why certain policies work and others do not.

6.2 Contributions

One of the purposes of this research was to provide an alternative to the traditional task-based frameworks and tools that did not accurately capture the concept of iteration in the absence of errors or lack of quality control. The knowledge evolution paradigm is a new approach that models the iterative process as a probabilistic one that brings out this side of product development projects.

Application of the knowledge evolution framework to projects have also demonstrated the existence of dominant probabilities and environmental factors that make certain probabilities dominant. Also, the framework has demonstrated that “holding back” does work in certain instances and has found that they are more likely to work when the learning rate by knowledge increase is high and the learning rate by iteration is low. These relationships are counter-intuitive and to some extent explain the mixed results that initiatives such as concurrent engineering and cross-functional teams have had on project performance.

Even at a basic level, the knowledge evolution framework has proven useful to deepening our understanding of managing complex product development projects. With some further research, the framework could be developed into a set of tools that can be used to plan for and manage these projects.

6.3 Further research

In order to achieve the goal of utilizing the knowledge evolution framework for the planning and management of product development frameworks, there are a few thrusts in which future research could prove potentially valuable. These thrusts represent different directions in which research could head covering different concentrations. The thrusts are:

- Calibration and project monitoring,
- Framework development, and
- Application development.

6.3.1 Calibration and project monitoring

This research has applied the knowledge evolution framework and analyzed relationships on simplified projects. After applying the framework on a project, one will realize the difficulty in estimating certain figures such as the probabilities of iteration and the learning rates. In reality, these will be extremely difficult to measure before a project starts which is when the simulations prove useful for policy and strategy testing.

This research assumes the relationships between the dispersion of an organization to the richness of the communication. In actuality, this would probably need to be benchmarked for different firms. How much of that communication relies on tacit transfer and how much relies on explicit transfer needs to be more closely studied. The assumption that tacit communication is "richer" but less precise also needs to be further verified.

One of the more pressing areas for research is that some benchmark for these figures for different types of processes and somehow calibrating them to the model for future uses. Questions that arise include whether these numbers would be firm specific, industry/practice specific, organizational structure specific. Some of the variables that would need some form of benchmarking and calibration include:

- Probabilities
- Learning Rates
- Equivalent dispersion (between geography, function and organization)

With some of this information, strategies and policies that are intended to improve project performance can be tried and tested both in the simulation model as well as implemented in true projects.

The strong points in using the Delta Design project (simple, constant organization, constant scope) also point to the weaknesses of the applicability of the findings. In reality, projects are never quite this simple. Another area of further research is therefore in applying the framework into more complex projects. Before this can happen, it may also be necessary to further develop the knowledge evolution framework to account for some of these complexities.

6.3.2 Framework Development

Another possible thrust of research could be focused on further developing the knowledge evolution framework. Firstly, the issues of scope, quality and rework have all been intentionally omitted to focus on the knowledge and information segregation issue. An immediate research need will be to incorporate these to evaluate the relative importance.

The model used in this research was kept simple. Some of the variables were kept to their first order approximations as the intention was really to demonstrate the use of the framework and that no there is no substantive evidence (without further research) of more meaningful estimates. For example, in the model, a uniform probability density function is used even though it is quite clear that iterations are more likely to occur at specific parts of the cycle. However, further research is needed to gather what more likely probability density functions would look like. Other areas where first order approximations were used and could see further research is the differentiation between in-stage and cross-stage iteration.

Another possible research area is reconciliation with the task-based environment. The knowledge evolution paradigm highlights certain areas of project management that have not previously been analyzed. This is, however, not to say that the traditional task-based paradigm is not useful. It has proven its use and will continue to contribute to the successful management of product development projects. Research is therefore necessary to somehow tie the two paradigms together and provide useful tools for managing projects

The third possible area of research in the framework development thrust is further detailed modeling. For example, if each stage of knowledge can be considered a stable subsystem, then the impact on differentiating between in stage iteration and cross stage iteration could be considered. Also, as resources in large product development firms are typically over multiple projects, the relationships arising from limited resources being applied to multiple projects could also be evaluated. These relationships are a necessary to understand in developing a framework that could be used as a useful tool for managing projects.

6.3.3 Applications Development

The final thrust of research possibilities is in the area of applications development. Although system dynamics is extremely useful and powerful for understanding the dynamic nature of projects, skills required to use it to improve project management are not that easily acquired. It seems therefore that some form of interface may be needed in order to provide a valuable toolset to project managers. This need calls for the development of suitable applications based on the knowledge evolution framework. Such applications will need to capture both the knowledge evolution paradigm as well as the task-based paradigm issues that the previous section describes.

One possibility is the further use of system dynamics engines with an improved user interface that allows for easy input of processes and scope. Another possibility is the use of multi-agent simulation models. Knowledge processes can be modeled as agents that interact with other agents.

6.4 Conclusions

This research has demonstrated how a knowledge evolution paradigm of modeling product development projects can be used to analyze issues that the traditional task-based tools have not. In particular, the knowledge evolution framework was able to account for iterations in projects in the absence of mistakes and lack of quality control. As the knowledge evolution framework is providing an alternative model of product development projects, it is capable of providing insights into project behavior where traditional task-based tools could not. Beyond insights, the framework can also provide measurements and levers for management that is directly linked to the knowledge in the project organization. This is especially important as firms are beginning to realize the importance of managing knowledge in an increasingly knowledge-based economy.

This research has also shown significant fit to actual simple projects. Six Deltan habitat design projects were executed concurrently and the results were similar to that predicted by the model showing that the framework can be used to simulate real projects. As the product development process is relatively complex and once system architecture and organizational decisions are implemented, they become extremely difficult to change; the knowledge evolution framework provides a means to test and simulate concepts at a relatively low cost.

Finally, this research has also highlighted some properties of projects that seem counter-intuitive. These include the dominance of probabilities of iteration and the value of “holding back”. The research has also allowed us to understand the circumstances in which these properties become significant. These properties have given not only given us a deeper understanding of product development projects but will also enable us to make more informed decisions when implementing strategies, policies and initiatives aimed at increasing project performance.

Understanding the knowledge evolution in product development projects is a first step towards managing the knowledge organization in firms that are involved in product development. Not only can the knowledge evolution paradigm provide insights as shown by this research, it is potentially a very useful project management tool. The knowledge evolution framework is the most direct way of leveraging a firm's human capital as it could provide information as to what type of knowledge is needed and when it would be needed. Understanding how knowledge is transferred within the project organization also enables firms to implement strategies and policies as to what media is suited for what type of knowledge transfer and whether the knowledge is tacit or explicit.

References

Abdel-Hamid, Tarek (1984). *The dynamics of software development project management: an integrative system dynamics perspective.* Unpublished doctoral theses. MIT, Cambridge, MA.

Abdel-Hamid, Tarek and Madnick, Stuart (1991). *Software project dynamics: an integrated approach* Prentice-Hall, Englewood Cliffs, NJ.

Abelson, H., Sussman, G.J., Sussman, J. (1985). *Structure and interpretation of computer programs.* MIT Press, Cambridge, MA.

Albano, Leonard D. (1992). *An axiomatic approach to performance based design.* Unpublished PhD Thesis, Massachusetts Institute of Technology, Cambridge, MA.

Allen, Thomas J. (1977). *Managing the flow of technology.* MIT Press, Cambridge, MA.

Ancona, Deborah, G. (1987). *Groups in organizations: extending laboratory models.* Annual review of personality and social psychology: group and intergroup processes. Sage, Beverly Hills, CA.

Ancona, Deborah .G. and Caldwell, David E. (1989). *Demography and design: predictors of new product team performance.* MIT Working Paper 3078-89.

Ancona, Deborah .G. and Caldwell, David E. (1991). *Cross functional teams: blessing or curse for new product development.* MIT Management Review, pp 11-16 Spring.

Bacon, Glenn et al. (1994). *Managing project to process management: an empirically based framework for analyzing product development time.* Management Science 41:3:458-484.

Beam, W.R. (1990). *Systems engineering: architecture and design.* McGraw-Hill Publishing Co., New York.

Belady, L.A. and Evangelisti, C.J. (1981). *System partitioning and its measure.* Journal of systems and software, vol. 2, no. 1, February 1981, pp. 23-29.

Berenato, D.A. (1988). *Issues and strategies for improving constructibility.* Unpublished Master's thesis, Massachusetts Institute of Technology, Cambridge, MA.

Bohn, Roger E. (1994). *Measuring and managing technological knowledge.* Sloan Management Review, Fall 1994.

Brethenoux, E. (1997). *The future of knowledge management, Part 1.* 1997

Bucciarelli, Louis L. (1991). *The Delta design game.* Massachusetts Institute of Technology, Cambridge, MA.

Bucciarelli, Louis L. (1994). *Designing engineers.* The MIT Press, Cambridge, MA.

Calantone, R. and Cooper, G. (1981). *New product scenarios: prospects for success.* Journal of Marketing, 45: 48-80.

Card, D.N. (1988). *The role of measurement in software engineering.* Software Engineering 88, Proceedings of the second IEE/BCS conference, Univ. of Liverpool, July 1988. Short Run Press, Exeter, U.K.

Churchman, C. West (1971). *The design of inquiring systems: basic concepts of systems and organization*, Basic Books ,New York.

Clark, Kim B. and Fujimoto, Takahiro (1991). *Product development performance, strategy, organization and management in the world auto industry.* Harvard Business School Press, Cambridge, MA.

Clark, Kim B. and Fujimoto, Takahiro (1991). *The power of product integrity: managing product life cycle from start to finish.* Harvard Business Review Paperback, Cambridge, MA.

Clark, Kim B. and Fujimoto, Takahiro (1991). *New product development performance.* Harvard Business School Press, Cambridge, MA.

Clark, Kim B. and Wheelwright, Steven (1993). *Managing new product and process development.* Harvard Business School Press, Cambridge, MA.

Cooper, Kenneth G. (1980). *Naval ship production: a claim settled and a framework built.* Interfaces. 10:6. The Institute of Management Sciences.

Cooper, Kenneth G. (1993a). *The rework cycle: benchmarks for the project manager.* Project Management Journal. 24:1.

Cooper, Kenneth G. (1993b). *The rework cycle: how it really works...and reworks....* Project Management Journal. 24:1.

Cooper, Kenneth G. (1993c). *The rework cycle: how projects are mismanaged.* Project Management Journal. 24:1.

Cooper, Kenneth G. (1994). *The $2000 hour: how managers influence project performance through the rework cycle.* Project Management Journal. 25:1.

Cooper, R.G. (1979). *The dimensions of industrial new product success and failure.* Journal of Marketing, 43: 93-103.

Crawley, Edward. (1999). *16.882 System Architecture. Class Notes.* Cambridge, MA: MIT,

Cronberg, A. and Saeterdal, A. (1975). *The potential of the performance concept – some questions.* Industrialization Forum, vol. 4, no. 5, pp 23-26.

Davenport, Thomas and Prusack, Laurence. (1998). *Working knowledge: How organizations manage what they know.* Harvard Business School Press, Cambridge, MA.

Dean, James W. Jr. and Susman, Gerald I. (1991). *Organizing for manufacturable design: managing product life cycles from start to finish.* Harvard Business Review Paperback. Cambridge, MA.

Deo, N. (1974). *Graph theory with applications to engineering and computer science.* Prentice-Hall, Englewood Cliffs, NJ.

Dougherty, D. (1987). *New products in old organizations: the myth of the better mousetrap in search of the beaten path.* Unpublished PhD thesis, Sloan School of Management, MIT, Cambridge, MA.

Eppinger, Steven D. et al. (1990). *Organizing the tasks in complex design projects.* ASME conference on design theory and methodology. Chicago, IL. Sept:39-46.

Eppinger, Steven D. (1994). *A model-based method for organizing tasks in product development.* Research in Engineering Design 6: 1-13, Springer-Verlag London Limited.

Festinger, L. (1954). *A theory of social comparison processes.* Human Relations, 1:117-140.

Ford, David N., Hou, Alex and Seville, Donald (1993). *An exploration of systems product development at Gadget, Inc.* Working paper D-4460. Systems Dynamics Group. Sloan School of Management, MIT, Cambridge, MA.

Ford, David N. (1995). *The dynamics of project management: an investigation of the impacts of project process and coordination on performance.* Unpublished Doctorate thesis. MIT, Cambridge, MA.

Ford, David N., and Sterman, J. D. (1998). *Dynamic modeling of product development processes.* System Dynamics Review, 14(1), 31-68.

Ford, David N., and Sterman, J. D. (1999). *Overcoming the 90% syndrome : iteration management in concurrent development projects.* MIT Sloan School of Management Working Paper.

Forrester, Jay W. (1961). *Industrial dynamics.* Productivity Press, Cambridge, MA.

Gates, Bill with Hemingway, Collin. (1999). *Business @ the speed of thought using a digital nervous system,* New York.

Galbraith, Jay T. (1977). *Organization design.* Addison-Wesley, Reading, MA

Goldratt, Eliyahu M. (1997). *Critical chain.* North River Press, Great Barrington, MA.

Goodman, P.S., Ravlin, E. and Schminke, M. (1987). *Understanding groups in organizations.* Research in Organizational behavior, 9:1-71. LL. Cummings and B.M. Staw (Eds)

Griffith, A. (1984). *Design rationalization and its effects on buildability and productivity.* Organizing and Managing Construction, Proceedings 4th International Symposium on Organization and Management of Construction, Univ. of Waterloo, April 1984, International Council for Building Research, pp 579-586.

Groth, Lars (1999). *Future organizational design: The scope for the IT-based enterprise.* John Wiley and Sons, Chichester.

Halpin, Daniel W. and Woodhead, Ronald W. (1980). *Construction management.* John Wiley & Sons, New York.

Hayes, Robert H., Wheelwright, Steven C. and Clark, Kim B. (1988). *Dynamic manufacturing: creating the learning organization.* The Free Press, New York.

Hoffman, E. (1985). *The effect of race-ratio composition on the frequency of organizational communication.* Social Psychological Quarterly, 29: 499-517.

Homer, Jack, Sterman, John, Greenwood, Brian and Perkola, Markku (1993). *Delivery time reduction in pulp and paper mill construction projects: a dynamic analysis of alternatives.* Proceedings of the 1993 International System Dynamics Conference. Cancun, Mexico, Monterey Institute of Technology.

Hubka, V. (1982). *Principles of engineering design* ed. By W.E. Eder, Butterworth Scientific. London.

Hubka, V. and Eder, W.E. (1988). *Theory of technical systems: A total concept theory for engineering design.* Springer-Verlag, Berlin/Heidelberg.

Iansiti, Marco (1993). *Real world R&D: Jumping the product generation gap.* Harvard Business Review. May-June:138-47. Cambridge, MA.

Jacobsen, H. (1955). *Information, reproduction, and the origin of life.* American Scientist, 43 (January 1995):119-127.

Jessen, Svein Arne (1990). *The motivation of project managers, a study of variation in Norwegian project manager's motivation and demotivation by triangulation of methods.*

Unpublished doctoral thesis. The Henley Management College and Brunel University, UK.

Kafura, D. and Henry, S. (1981). *Software quality metrics based on inter-connectivity.* Journal of systems and software, vol. 2, no. 2, June 1981, pp. 121-131.

Katz, R. (1982). *The effects of group longevity on project communication and performance.* Administrative Science Quarterly, 27: 81-104.

Kiesler, S.B. (1978). *Interpersonal processes in groups and organizations.* AHM Publishing, Arlington Heights, IL

Kim, Daniel H. (1988). *Sun Microsystems, Sun3 product development/release model.* Technical report D4113, System Dynamics Group. MIT, Cambridge, MA.

Klein, Janice A. (1994). *Maintaining expertise in multi-skilled teams.* Advances in Interdisciplinary studies of work teams, Volume 1 pp146-165, JAI Press..

Koomen, C.J. (1985). *The entropy of design: a study on the meaning of creativity.* IEEE Transactions on systems, man and cybernetics, vol. SMC-15, no. 1, January/February 1985, pp. 16-30.

Lavine, Jerrold I. (1999). *Parametric design constraints management using the design structure matrix: creation of an electronic catalog for a safety belt system.* Unpublished Master's Thesis, Massachusetts Institute of Technology, Cambridge, MA.

Leonard, Dorothy (1995). *Wellsprings of knowledge: building and sustaining the sources of innovation.* Harvard Business School Press, Cambridge, MA.

Li, Michael I. (1999). *A robust planning and control methodology for design-build fast-track civil engineering and architectural projects.* Unpublished Master's Thesis, Massachusetts Institute of Technology, Cambridge, MA.

Malone, Thomas W. and Crowston, Kevin (1990). *What is coordination theory and how can it help design cooperative work systems.* Proceedings of the conference on computer supported cooperative work. Los Angeles, CA.

Mar, B.W. and Palmer, R.N., (1989). *Does civil engineering need system engineering?* Journal of Professional Issues in Engineering, vol. 115, no. 1, January 1989, pp45-52.

McCain, B.E., O'Reilly, C.A. and Pfeffer, J. (1983). *The effects of departmental demography on turnover: the case of a university.* Administrative Science Quarterly, 26: 626-641.

Merrils, Roy (1991). *How northern telecom competes on time. Managing product life cycles from start to finish.* Harvard Business Review Paperback. Cambridge, MA.

Meyer, Marc H. and Lehnerd, Alvin P. (1997). *The power of product platforms: building value and cost leadership* The Free Press, New York.

Mintzberg, Henry (1979). *The structuring of organizations.* Englewood Cliffs, Prentice Hall.

Moder, Joseph, J., Phillips, Cecil R and Davis, Edward W. (1983). *Project management with CPM, PERT and precedence diagramming.* Van Nostrand Reinhold Co. New York.

Mohanty, S.N. (1981). *Entropy metrics for software design evaluation.* Journal of systems and software, vol. 2, no. 1, February, 1981, pp. 39-46.

Mueller, F.W. (1986). *Integrated cost and schedule control for construction projects.* Van Nostrand Reihnhold, New York.

Nadler, David, and Tushman, Michael. (1988). *Strategic organization design.* Scott, Foresman. Glenview, IL.

Nevens, T. Michael, Summe, Gregory L. and Uttal, Bro. (1991). *Commercializing technology: what the best companies do. Managing product life cycles from start to finish.* Harvard Business Review Paperback. Cambridge, MA.

Nevins, James L. and Whitney, Daniel (1989). *Concurrent design of products and processes, a strategy for the next generation in manufacturing.* McGraw-Hill, New York.

Newcomb, T.M. (1961). *The acquaintance process.* Holt, Reinhart and Winston, New York.

Nonaka, Ikujiro and Hirotaka Takeuchi. (1995). *The knowledge-creating company,* Oxford University Press.

Nonaka, Ikujiro (1991). *The knowledge creating company.* Harvard Business Review, November-December 1991

O'Reilly, C.A. and Flatt, S. (1989). *Effective team demography, organizational innovation and firm performance.* Working paper, University of California, Berkeley.

Osborne, Sean M. (1993). *Product development cycle time characterization through modeling of process iteration.* Unpublished master's thesis. MIT Sloan School of Management. Cambridge, MA.

Pahl, G. and Beitz, W. (1988). *Engineering design – a systematic approach.* The Design Council, London.

Peña-Mora, F. and Li, M. (2000), *A Robust Planning and Control Methodology for Design-Build Fast-Track Civil Engineering and Architectural Projects,* Journal of Construction Engineering and Management, ASCE, April, 2000.

Peña-Mora, F and Park, M. (2001a), *Robust Control of Cost Impact on Fast-tracking Building Construction Projects*, Journal of Construction Engineering and Management, ASCE, Approved Jan. 2001.

Peña-Mora, F and Park, M. (2001b), *Reliability Buffering for Concurrent Construction*, Journal of Construction Engineering and Management, ASCE, Submitted Mar. 2001.

Pimmler, Thomas U. and Eppinger, Steven D. (1994), *Integration analysis of product decompositions*, MIT Sloan School of Management Working Paper 3690-94.

Price, D.J. DesoSolla. (1965), *Networks of scientific papers,* Science 149: 510-515

Polanyi, Michael (1958). *Personal knowledge: towards a post-critical philosophy.* The University of Chicago Press. Chicago, IL.

Polanyi, Michael (1966). *The tacit dimension.* Doubleday. New York, NY.

Reichelt, Kimberly S. (1990). *Halter marine: a case study in the dangers of litigation.* Unpublished Master's Thesis. MIT Sloan School of Management, Cambridge, MA.

Rechtin E. and Maier M.W.(1997), *The art of systems architecting*, CRC Press, Boca Raton, FL

Richardson, George P. and Pugh III, Alexander L. (1981). *Introduction to system dynamics modeling with Dynamo.* MIT Press, Cambridge, MA.

Roberts, Edward, B. (1974). *A simple model of R&D project dynamics.* R&D Management. 5:1.

Roberts, E.B. and Wainer, H.A. (1971). *Some characteristics of technical entrepreneurs.* IEEE Transactions on Engineering Management, E.M – 18, 3.

Rodrigues, Alexander and Bowers, John. (1996). *System dynamics in project management: a comparative analysis with traditional methods.* System Dynamics Review. 12:2:121-139.

Rosenau, Milton D. and Moran, John J. (1993). *Managing the development of new products, achieving speed and quality simultaneously through multifunctional teamwork.* Van Nostrand Reinold, New York.

Sanderson, Susan and Uzumeri, Mustafa (1996). *Innovation Imperative.* Irwin Professional Publishing, Burr Ridge, IL.

Seville, Donald A. and Kim, Daniel H. (1993) *New product development management, flight simulator facilitator's guide v2.08.* Unpublished report, organizational learning center, Sloan School of Management, MIT, Cambridge, MA.

Shapero, A. (1967). *Preliminary analysis of inter-specialty mobility of technical professional manpower resources.* National Science Foundation.

Shannon, C. (1948). *A mathematical theory of communication.* Bell System Technical Journal, vol. 27, July 1948, pp 379-423.

Simon, Herbert A. (1969). *The sciences of the artificial.* MIT Press, Cambridge, MA/ London.

Simon, Herbert A. (1976). *Administrative behavior,* Third Edition, New York, The Free Press (First published in 1945).

Smith, Preseton and Reinertsen, Donald. (1991). *Developing products in half the time.* Van Nostrand Reinhold, New York.

Smith, Robert P. and Eppinger, Steven D. (1995). *Identifying controlling features of engineering design iteration.* Working Paper WP#3348. MIT Sloan School of Management. Cambridge, MA.

Steward, Donald V. (1981). *The design structure system: a method for managing the design of complex systems.* IEEE Transactions of Engineering Management.

Suh, N. P. (1990). *The principles of design.* Oxford University Press, New York.

Sveiby, Karl E. (1997). *The new organizational wealth: managing & measuring knowledge-based asset.* Berrett-Koehler Publishers, San Francisco, CA.

Takeuchi, Hirotaka and Nonaka, Ikujiro. (1991). *The new product development game. managing product life cycles from start to finish.* Harvard Business review paperback. Cambridge, MA.

Taylor, R.S. (1985). *The influence of research and development on design and construction.* Proceedings of the Institution of Civil Engineers, Part 1, vol. 78, June 1985, pp. 469-497.

Troy, D.A. and Zweben, S.H. (1981). *Measuring the quality of structured designs.* Journal of systems and software, vol. 2, no. 2, June 1981, pp. 113-120.

Ulrich, Karl T. and Eppinger, Steven D. (1994). *Methodologies for Product Design and development* McGraw-Hill, New York.

van Emden, M.H. (1970). *Hierarchical decomposition of complexity.* Machine intelligence 5, ed. B. Meltzer and D. Michie, Halstead Press, NY. Pp. 361-380.

Voss, C.A. (1985). *Determinants of success in the development of application software.* Journal of Product Innovation Management, 2:122-129.

Wagner, W.G., Pfeffer, J. and O'Reilly, C. A. (1984). *Organizational demography and turnover in top management groups.* Administrative Science Quarterly, 29: 74-92.

Ward, R.A., LaGory, S. and Sherman, S.R. (1985). *Neighborhood and network age concentration: does age homogeneity matter?* Social Psychological Quarterly, 48: 138-149.

Wetherbe, James C. (1995). *Principles of cycle time reduction: you can have your cake and eat it too.* Cycle Time Research Vol 1. No 1, pp1-24. 1995. FedEx Center for Cycle Time Research. Memphis, TN.

Wheelwright, Steven C. and Clark K.B. (1992). *Revolutionizing product development: quantum leaps in speed efficiency and quality* The Free Press, New York.

Wheelwright, Steven C. and Sasser, Earl W. Jr. (1991). *The new product development map. Managing product life cycles from start to finish.* Harvard Business review paperback. Cambridge, MA.

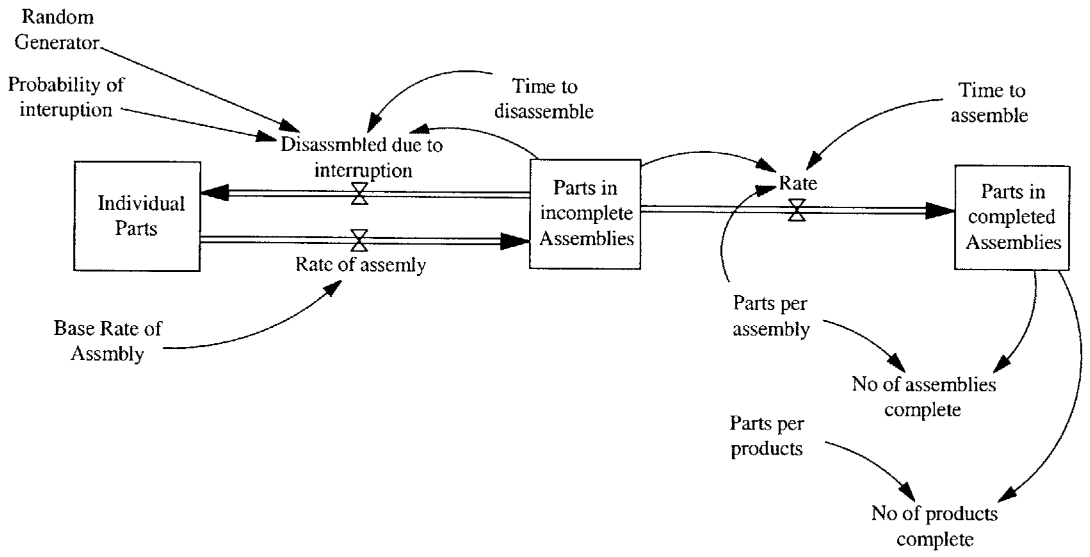
Womack, James P., Jones, Daniel T. and Roos, Daniel (1990). *The machine that changed the world, the story on lean production.* Rawson Associates. New York.

Wurman, Richard S. (1989). *Information anxiety.* Doubleday, New York.

Zaccai, Gianfranco. (1992) *How to make the client/consultant relationship more like a basketball game than a relay race.* Design Management Journal. 2.2.

Zenger, T.R. and Lawrence, B.S. (1989). *Organizational demography: the differential effects of age and tenure distributions on technical communication.* Academy of Management Journal, 32:353-376.

Appendix 1 Simon's Watchmakers Analogy



(01) Base Rate of Assmbly=

10

Units: Parts/Hour

(02) Disassmbled due to interruption=

IF THEN ELSE(Random Generator>(1-Probability of interuption), Parts in incomplete Assemblies

/Time to disassemble

```
            , 0 )
        Units: Parts/Hour

(03)    FINAL TIME  = 100000
        Units: Hour

(04)    Individual Parts= INTEG (
                Disassmbled due to interruption-Rate of assemly,
                        1e+006)
        Units: Parts

(05)    INITIAL TIME  = 0
        Units: Hour

(06)    No of assemblies complete=
                Parts in completed Assemblies/Parts per assembly
        Units: Dmnl

(07)    No of products complete=
                Parts in completed Assemblies/Parts per products
        Units: Dmnl

(08)    Parts in completed Assemblies= INTEG (
                Rate,
```

0)

Units: Parts

(09) Parts in incomplete Assemblies= INTEG (

+Rate of assemly-Disassmbled due to interruption-Rate,

0)

Units: Parts

(10) Parts per assembly=

100

Units: Parts

(11) Parts per products=

1000

Units: Parts

(12) Probability of interuption=

0.01

Units: Dmnl

(13) Random Generator=

RANDOM UNIFORM(0, 1,1)

Units: Dmnl

(14) Rate=

IF THEN ELSE(Parts in incomplete Assemblies>(Parts per assembly-1), Parts in incomplete Assemblies

/Time to assemble ,0)

Units: Parts/Hour

(15) Rate of assemly=

Base Rate of Assmbly

Units: Parts/Hour

(16) SAVEPER = 100

Units: Hour

(17) TIME STEP = 1

Units: Hour

(18) Time to assemble=

1

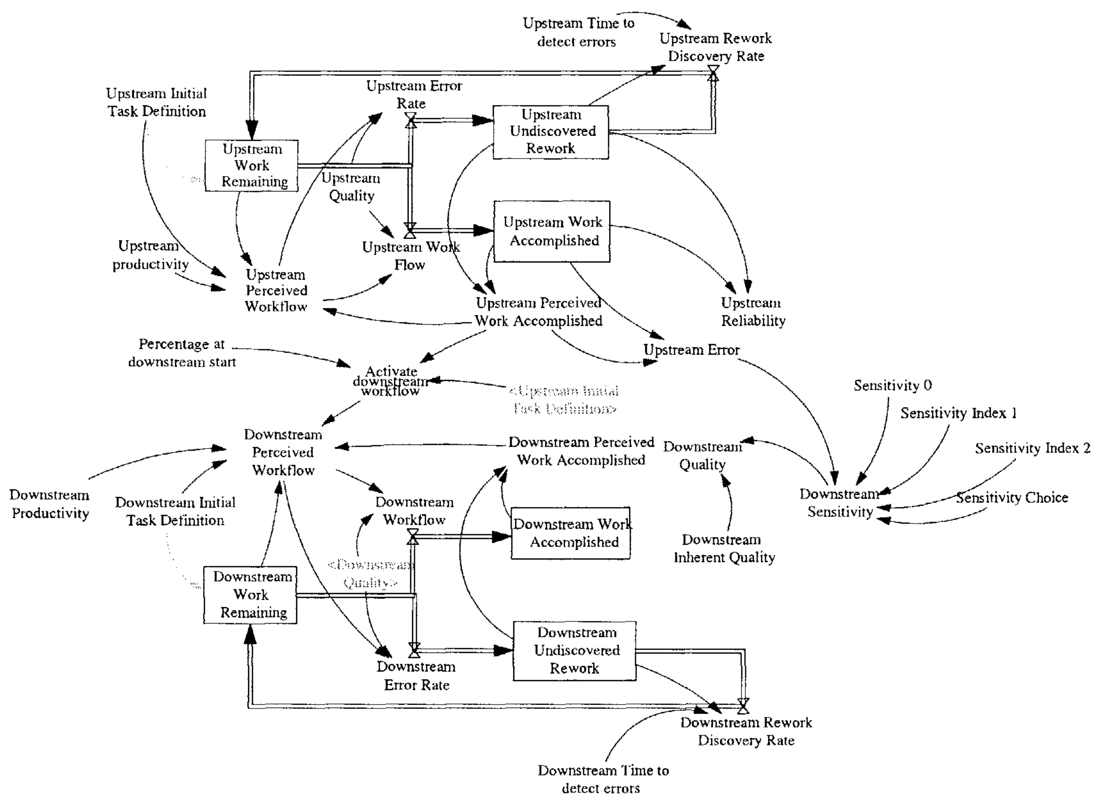
Units: Hour

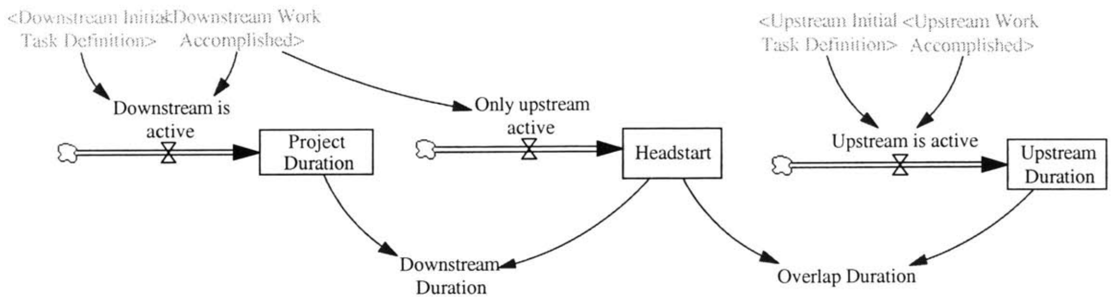
(19) Time to disassemble=

1

Units: Hour

Appendix 2 "Rework and Overlap" Model


Upstream Time to detect errors
Upstream Rework Discovery Rate
Upstream Initial Task Definition
Upstream Error Rate
Upstream Undiscovered Rework
Upstream Work Remaining
Upstream Quality
Upstream Work Accomplished
Upstream productivity
Upstream Work Flow
Upstream Perceived Workflow
Upstream Perceived Work Accomplished
Upstream Reliability
Percentage at downstream start
Upstream Error
Activate downstream workflow
<Upstream Initial Task Definition>
Sensitivity 0
Sensitivity Index 1
Sensitivity Index 2
Downstream Perceived Workflow
Downstream Perceived Work Accomplished
Downstream Quality
Downstream Productivity
Downstream Initial Task Definition
Downstream Workflow
Downstream Work Accomplished
Downstream Sensitivity
Sensitivity Choice
Downstream Inherent Quality
<Downstream Quality>
Downstream Work Remaining
Downstream Undiscovered Rework
Downstream Error Rate
Downstream Rework Discovery Rate
Downstream Time to detect errors



(01) Activate downstream workflow=

if then else(Upstream Perceived Work Accomplished/Upstream Initial Task Definition

>Percentage at downstream start,1,0)

Units: Dmnl

(02) Downstream Duration=

Project Duration-Headstart

Units: Month

(03) Downstream Error Rate=

Downstream Perceived Workflow*(1-Downstream Quality)

Units: Unit/Month

(04) Downstream Inherent Quality=

0.7

Units: Dmnl

(05) Downstream Initial Task Definition=

1000

Units: Unit

(06) Downstream is active=

if then else(Downstream Initial Task Definition-Downstream Work Accomplished

>0,1,0)

Units: Dmnl

(07) Downstream Perceived Work Accomplished=

Downstream Undiscovered Rework+Downstream Work Accomplished

Units: Unit

(08) Downstream Perceived Workflow=

if then else(Downstream Work Remaining>0:AND:Activate downstream workflow

=1, 100*Downstream Productivity(Downstream Perceived Work Accomplished

/Downstream Initial Task Definition

),0)

Units: Unit/Month

(09) Downstream Productivity(

[(0,0)-(1,3)],(0,0.51),(0.075,0.89),(0.124,1.47),(0.175,1.76),(0.248,2.01
),(0.311,2.05),(0.404,2.04),(0.492,1.92),(0.55,1.81),(0.586,1.75),(0.656,1.63
),(0.691,1.49),(0.737,1.22),(0.779,0.82),(0.843,0.482),(0.909,0.237),(1.01
,0.079))

Units: Unit/Month

(10) Downstream Quality=

Downstream Sensitivity*Downstream Inherent Quality

Units: Dmnl

(11) Downstream Rework Discovery Rate=

Downstream Undiscovered Rework/Downstream Time to detect errors

Units: Unit/Month

(12) Downstream Sensitivity=

if then else(Sensitivity Choice=1, Sensitivity 0(Upstream Error),

if then else (Sensitivity Choice=2, Sensitivity Index 1(Upstream Error),

Sensitivity Index 2(Upstream Error)))

Units: Dmnl

(13) Downstream Time to detect errors=

6

Units: Month

(14) Downstream Undiscovered Rework= INTEG (

Downstream Error Rate-Downstream Rework Discovery Rate,

0)

Units: Unit

(15) Downstream Work Accomplished= INTEG (

Downstream Workflow,

0)

Units: Unit

(16) Downstream Work Remaining= INTEG (

+Downstream Rework Discovery Rate-Downstream Error Rate-Downstream Workflow

,

Downstream Initial Task Definition)

Units: Unit

(17) Downstream Workflow=

Downstream Perceived Workflow*Downstream Quality

Units: Unit/Month

(18) FINAL TIME = 200

Units: Month

(19) Headstart= INTEG (

Only upstream active,

0)

Units: Month

(20) INITIAL TIME = 0

Units: Month

(21) Only upstream active=

if then else(Downstream Work Accomplished=0,1,0)

Units: Dmnl

(22) Overlap Duration=

Upstream Duration-Headstart

Units: Month

(23) Percentage at downstream start=

0.5

Units: Dmnl

(24) Project Duration= INTEG (

Downstream is active,

0)

Units: Month

(25) SAVEPER =

TIME STEP

Units: Month

(26) Sensitivity 0(

[(0,0)-(1,1)],(0,1),(1,0))

Units: **undefined**

(27) Sensitivity Choice=

2

Units: Dmnl

(28) Sensitivity Index 1(

[(0,0)-(1,1)],(0,1),(0.03,0.5),(0.125,0.125),(0.5,0.03),(1,0),(1.1,0))

Units: Dmnl

(29) Sensitivity Index 2(

[(0,0)-(2,1)],(0,1),(0.25,0.5),(0.5,0.25),(0.75,0.125),(1,0),(1.1,0))

Units: Dmnl

(30) TIME STEP = 1

Units: Month

(31) Upstream Duration= INTEG (

Upstream is active,

0)

Units: Month

(32) Upstream Error=

if then else (Upstream Perceived Work Accomplished>0,

(Upstream Perceived Work Accomplished-Upstream Work Accomplished)/Upstream Perceived Work Accomplished

, 0)

Units: Dmnl

(33) Upstream Error Rate=

Upstream Perceived Workflow*(1-Upstream Quality)

Units: Unit/Month

(34) Upstream Initial Task Definition=

1000

Units: Unit

(35) Upstream is active=

if then else(Upstream Initial Task Definition-Upstream Work Accomplished>

0,1,0)

Units: Dmnl

(36) Upstream Perceived Work Accomplished=

Upstream Undiscovered Rework+Upstream Work Accomplished

Units: Unit

(37) Upstream Perceived Workflow=

if then else (Upstream Work Remaining>0,100*Upstream productivity(Upstream Perceived Work Accomplished

/Upstream Initial Task Definition), 0)

Units: Unit/Month

(38) Upstream productivity(

[(0,0)-(1,3)],(0,0.51),(0.075,0.89),(0.124,1.47),(0.175,1.76),(0.248,2.01

),(0.311,2.05),(0.404,2.04),(0.492,1.92),(0.55,1.81),(0.586,1.75),(0.656,1.63

),(0.691,1.49),(0.737,1.22),(0.779,0.82),(0.843,0.482),(0.909,0.237),(1.01

,0.079))

Units: Unit/Month

(39) Upstream Quality=

0.9

Units: Dmnl

(40) Upstream Reliability=

if then else (Upstream Work Accomplished>0, 1-Upstream Undiscovered Rework

/Upstream Work Accomplished,1)

Units: Dmnl

(41) Upstream Rework Discovery Rate=

Upstream Undiscovered Rework/Upstream Time to detect errors

Units: Unit/Month

(42) Upstream Time to detect errors=

6

Units: Month

(43) Upstream Undiscovered Rework= INTEG (

Upstream Error Rate-Upstream Rework Discovery Rate,

0)

Units: Unit

(44) Upstream Work Accomplished= INTEG (

Upstream Work Flow,

0)

Units: Unit

(45) Upstream Work Flow=

Upstream Perceived Workflow*Upstream Quality

Units: Unit/Month

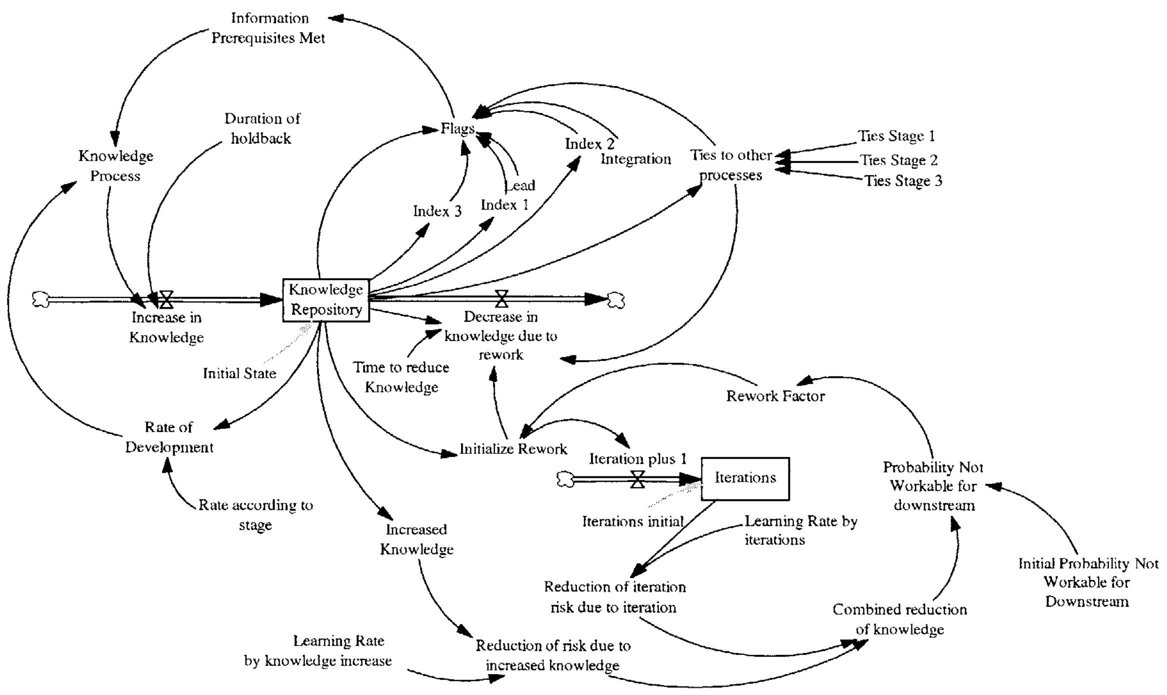
(46) Upstream Work Remaining= INTEG (

-Upstream Error Rate-Upstream Work Flow+Upstream Rework Discovery Rate,

Upstream Initial Task Definition)

Units: Unit

Appendix 3 KE Model of Delta Design



(01) Combined reduction of knowledge[Process]=

Reduction of iteration risk due to iteration[Process]*Reduction of risk due to increased knowledge

[Process]

Units: Dmnl

```
(02)    Decrease in knowledge due to rework[Process]=
                if then else (Initialize Rework[Process]=1,5,
                if then else (Initialize Rework[Architect]=1:AND:Ties to other processes[
        Process,Architect]>0 , (Knowledge Repository[Process
                ]-integer(Knowledge Repository[Process]))/(2*Time to reduce
Knowledge),
                 if then else (Initialize Rework[ProjectManager]=1:AND:Ties to other
processes
        [Process,ProjectManager]>0 , (Knowledge Repository
                [Process]-integer(Knowledge Repository[Process]))/(2*Time to reduce
Knowledge
        ),
                 if then else (Initialize Rework[StructuralEngineer]=1:AND:Ties to other
processes
        [Process,Architect]>0 , (Knowledge Repository
                [Process]-integer(Knowledge Repository[Process]))/(2*Time to reduce
Knowledge
        ),
                  if then else (Initialize Rework[ThermalEngineer]=1:AND:Ties to other
processes
        [Process,ThermalEngineer]>0 , (Knowledge Repository
                [Process]-integer(Knowledge Repository[Process]))/(2*Time to reduce
Knowledge
        ),0)
```

```
            )
           )
          ))
     Units: 1/Minute


(03) Duration of holdback[Process]=
          0,0,0,4
     Units: Minute


(04) FINAL TIME = 120
     Units: Minute


(05) Flags[Process, Target]=
          if then else(Ties to other processes[Process,Target]=0, 0,
          if then else(Ties to other processes[Process,Target]>0:AND:Knowledge
Repository
     [Target]>=3,0,
          if then else(Ties to other processes[Process,Target]=1:AND:ABS(Index
1[Process
     ,Target])<Integration, 0,
          if then else(Ties to other processes[Process,Target]=2:AND:ABS(Index
3[Process
     ,Target])<1, 0,
```

if then else(Ties to other processes[Process, Target]=3:AND:Index 1[Process

,Target]>0, 0,

if then else(Ties to other processes[Process, Target]=4:AND:Index 1[Process

,Target]>Lead, 0,

if then else(Ties to other processes[Process, Target]=5:AND:Index 2[Process

,Target]>=1, 0,

1)))))))

Units: Dmnl

(06) Increase in Knowledge[Process]=

DELAY FIXED(Knowledge Process[Process],Duration of holdback[Process] , 0)

Units: 1/Minute

(07) Increased Knowledge[Process]=

Knowledge Repository[Process]-integer(Knowledge Repository[Process])

Units: Dmnl

(08) Index 1[Architect,Process]=

Knowledge Repository[Process]-Knowledge Repository[Architect]

Index 1[ProjectManager,Process]=

Knowledge Repository[Process]-Knowledge Repository[ProjectManager]

Index 1[StructuralEngineer,Process]=

Knowledge Repository[Process]-Knowledge Repository[StructuralEngineer]

Index 1[ThermalEngineer,Process]=

Knowledge Repository[Process]-Knowledge Repository[ThermalEngineer]

Units: Dmnl

(09) Index 2[Architect,Process]=

Knowledge Repository[Process]-integer(Knowledge Repository[Architect])

Index 2[ProjectManager,Process]=

Knowledge Repository[Process]-integer(Knowledge Repository[ProjectManager

])

Index 2[StructuralEngineer,Process]=

Knowledge Repository[Process]-integer(Knowledge Repository[StructuralEngineer

])

Index 2[ThermalEngineer,Process]=

Knowledge Repository[Process]-integer(Knowledge Repository[ThermalEngineer

])

Units: Dmnl

(10) Index 3[Architect,Process]=

integer(Knowledge Repository[Process])-integer(Knowledge Repository[Architect

])

Index 3[ProjectManager,Process]=

integer(Knowledge Repository[Process])-integer(Knowledge Repository[ProjectManager

])

Index 3[StructuralEngineer,Process]=

integer(Knowledge Repository[Process])-integer(Knowledge Repository[StructuralEngineer

])

Index 3[ThermalEngineer,Process]=

integer(Knowledge Repository[Process])-integer(Knowledge Repository[ThermalEngineer

])

Units: Dmnl

(11) Information Prerequisites Met[Process]=

if then else(SUM(Flags[Process,Target!])>=1, 0,1)

Units: Dmnl

(12) Initial Probability Not Workable for Downstream[Process, Target]=

GET XLS CONSTANTS('delta05.xls','Sheet1','C10')

Units: Dmnl

(13) Initial State[Process]=

0,0,0,0

Units: Dmnl

(14) INITIAL TIME = 0

Units: Minute

(15) Initialize Rework[Process]=

if then else(Knowledge Repository[Process]>1:AND:Rework Factor[Process]>10

,1,0)

Units: Dmnl

(16) Integration=

1.2

Units: Dmnl

(17) Iteration plus 1[Process]=

if then else(Initialize Rework[Process]=1, 10,0)

Units: 1/Minute

(18) Iterations[Process]= INTEG (

Iteration plus 1[Process],

Iterations initial[Process])

Units: Dmnl

(19) Iterations initial[Process]=

0,0,0,0

Units: Dmnl

(20) Knowledge Process[Process]=

Information Prerequisites Met[Process]*Rate of Development[Process]

Units: 1/Minute

(21) Knowledge Repository[Process]= INTEG (

+Increase in Knowledge[Process]-Decrease in knowledge due to rework[Process

],

Initial State[Process])

Units: Dmnl

(22) Lead=

0

Units: Dmnl

(23) Learning Rate by knowledge increase[Process]=

0.05,0.05,0.05,0.05

Units: Dmnl

(24) Probability Not Workable for downstream[Process,Target]=

(1-(1-Initial Probability Not Workable for Downstream[Process,Target]*Combined reduction of knowledge

[Process])^(1/50))*10

Units: Dmnl

(25) Process:

Architect, ProjectManager, StructuralEngineer, ThermalEngineer

(26) Rate according to stage[Architect](

[(0,0)-(20,0.3)],(0,0.1),(1,0.1),(1.001,0.2),(2,0.2),(2.001,0.2),(3,0.2),

(3.001,0),(11,0))

Rate according to stage[ProjectManager](

[(0,0)-(20,0.4)],(0,0.2),(1,0.2),(1.001,0.1),(2,0.1),(2.001,0.2),(3,0.2),

(3.001,0),(11,0))

Rate according to stage[StructuralEngineer](

[(0,0)-(20,0.4)],(0,0.08),(1,0.08),(1.001,0.2),(2,0.2),(2.001,0.2),(3,0.2

),(3.001,0),(11,0))

Rate according to stage[ThermalEngineer](

[(0,0)-(20,0.4)],(0,0.07),(1,0.07),(1.001,0.2),(2,0.2),(2.001,0.2),(3,0.2

),(3.001,0),(11,0))

Units: 1/Minute

(27) Rate of Development[Process]=

Rate according to stage[Process](Knowledge Repository[Process])

Units: 1/Minute

(28) Reduction of iteration risk due to iteration[Process]=

1/(1+Iterations[Process])^Reduction Order[Process]

Units: Dmnl

(29) Reduction of risk due to increased knowledge[Process]=

1-Increased Knowledge[Process]^(1/Order of Reduction[Process])

Units: Dmnl

(30) Learning rate by iterations[Process]=

1,1,1,1

Units: Dmnl

(31) Rework Factor[Process]=

MAX (

RANDOM UNIFORM(Probability Not Workable for downstream[Process,Architect]

,10+Probability Not Workable for downstream[Process,Architect],0), MAX(

RANDOM UNIFORM(Probability Not Workable for downstream[Process,ProjectManager

] ,10+Probability Not Workable for downstream[Process,ProjectManager], 0)

,

MAX(RANDOM UNIFORM(Probability Not Workable for downstream[Process,StructuralEngineer

] ,10+Probability Not Workable for downstream[Process,StructuralEngineer],

0),

RANDOM UNIFORM(Probability Not Workable for downstream[Process,ThermalEngineer

] ,10+Probability Not Workable for downstream[Process,ThermalEngineer], 0)

)))

Units: Dmnl

(32) SAVEPER =

TIME STEP

Units: Minute

(33) Target:

Architect,ProjectManager,StructuralEngineer,ThermalEngineer

(34) Ties Stage 1[Process, Target]=

GET XLS CONSTANTS('delta05.xls','Sheet1','C3')

Units: Dmnl

(35) Ties Stage 2[Process, Target]=

GET XLS CONSTANTS('delta05.xls','Sheet1','h3')

Units: Dmnl

(36) Ties Stage 3[Process, Target]=

GET XLS CONSTANTS('delta05.xls','Sheet1','M3')

Units: Dmnl

(37) Ties to other processes[Process,Target]=

if then else (Knowledge Repository[Process]<1,Ties Stage 1[Process,Target

],

if then else(Knowledge Repository[Process]<2, Ties Stage 2[Process,Target

],

Ties Stage 3[Process,Target]))

Units: Dmnl

(38) TIME STEP = 0.1

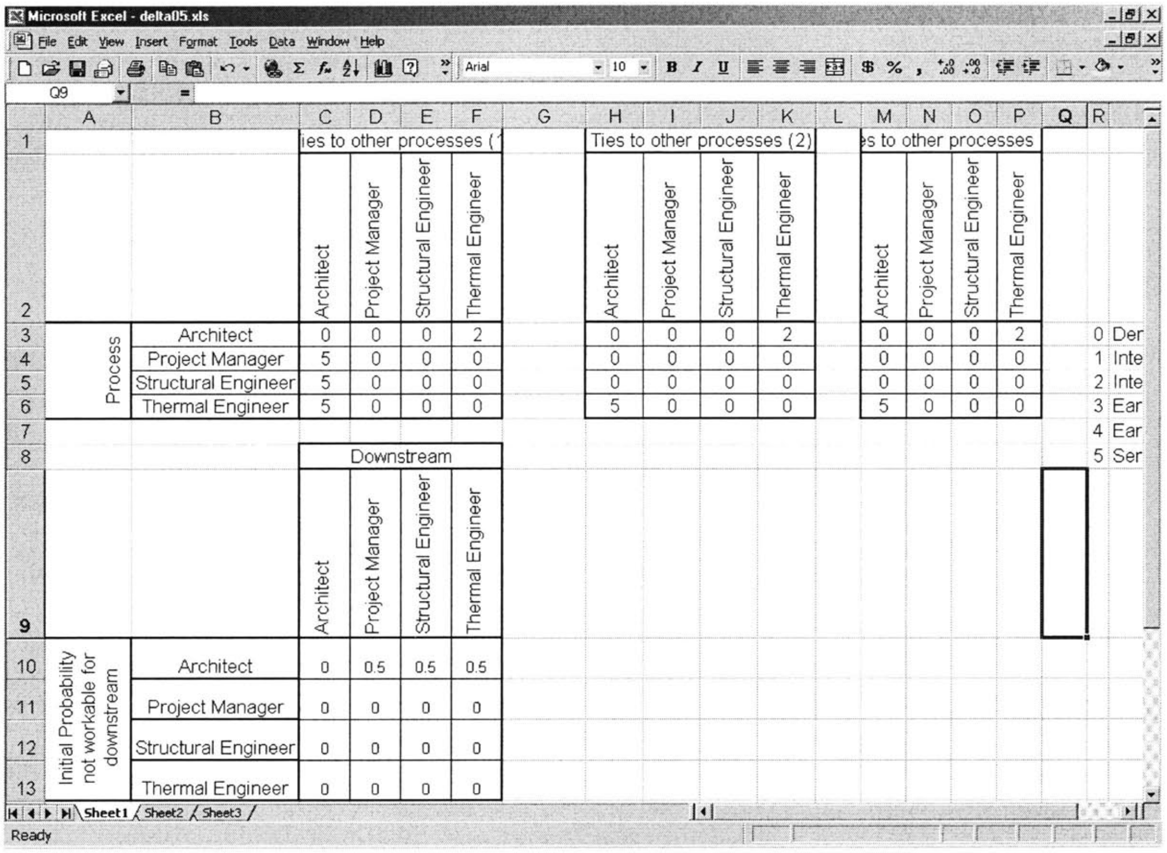
Units: Minute

(39) Time to reduce Knowledge=

0.1

Units: Minute

(40) Inputs from Delta05.xls


Microsoft Excel - delta05.xls
File Edit View Insert Format Tools Data Window Help
Arial
10
Q9
Sheet1
Sheet2
Sheet3
Ready

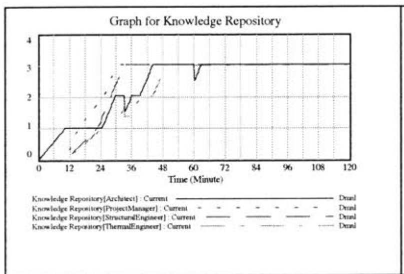
		ies to other processes (1					Ties to other processes (2)					s to other processes					
		Architect	Project Manager	Structural Engineer	Thermal Engineer		Architect	Project Manager	Structural Engineer	Thermal Engineer		Architect	Project Manager	Structural Engineer	Thermal Engineer		
Process	Architect	0	0	0	2		0	0	0	2		0	0	0	2		0 Der
	Project Manager	5	0	0	0		0	0	0	0		0	0	0	0		1 Inte
	Structural Engineer	5	0	0	0		0	0	0	0		0	0	0	0		2 Inte
	Thermal Engineer	5	0	0	0		5	0	0	0		5	0	0	0		3 Ear
																	4 Ear
																	5 Ser

		Downstream			
		Architect	Project Manager	Structural Engineer	Thermal Engineer
Initial Probability not workable for downstream	Architect	0	0.5	0.5	0.5
	Project Manager	0	0	0	0
	Structural Engineer	0	0	0	0
	Thermal Engineer	0	0	0	0

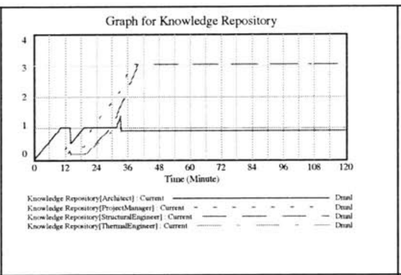
Appendix 4 Delta Model Simulations

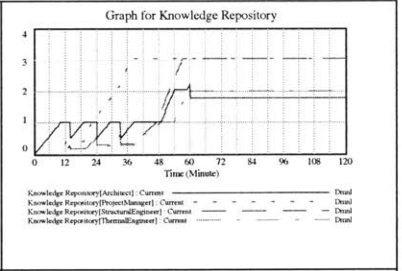
Appendix 4-1 Varying probabilities

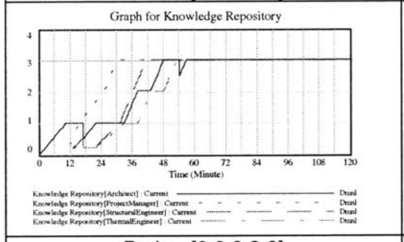
These are probabilities the various processes affect the Architect. The order of the probabilities are [Architect, Project Manager, Structural Engineer, Thermal Engineer]

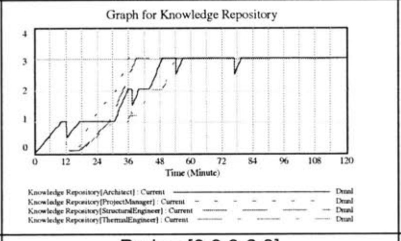


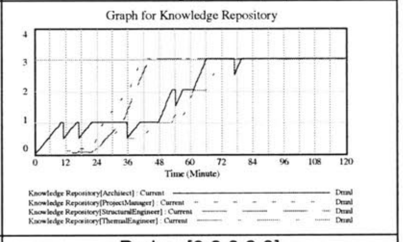
Prob = [0,0.3,0,0]

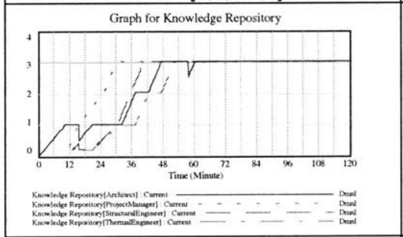

Prob = [0, 0.6, 0, 0]

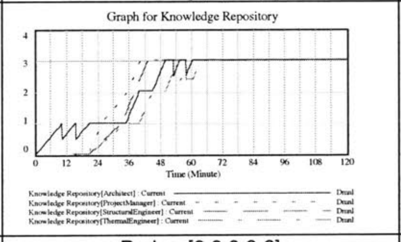

Prob = [0, 0.9, 0, 0]

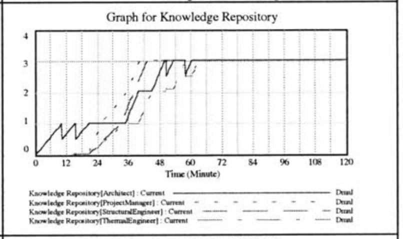

Prob = [0,0,0.3,0]

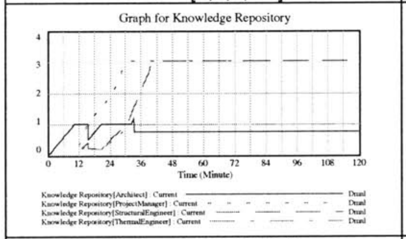

Prob = [0,0,0.6,0]

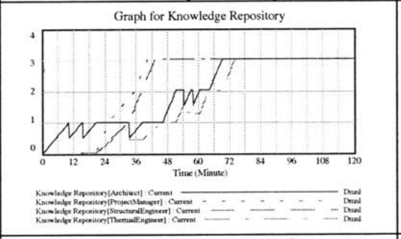

Prob = [0,0,0.9,0]

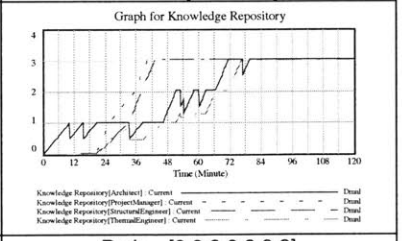

Prob = [0,0,0,0.3]

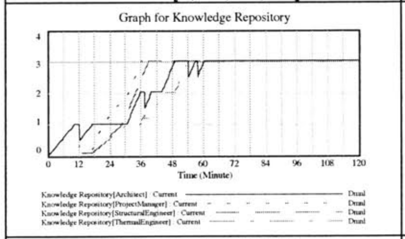

Prob = [0,0,0,0.6]

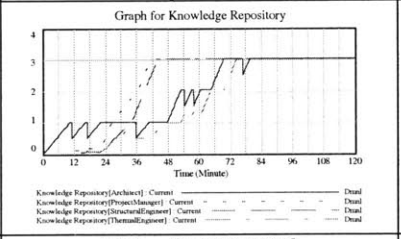

Prob = [0,0,0,0.9]

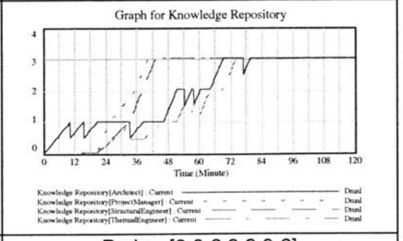

Prob = [0,0.3,0.3,0.3]


Prob = [0,0.6,0.6,0.6]

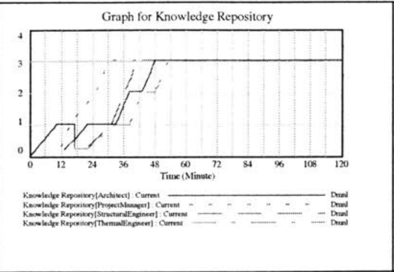
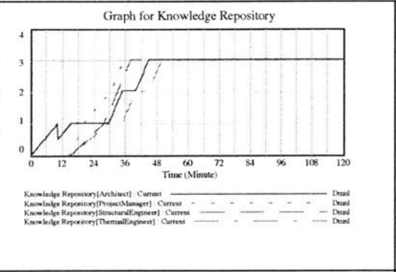
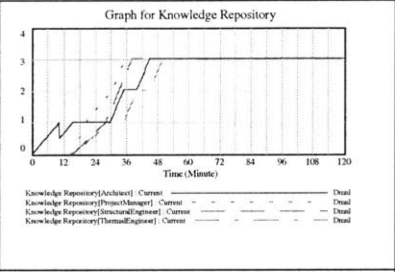
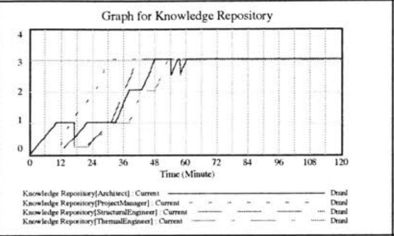
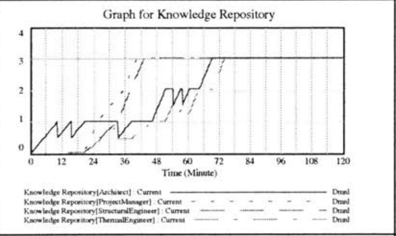
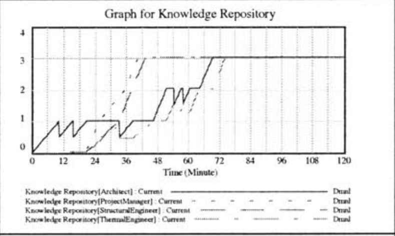
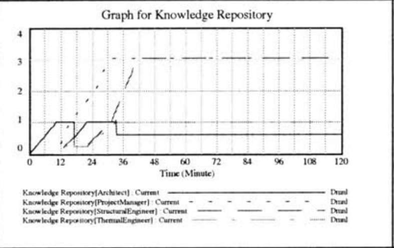
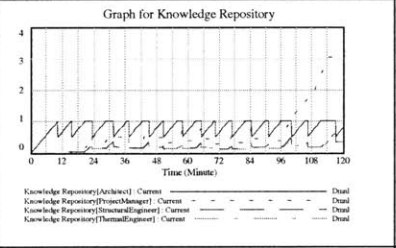
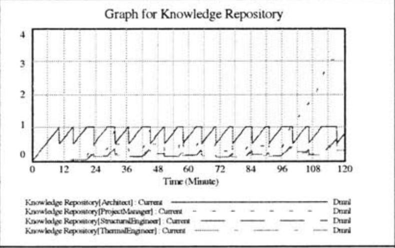
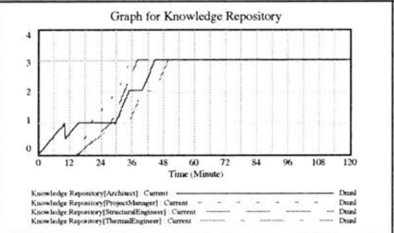
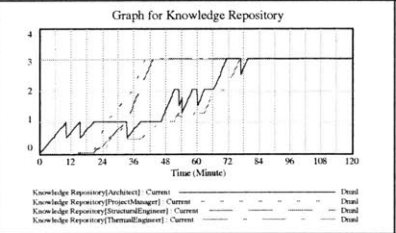
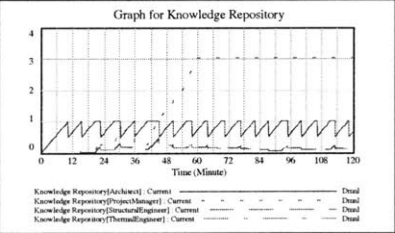
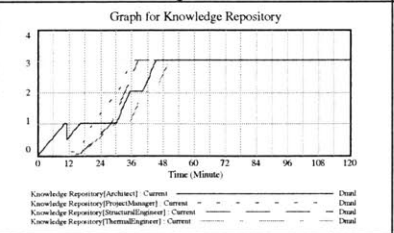
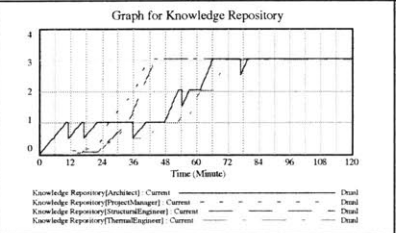
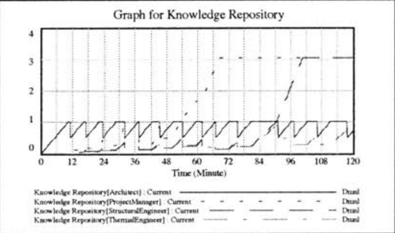

Prob = [0,0.9,0.9,0.9]


Prob = [0,0.3,0.6,0.3]


Prob = [0,0.3,0.9,0.3]


Prob = [0,0.6,0.9,0.6]

Appendix 4-2 Varying Learning Rates

		
Learning Rate (Iteration)=10 Learning Rate (Stage)=10	Learning Rate (Iteration)=10 Learning Rate (Stage)=1	Learning Rate (Iteration)=10 Learning Rate (Stage)=0.1
		
Learning Rate (Iteration)=1 Learning Rate (Stage)=10	**Learning Rate (Iteration)=1** **Learning Rate (Stage)=1**	Learning Rate (Iteration)=1 Learning Rate (Stage)=0.1
		
Learning Rate (Iteration)=0.1 Learning Rate (Stage)=10	Learning Rate (Iteration)=0.1 Learning Rate (Stage)=1	Learning Rate (Iteration)=0.1 Learning Rate (Stage)=0.1
		
Prob =[0,0.9,0.9,0.9] Learning Rate = 10	Prob =[0,0.9,0.9,0.9] Learning Rate = 1	Prob =[0,0.9,0.9,0.9] Learning Rate = 0.1
		
Prob = [0,0,0.9,0] Learning Rate = 10	Prob = [0,0,0.9,0] Learning Rate = 1	Prob = [0,0,0.9,0] Learning Rate = 0.1

[illegible]