Coordination in Complex Product Development

Samuel Suss

Submitted to the Faculty of Graduate Studies and Research

McGill University in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Engineering

November 2011

Acknowledgements

I wish to thank Professor Vince Thomson for supervising this research, and for his encouragement, advice and, support.

I would like to express my sincere gratitude to Professor Michael Paidoussis for introducing me to the fascinating world of engineering research and for his encouragement and advice throughout my career.

I am especially grateful to Dr. Khadidja Grebici, Dr. Onur Hisarciklilar, Dr. David Wynn, and many other research associates, postdoctoral fellows and graduate students at McGill University and at the Engineering Design Centre in Cambridge University, U.K., for their ongoing encouragement, advice and the privilege of participating in the many discussions I greatly enjoyed.

I wish to thank the Consortium for Research and Innovation in Aerospace in Quebec (CRIAQ) and NSERC for financially supporting this research.

Last but not least, I wish to express my heartfelt gratitude to my family, Chana, Matthew, Jonathan, and Joel Suss for their love and support during this endeavour.

Contents

List of Figures

List of Tables

Abstract

Complex product development is studied and its important characteristics are incorporated into an original model. The process is treated as a dynamic system in which information is created in each task as work is accomplished, and the amount of communicated information between tasks depends upon the magnitude of the interdependency between tasks. Information flow is modelled explicitly, and the model captures the dynamic complexity of projects with interdependent tasks. This is accomplished through the linkage of information exchange to the work accomplished in each task, the availability of resources, and the techniques used to manage the product development process. The uncertainty of information is explicitly modelled and this influences the development and magnitude of rework according to the way the process dynamically unfolds.

In this way, the influence of impediments to information flow on overall span time and effort are captured. The model is applied to the investigation of coordination and its effects on process behaviour under various conditions. Coordination mechanisms are applied in the model through the choice of input parameters that influence the degree of overlapping of tasks, the management of resources that process information, the delay of communication of information, and the interval of communication between tasks. Findings uncover the mechanisms driving the pace of progress in engineering design processes and highlight strategies that reduce span time in complex product development.

Simulations with the model illustrate the limits and benefits of overlapping and of set-based coordination with sequentially dependent tasks and different profiles of epistemic uncertainty reduction. Delays to information flow are shown to combine non-linearly to reach tipping points that greatly impact span time and effort. Different schemes for the management of critical resources that make use of data from the process itself are shown to be effective in reducing rework. Simulation of coordination schemes analogous to agile product development

methods demonstrate that important reduction in span times can be obtained for groups of tasks with high interdependence.

Abrégé

Le développement de produits complexes est étudié et ses caractéristiques importantes sont intégrées dans un modèle original. Le processus est traité comme un système dynamique dans lequel l'information est créée dans chaque tâche qui est en train d'être accomplie. La quantité d'informations échangées entre les tâches dépend de l'ampleur de l'interdépendance entre les tâches. La circulation de l'information est modélisée de façon explicite, et le modèle saisit la complexité dynamique de projets avec des tâches interdépendantes. Cela est accompli par le lien entre l'échange d'information et le progrès du travail dans chaque tâche, la disponibilité des ressources, et les techniques utilisées pour gérer le processus de développement du produit. L'incertitude de l'information est explicitement modélisée et cela influence le développement et l'ampleur des modifications en fonction du déroulement dynamique du processus.

De cette façon, l'influence des obstacles à la circulation de l'information sur le temps et l'effort global sont capturés. Le modèle est appliqué à l'enquête de coordination et de ses effets sur le comportement des processus sous des conditions différentes. Les mécanismes de coordination sont appliqués dans le modèle par le choix des paramètres d'entrée qui influencent le degré de chevauchement des tâches, la gestion des ressources qui traitent l'information, le délai de communication de l'information, et l'intervalle de communication entre tâches. Les résultats découvrent les mécanismes qui influencent le progrès dans le processus de conception d'ingénierie et qui mettent en évidence les stratégies qui réduisent le temps de développement des produits complexes.

Des simulations avec le modèle illustrent les limites et les avantages des différentes façons de coordination pour les processus qui ont des profils diverses de réduction de l'incertitude épistémique. Les résultats démontrent que les retards aux flux d'information se combinent de façon non linéaire pour atteindre les points de basculement qui influence beaucoup la durée et l'effort des projets du

développement. Les façons différentes de gérer les ressources importantes qui font usage de données du processus lui-même se sont avérées efficaces en évitant du travail supplémentaire. La simulation des mécanismes de coordination qui sont analogues aux méthodes agiles pour le développement des produits démontre qu'un réduction importante de la durée peut être obtenue pour des tâches fortement interdépendantes.

1 Introduction and Overview

1 And the whole earth was of one language, and of one speech. 2 And it came to pass, as they journeyed from the east, that they found a plain in the land of Shinar; and they dwelt there. 3 And they said one to another, Go to, let us make brick, and burn them thoroughly. And they had brick for stone, and slime had they for mortar. 4 And they said, Go to, let us build us a city and a tower, whose top may reach unto heaven; and let us make us a name, lest we be scattered abroad upon the face of the whole earth. 5 And the Lord came down to see the city and the tower, which the children built. 6 And the Lord said, Behold, the people is one, and they have all one language; and this they begin to do; and now nothing will be restrained from them, which they have imagined to do. 7 Go to, let us go down, and there confound their language, that they may not understand one another's speech. 8 So the Lord scattered them abroad from thence upon the face of all the earth: and they left off to build the city. 9 Therefore is the name of it called Babel; because the Lord did there confound the language of all the earth: and from thence did the Lord scatter them abroad upon the face of all the earth.

(From the King James Version of the Bible, Book of Genesis chapter 11)

Large complex endeavours require the combined efforts of many individuals to be attainable. We have become accustomed to being surrounded by objects built by groups of people with specialized knowledge, skills, and equipment working together in organizations.

There are two major types of methods that are seen as essential for organizations to go about the process of trying to accomplish objectives. These are "the differentiation of functions and positions" and "the deliberate conscious, intendedly rational, planful attempts to coordinate and direct activities" (Porter, Lawler et al. 1975).

This "differentiation of functions and positions" or division of labour has facilitated tremendous improvements in productivity and technology in specific functions by specialization of expertise. Division of labour into individual subtasks is believed to be more productive because it overcomes physical and cognitive knowledge limitations in people. Each individual can become more practised at a smaller task. Each of these subtasks can be further subdivided into more differentiated sub-subtasks so that each of these in turn can be done more

skilfully, more quickly and eventually with less effort. However, division of labour into differentiated subtasks often creates difficulties because of the interdependence between the subtasks. The interdependence between different people working on differentiated subtasks creates problems of coordination and a need for reliability. The problems that come about are nicely illustrated by Bavelas (1960).

> "When a job is made up of separate parts, and parts fit together, small errors accumulating in different parts may easily ruin the final product. Any beginner in woodworking will attest to that. He learns early, and often sadly, to study plans and consult them frequently, to work slowly, and to check his measurements.
>
> When the interdependent parts of a job are distributed among many different persons, all of the usual problems remain and new ones appear. The new problems stem from the nature of distributed work.
>
> A single workman who finds that the interlocking faces of a joint that he is building do not quite match will decide which face to modify or will scrap them both and begin again. When two men are involved, questions may arise as to which one of them will make the adjustment, and which of the two of them was in error. When work is distributed such problems are always latent in the relationship among men and functions. And the more a job is fragmented, the more numerous and the more difficult these problems may become!"

This little scenario gives a good sense of the interdependence that arises when work is divided.

The problems are more difficult when the product is intangible such as a curriculum, an education policy, or an engineering design. The limit in the ability of an organization to coordinate differentiated activities is often seen as one of the limiting factors in the effectiveness of large scale endeavours (Brooks 1975). The

effectiveness of the coordination strategies and methods used is often the way in which a large project can succeed in meeting its goals.

This thesis is about the coordination of activities in a complex product development process. In this context a complex process means that it involves many people performing differentiated functions. Complex product development takes place in the aerospace, defence, and automobile industries for example, where there are often hundreds or thousands of engineers collaborating in the design process. The work involved in product development is carried out in the minds of the participants with the use of tools to perform analyses and to describe the state of the work. Unlike a manufacturing process where materials move from activity to activity and undergo operations that change their state in an observable way, product development is information processing. It is difficult to observe the change in state of the object being worked on, the design of the product. The differentiated subtasks in product development are not only the design tasks for different parts of the product, but are also the different tasks of design, analysis and testing of each of these parts. The artefacts in the process are reports, models, drawings, simulations, and prototypes of parts of the object being designed.

Product development (PD) is a critically important part of the product lifecycle, consuming a large proportion of the overall time period of bringing the product to market, and setting about 70% of the product cost (Wheelwright and Clark 1992). It has been estimated that each day's delay in introducing a new model of an automobile into the market represents a one million dollar loss in profit (Clark, Chew et al. 1987). In the electronics sector, the rule of thumb is that the first two manufacturers that get a new generation product to market lock up 80% of the business (Port 1989).

The implementation of engineering design tools, concurrent engineering practices and product data management systems has contributed to reduced PD cycle times in recent years. However, in large PD projects where hundreds of engineers work to develop complex products, there remain significant inefficiencies. The amount

of waste in aerospace and defence PD programs is estimated at 60–90% of the charged time with about 60% of all tasks being idle at any given time (Oppenheim 2004). The actual time engineers working on PD spend on value-added activities is much less than half of their total working time. There is much efficiency lost in wasted communication, waiting for information and lack of coordination. Moreover, the increased use of inter-organizational collaboration in PD has highlighted the need for better coordination mechanisms.

With a focus on the aerospace industry, the goal of this dissertation is to find methods for achieving significantly faster new product development, i.e., the reduction of span time, the calendar time taken to create a product. The thesis is that this can be done with better coordination of the complex engineering design process and that this in turn can be achieved through a better understanding of how coordination strategies and tactics impact the process, and under what conditions they will be effective. It is believed that by treating the product development process as a complex system of elements: resources, tasks, and developing information, that interact to change the state of the system, new product development can be studied with computer modelling and simulation that are often used to analyze complex systems.

To this end, this thesis has the following objectives:

1. Identify or define a typology of coordination mechanisms that can be applied to PD;
2. Develop a model of PD that can be used to capture the impact of various coordination mechanisms on reduction of span time under many potential scenarios;
3. Use the model to determine the general principles governing the reduction of span time and the level of effort for various types of coordination mechanisms under various types of PD programs.

The complexity of the products in the aerospace industry makes prevalent the decomposition of the overall product into subsystems, and often into smaller

components in order to effectively organize development work. In recent years, there has been a strong tendency to outsource the development of some of these subsystems and components to external partners or suppliers. The focus of the research in this thesis is the mechanisms used to coordinate PD, where there is an overall system level of integration that takes place across design teams (internally or externally to the lead organization).

This thesis studies how to improve present coordination mechanisms:

- task decomposition and sequencing;
- systems to support information processing and decision making;
- the number and timing of approval cycles employed at various stages;
- the type, frequency, and efficiency of information exchange; and
- the manner in which shared resources are allocated.

In the following thesis, a model of the product development process is proposed based on observation of aerospace processes. The model is based on the flow of information that takes place in PD, incorporating the important mechanisms and impediments to the flow of this information, and how it relates to the interdependencies between tasks in the process and the evolving levels of uncertainty in the various tasks. Due to the dynamic nature of the interactions within the process, simulation is needed to execute the model. The model is then used to predict the performance of various product development processes and how they are affected by different coordination strategies that affect information flow. It is through the use of this model that better understanding of the importance of different approaches to the division of labour, timing of design decisions, and policies for allocation of resources and communication on the information flow in PD can be achieved.

This research is expected to contribute to knowledge at three levels:

- the development of a method to quantify the effects of coordination mechanisms under different conditions of uncertainty and task decomposition (division of labour);

- the study of the effects of various coordination mechanisms on PD;
- the development and validation of mechanisms that can greatly improve coordination among partners doing PD and reduce development times.

The dissertation is organized as follows: chapter 2 reviews the relevant research work in the fields of organization theory, coordination, complex systems theory, uncertainty, and engineering design methodology; chapter 3 is a detailed description of the model of the engineering design process; chapter 4 is a description of results of simulation experiments with the model under various conditions and scenarios; chapter 5 discusses the simulation experiments, verification and validation; and chapter 6 examines the achievement of objectives.

2 Literature Review and Discussion

The literature on the product development process and related fields is very broad. There has been relevant research in the fields of organization design, coordination theory, concurrent engineering, engineering design methodology, systems engineering, complexity, uncertainty, and iteration. In the following sections a review and discussion of concepts relevant to this thesis is presented and summarized.

2.1 Organization theory and the information processing framework

Research on organization theory has explored the ways in which large and complex tasks can most efficiently be performed and has led to the use of an information processing model (Galbraith 1977). This model shows how the efficient breakdown of a task into differentiated subtasks creates interdependencies that necessitate greater coordination of activities. Uncertainty is the core concept upon which this model is based and is defined as the difference between the information required to accomplish a task and the information currently residing with the actor charged with performing it. The basic proposition is that as the amount of uncertainty increases, there is an increasing frequency of non-routine, unique, consequential events which cannot be foreseen, and which require decisions to be made. Making decisions requires information gathering and communication about the state of affairs that led to the events (as well as authority granted by the owners or the stakeholders of the process). This is referred to as information processing; it takes time and requires resources. Therefore, increasing uncertainty increases workloads and time delays in the decision making mechanism.

As an example of an event that requires a decision to be made in a product development process, consider an aircraft design scenario in which a subsystem such as a landing gear assembly turns out to require more space in the wing than

was anticipated in an earlier design phase. A decision here is required by the organization as to how to accommodate this. Does the landing gear need to be redesigned, the wing redesigned, or both? To make the correct decision requires information such as what are the other systems that may be affected by the redesign of either the wing or landing gear? What is the effect on aircraft performance of each possible choice? What is the impact on the development schedule and cost budget? An organization that comes across these issues during the course of product development must have mechanisms in place to allow these decisions to be made optimally and in a timely manner. Organizations that have to deal with high levels of uncertainty must organize themselves to deal with high demands for information processing or must find more distributed forms of decision making in order to continue to effectively coordinate the actions of individuals.

Organizations have invented coordination mechanisms for collecting information, deciding, and disseminating information to resolve conflicts and guide interdependent actions. The collection of mechanisms used constitutes the organizing mode of the organization. Thompson (1967) proposed that there are three different types of dependence for which a different type of coordination mechanism is appropriate. Figure 2-1 illustrates these three types of dependency: pooled dependence (also known as fit dependence where several tasks work independently to produce part of the required result) is often coordinated by rules and standards; sequential dependence by planning; and reciprocal dependence by mutual adjustment. Other authors have identified programming (specifying predetermined behaviour), planning, and feedback as the basic coordination mechanisms and one chooses the appropriate mechanisms based on task situations (Hage, Aiken et al. 1971). The less routine and more diverse the situation, the more one chooses feedback as opposed to programming and planning.

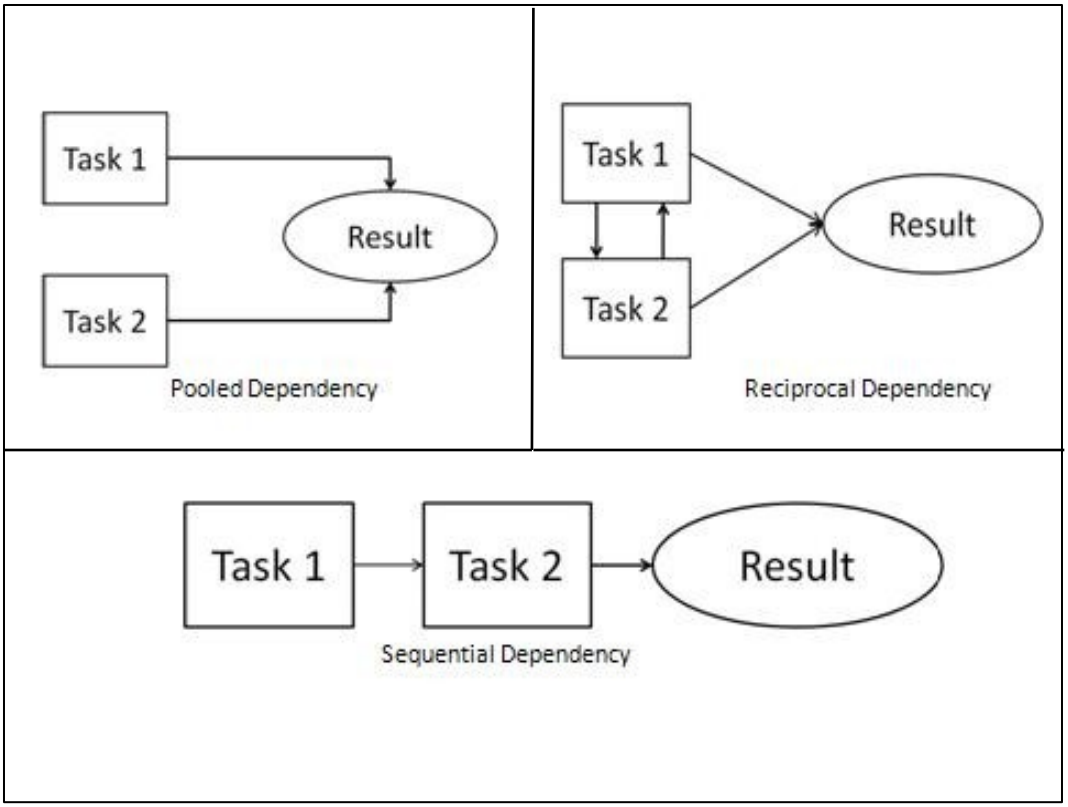


Figure 2-1 Types of dependency

Galbraith (1977) relates the use of coordination mechanisms in organizations to the level of uncertainty with which an organization must deal. The types of coordination mechanisms he describes are hierarchy, standardization, and goal setting. Then, as uncertainty continues to increase (with growth, product diversification, increasing competition) more exceptions to normal operating procedures necessitate further coordination mechanisms that would either require an increase in the capacity of the organization to process information to continue to make use of the hierarchy, or reduce the need for information processing by finding other ways to deal with exceptions, or both.

2.1.1 Use of hierarchy, standardization and goal setting

To illustrate these ideas further consider Figure 2-2. In order to complete the task shown at a high level of performance, the activities that take place in these various groups must be coordinated. The work of product design engineers must

be coordinated with process design engineers, etc. In development of complex products where the behaviour of thousands of people must be coordinated, it is impossible for them all to communicate with each other. The organization is simply too large to permit face-to-face communication to be a mechanism of coordination.

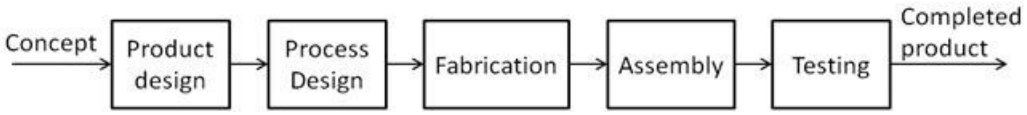


Figure 2-2 Horizontal work flow across a functional division of labour

In order to manage the division of labour, coordination is initially done in organizations by selecting some members to play a coordinating or managing role. These are invariably organized in a hierarchical form. In this way, referring to Figure 2-3, manager 1 can act to resolve an issue between product and process design while the general manager is required to resolve a problem between fabrication and product design. A hierarchy thus clearly establishes the lines of authority for everyone in the organization. It is also an efficient information processing mechanism in that far fewer communication channels are required than if direct communication was permitted between each subunit. If communication to coordinate interdependence takes place directly, there are $n(n\text{-}1)/2$ communication channels needed, where n is the number of subunits. If the hierarchical structure has a uniform span of control equal to s, then there are $(n\text{-}1)\ s/(s\text{-}1)$ communication channels only. Thus, as n increases the hierarchical structure grows according to n and not n^2 (Hage, Aiken et al. 1971). However, the cost for this economy in information processing capacity is the limited capacity of each channel and the restriction that each position can communicate directly only with those above and below it. Other contacts must take place through one or more intervening nodes. An increase in uncertainty creates more situations where people performing work need issues resolved, leading to greater requirements for information processing. This eventually overloads the hierarchical communication

channels and introduces delays and distortions that come about from the resultant loss of synchronicity of the information.

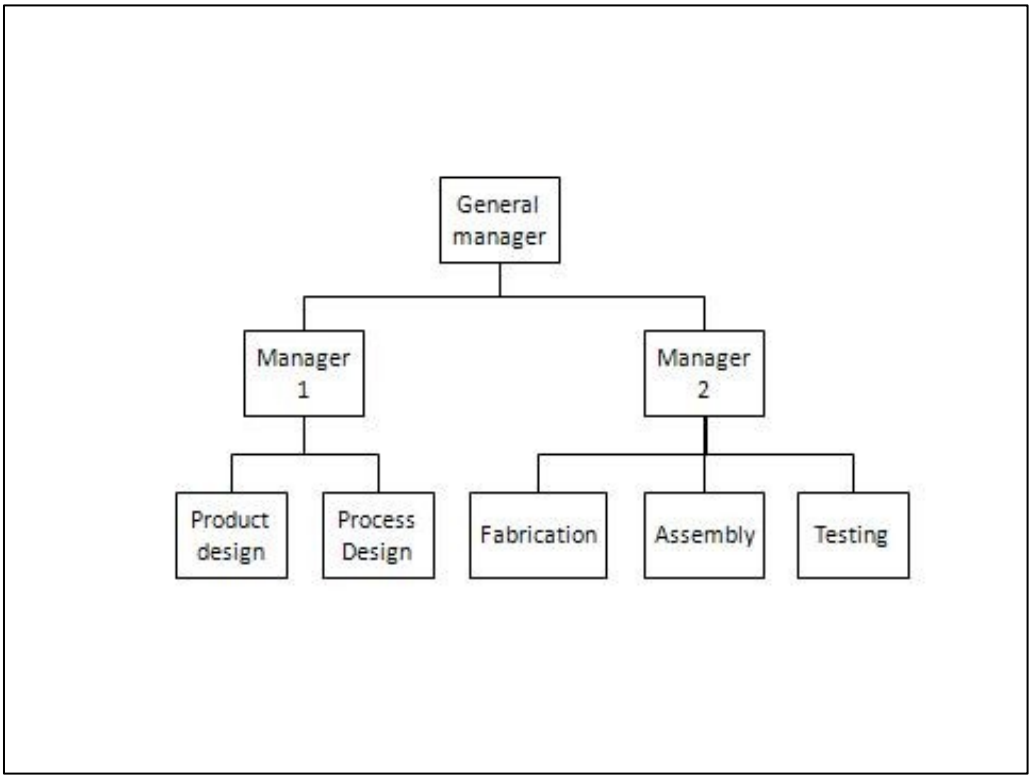


Figure 2-3 Hierarchical organization structure

To the extent that coordination decisions can be anticipated in advance, they can be codified as rules or procedures. Then, as situations arise on a day-to-day basis, the employees act out the behaviours appropriate to the situation and the resultant aggregate response is a coordinated pattern of behaviour. In this way decisions for those situations that can be anticipated in advance are in effect decentralized to the lowest levels and the vulnerable hierarchical channels are reserved only for those situations that are unforeseen. An example of the application of this coordination mechanism in a manufacturing process that is taken from the development of mass production in the Ford Motor Company in the early part of the 20th century (Hounshell 1984). Here, Ford created a structure based on the division of labour between those that manufactured different parts of the automobile and those that assembled it, where previously craftsmen would

painstakingly make and assemble each part specifically for one automobile. In order to eliminate the issues arising from different parts not fitting together it was necessary to find a way to manufacture the parts to standards so that the parts became interchangeable. Parts that were made according to the standards were sure to fit correctly. Thus, coordination between the different tasks in the process was achieved by the standardization of the parts and by the manufacturing processes that were able to produce them with tolerances acceptable for assembly.

At a level of task uncertainty where the number of exceptions increases until the hierarchy is overloaded, it becomes more efficient to bring the points of decision down to the points of action where the information exists. However, as the discretion exercised at lower levels of the organization is increased, the organization faces the question of how to ensure that employees consistently choose the appropriate response from the organization's point of view for job related situations with which they are faced. This is accomplished by the use of goal setting or delegating local discretion by planned targets and goals.

2.1.2 Coordination mechanisms with greater uncertainty

The ability of an organization to successfully utilize coordination by goal setting, hierarchy and standardization depends on the combination of the frequency of exceptions and the capacity of the hierarchy to handle them. Since greater uncertainty leads to more exceptions in normal operating procedures, continuing to use a hierarchy to coordinate successfully necessitates more information processing capacity. An alternative strategy would be to use other mechanisms to provide coordination without requiring further information processing capacity. The former alternative leads to investments in information systems and/or the creation of lateral relations. The latter alternative leads to methods to reduce input uncertainty by managing the environment in which the organization operates, the creation of self-contained tasks which can deal entirely with all exceptions, or the default which can be the creation or tolerance of slack (extra) resources or capacities. For example, in wing design, the scheduled time, weight allowance, or man-hours can be increased. In each case more resources are consumed, but the

number of exceptions requiring decision making is reduced, thus lessening the load on hierarchical channels.

The information processing model is borne out by empirical evidence shown in Figure 2-4. Here, it can be seen that as uncertainty increases the coordination mechanisms of rules and plans are used less often, whereas horizontal channels, the use of scheduled and unscheduled meetings are used more often, and the use of vertical channels of a hierarchy remains about the same.

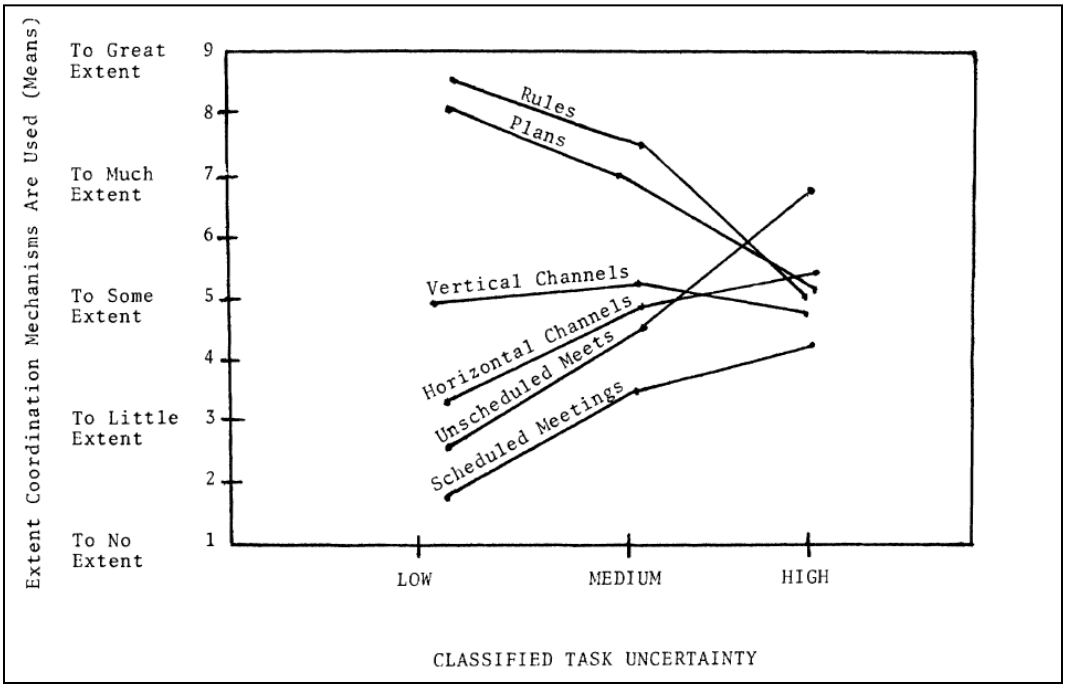


Figure 2-4 Profile of coordination mechanisms on classified levels of task uncertainty (VanDeVen, Delbecq et al. 1976)

In Table 2-1 a list of common mechanisms of coordination taken from a study in the management literature is shown (Martinez and Jarillo 1989). This survey of the types of actions taken by multinational corporations to make the most of diverse activities concludes that "there has been an evolution from unidimensional to multidimensional perspectives on coordination." Thus, the making of organizational structures that facilitate coordination in an organization is only one step towards better coordination. The implementation of the more informal, subtle

mechanisms shown in the lower part of Table 2-1 are a key part of making the formal structures actually facilitate coordination.

Table 2-1 Common coordination mechanisms (Martinez and Jarillo 1989)

Structural and formal mechanisms	
1.	Departmentalization or grouping of organizational units, thus shaping the formal structure
2.	Centralization or decentralization of decision making through the hierarchy of formal authority
3.	Formalization and standardization: written policies, rules, job descriptions, and standard procedures through instruments such as manuals, charts, etc.
4.	Planning: strategic planning, budgeting, functional plans, scheduling, etc.
5.	Output and behaviour control: financial performance, technical reports, sales and marketing data, direct supervision, etc.
Other mechanisms, more informal and subtle	
6.	Lateral or cross-functional relations: direct managerial contact, temporary or permanent teams, task forces, committees, integrators, and integrative departments
7.	Informal communication: personal contacts among managers, management trips, meetings, conferences, transfer of managers, etc.
8.	Socialization: building an organizational culture of known and shared strategic objectives and values by training, transfer of managers, career path management, measurement and reward systems, etc.

2.2 Coordination theory

We understand what coordination means intuitively when we attend a well-run event, observe a smoothly running manufacturing process, or watch a successful sports team, and notice how well coordinated the actions of the people involved are. Coordination is defined as "the management of dependencies between activities" (Malone, Crowston et al. 1999). This definition is consistent with common intuition and with the concepts about dependencies due to the division of labour into differentiated subtasks put forth in organization theory and described in section 2.1.

Coordination theory suggests that dependencies among activities and resources create coordination problems that constrain how the activities can be performed.

To avoid or overcome these constraints, additional work must be performed in the form of coordination mechanisms that manage the dependencies (Crowston 2003). Coordination theory catalogues possible dependencies, identifies alternative coordination mechanisms that can be used to manage each dependency, and describes the tradeoffs among these mechanisms. The typology developed from the work done in this area can help in identifying alternate coordination mechanisms by drawing on the similarities between a type of dependency in the typology, coordination mechanisms used to manage that dependency type, and examining whether a mechanism of that type can be implemented in a particular situation under examination.

In Table 2-2 some fundamental dependencies and examples of coordination mechanisms that can be used to manage them are shown. One ubiquitous type of dependency among different activities is the requirement to share the same limited resources. Solutions to the resource allocation problem are widely studied finding application in economics, organization theory, computer science, and industrial engineering. The table shows that shared-resource constraints can be managed by a variety of mechanisms such as "first come/first served," priority order, budgets, managerial decision, and markets.

The indentation in the dependency column of Table 2-2 indicates more specialized versions of general dependency types, and shows that coordination mechanisms for the general case can also be considered for the specialized case. One important special case of resource allocation is task assignment. This is allocating the scarce time of actors to the tasks they perform. The classification approach of coordination theory is that all the resource allocation methods listed in Table 2-2 are potentially applicable for task assignment as well. In recent years, the market mechanism indicated in the table as a form of coordination of shared resources (whereby potential actors bid for assignment to subtasks) is increasingly being used by original equipment manufacturers to find partners to which to outsource development tasks (Twigg 1998).

Table 2-2 Examples of common dependencies between activities and alternate coordination mechanisms for managing them (Malone and Crowston 1994)

Dependency	***Coordination Mechanisms***
Shared resources	First come/first served, priority order, managerial decision, market bidding
Task assignment	Same as for shared resources
Producer/consumer	
Prerequisite constraints	Notification, sequencing, tracking
Transfer	Inventory management (just in time, economic order quantity)
Usability	Standardization, ask users, participatory design
Design for manufacturability	Concurrent engineering
Simultaneity constraints	Scheduling, synchronization
Task/ subtask	Goal selection, task decomposition

Also of interest in product development processes is the producer/consumer type of dependency. In these processes there is information produced by one activity that is required by others and in particular cases there are prerequisite constraints, where the downstream/receiving activity cannot begin without the information from upstream/sending activities. Mechanisms used to manage this type of dependency range from simple notification given when the producer activity has completed its work and the consumer activity can begin, to explicit sequencing and tracking mechanisms to ensure that producer activities in a process complete their activities according to a schedule. Techniques using schedules such as PERT, GANTT charts and critical path methods are often used in organizations to help synchronize multiple activities and complex prerequisite structures (PMI 2001). The actions of scheduling and monitoring the status of 'deliverables', and taking appropriate measures to ensure that the scheduled inter-activity delivery dates are met is a common coordination method.

The coordination of the transfer dependency (where the actual transportation of the deliverable to the consumer activity takes place) has been greatly enhanced in recent years for PD with the widespread use of electronic documents that can be transferred almost instantaneously (Sweat 2001). In addition to simply transporting things or information, managing the transfer dependency also involves storing things being transferred from one activity to another. For instance, information is typically 'stored' as it develops prior to undergoing the process of communication. Much like the transfer of goods, there is the equivalent of an 'economic quantity' below which the cost of preparation of information for communication is too high relative to the amount of information being transferred. Similarly for the receiver, there is an economy in receiving the right information at the right time. Receiving information before it is required requires the user to invest time in examining it before it can be used, and receiving information too late often leads to a slowdown or outright stoppage of work requiring it. These factors need to be managed in the sense that decisions must be made as to how often to transfer developing information from the point of view of both the sender and the receiver.

Other dependencies that occur in a producer/consumer relationship between activities remain largely unaffected by enhancements in the ability to transfer information, namely the *usability* of the information provided. Here, common coordination mechanisms are: (i) standardization or creation of uniformly interchangeable outputs in a form that users already expect; (ii) users of the supplied items are asked what characteristics they want; (iii) participatory design or having the users of a product actively participate in its design. The concept of concurrent engineering is a special case of managing the usability dependence, applied in Table 2-2 to dependency between design and manufacturing.

Coordination to ensure and facilitate usability in engineering design must be related to the output delivered from one activity to another, which is information. Standardization can refer to having reports and drawings in a standard format and using well defined nomenclature so that the information is easily found and

understood. With the advent of electronic formatting of reports and drawings, this standardization can take on easily searchable forms that can be retrieved by computers. Information in the form of numeric or graphical data can be issued in a format that is directly usable by the downstream activity or at least in an agreed upon format that can then be manipulated electronically by the downstream activity. For instance, in the case of the design of an object that has forces imposed on it by its motion through the air, often the external shape is analyzed to determine the aerodynamic loads. The data describing the external surface of the object is typically generated using 3D modelling software. This data can be more readily used in computational fluid dynamics analysis software, if the format is translated appropriately.

Another basic dependency is that of simultaneity constraints. This can be regarded as a special case of the producer/consumer dependency if we consider that a producer/consumer relationship can be mirrored to one of reciprocal dependency, where the producer is also a consumer and vice versa. In this case there needs to be some kind of synchronicity between the activities in order to insure that the final result is satisfactory. This type of dependency is commonly encountered in design problems.

Task/subtask dependencies occur when a task is broken down into a group of activities that are subtasks for achieving the overall goal. There are, in general, many ways a given goal can be broken into work tasks, and a long-standing topic in organization theory involves analyzing different possible decompositions such as by function, by product, by customer, and by geographical region (Mintzberg 1979). In product development processes there is often a breakdown of the design task into subtasks based on engineering discipline (such as aerodynamics, stress analysis, etc.) as well as into different subassemblies or subsystems of the product (Pahl and Beitz 1996).

Coordination theory builds upon the fundamental types of dependency and mechanisms of coordination, and catalogues the specialization of processes that

are examples of these fundamental mechanisms in various scenarios. Here, we speak not only about the tactical coordination or action taken during the execution of a process to ensure that each subtask reliably has what it needs, when it needs it. Coordination is also strategic, where the structure of a process, the information processing and decision-making mechanisms are designed so that the execution of each subtask can take place in concert with the others with minimum effort and minimum delay.

2.3 Engineering design

In this thesis we are concerned with a particular activity, the development of a complex product and more specifically the engineering tasks associated with such a product. In a classical work on the methodology of engineering design (Pahl and Beitz 1996), the authors laid out what are, in essence, strategic guidelines and split the design process into four main phases:

- Product planning and clarifying the task (collecting information about the requirements that have to be fulfilled by the product and also about the existing constraints and their importance)
- Conceptual design (determination of the specification of principle by abstracting the essential problems, establishing function structures, searching for suitable working principles, and then combining these principles into a working structure)
- Embodiment design (often called layout or draft design. Here, variants that meet the specification of principle defined in the conceptual design phase are evaluated against technical and economic criteria. By appropriate combination of ideas and solutions the best layout is obtained)
- Detail design (in this phase the arrangements, forms, dimensions, surface properties of all of the parts are determined, materials are specified, production possibilities assessed, costs estimated, and all drawings and production documents produced).

The process is shown graphically in Figure 2-5. Note that iteration between the phases is described in recognition of the evaluation and feedback that takes place at each stage. This however does not portray what occurs within a phase, where much of the work of design consists of proposing solutions for specific design problems, and evaluating whether or not the solutions meet the criteria developed for the design problem. This aspect of the design process is discussed in the next section.

The complete product and each subsystem or component that is being developed has its own design process that generally follows the four mentioned phases. At the end of each phase, a design review is typically conducted to decide if the development has met the solution criteria. This design review is an opportunity to provide feedback and to synchronize the work of all those participating in the design. A result of the design review is a decision to continue to the next phase or to return to a previous phase with revised or clarified input information.

In the following sections several important aspects of engineering design are discussed.

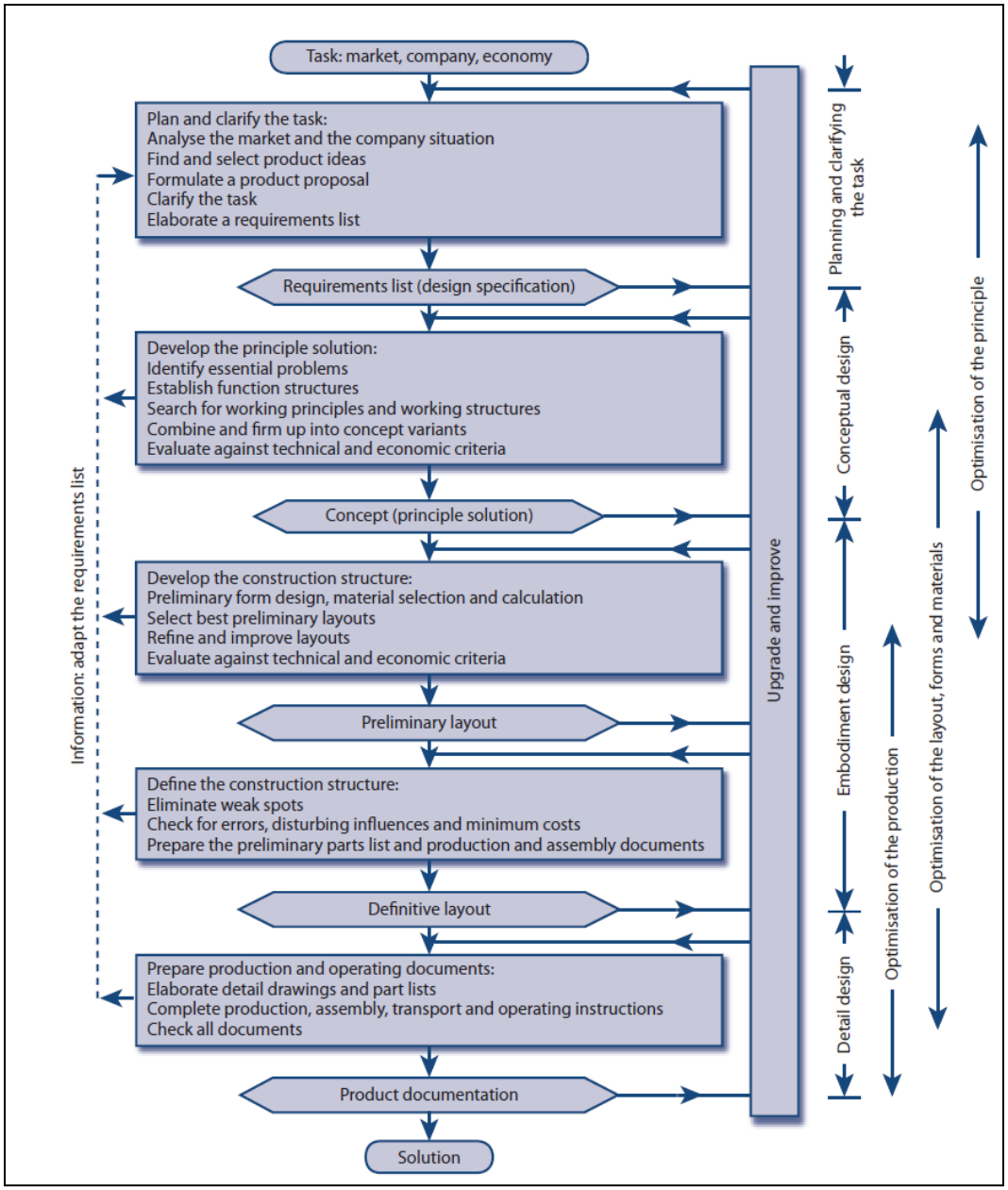


Figure 2-5 Pahl and Beitz' (1996) model of the engineering design process

2.4 Iteration in engineering design

It has been observed by many researchers that complex engineering design processes are often iterative, e.g., (Kline 1985; Peters 1986; Whitney 1990; Susman 1992; Fredriksson 1994; Safoutin and Smith 1996). Design iteration implies rework or refinement of activities to account for changes in their inputs

(Browning 1998). These changes can come from additional or changed information or failure to meet design objectives (Smith and Eppinger 1997). New information comes as an input to an activity from:

a) previously worked (upstream) activities changing their outputs, often as a result of a change in externally supplied requirements or assumptions
b) concurrent, coupled activities changing shared assumptions
c) subsequent (downstream) activities feeding back changes as errors or incompatibilities are discovered (Eppinger, Whitney et al. 1994).

Note that these three cases are all in fact consequences of the dependency of information between activities, where an 'activity' in a design process may also be a decision about whether or not a design solution meets set out criteria. The difference between the cases is the sequence in which the activities are executed relative to each other.

In particular, coupled activities must iterate as indicated in b) as the only means by which to arrive at a solution that will satisfy the requirements of each activity. An example of this is in the design of an aircraft where the aircraft's external shape determines the aerodynamic loads at each part of its mission profile, which determines the loads on the structure which in turn determine the structural design of the aircraft and ultimately influences its external geometry.

The design cycle for an aircraft typically begins with a proposal for an external shape that is expected to meet the aerodynamic, stability, and structural requirements. Upon verification of the stability and aerodynamic analyses, it may turn out that some change must be made to the proposed shape in order to meet all of the requirements or to optimize the aerodynamic or flight performance of the proposed configuration. A detailed calculation of the loads may then lead to another modification to the shape in order to provide sufficient structural strength to withstand the loads under certain flight conditions.

Thus, an iteration loop takes place which may converge rapidly or not, and this depends on the knowledge possessed by the organizational elements working on

the design and analysis activities about the type of aircraft configuration being developed. Note that here the discussion is not how long or how much effort is required to perform the design and analysis activities, which is dependent on other issues such as the innate difficulty of the design and analysis tasks, but rather the number of times that the reciprocally dependent activities need to iterate with interim information before a satisfactory solution is achieved. A design team that is experienced with a particular type of aircraft is able to make its initial design of the external shape so that it requires very little modification after analysis of aerodynamics, loads, and stress, because the team understands well how to balance the requirements of each of these disciplines and the requirements for the new design. If either the design group is less experienced or the new product design is of a different type with which the design group has no experience, it is more likely that the results of the analyses will lead to more redesign of the initial external shape. This 'newness' of a design problem relative to the design organization's capabilities is another characteristic of product development processes called uncertainty and is discussed further below.

Processes can be characterized as possessing a goal and a set of constituent actions which act to complete an outcome that instantiates the goal. Process incrementation is the sequential performance of constituent actions that results in the incremental completion of an outcome. Repetition is a specific pattern by which a process may incrementally move toward an outcome. However, most definitions of iteration in engineering design suggest that it consists of more than repetition, but rather it is associated with the improvement, evolution, or refinement of a design (Eppinger, Nukala et al. 1997). This characterization suggests a process that achieves a succession of intermediate improvements on the way towards a final outcome rather than a strictly incremental process that arrives at a single final outcome. Safoutin (2003) refers to this latter type of iteration as progression. Progression is a pattern of successive completion of intermediate outcomes. Feedback is an evaluation of an intermediate outcome generated in a progressive process relative to the goal. When progression iteration

is guided by feedback regarding the fitness of intermediate outcomes, it is referred to as feedback iteration. These latter three concepts are commonly associated with iteration in engineering design, leading to the recognition of three varieties of iteration: repetition, progression, and feedback (Safoutin 2003).

The partitioning of engineering design into phases as indicated in Figure 2-6 is an illustration of a progressive process that generates intermediate outcomes. Contrast this with an incremental approach in which each required function of a product is successively conceived, laid out, and finalized (Figure 2-6(a)). The progressive approach allows for intermediate design reviews during which a decision can be made with the entire product system in mind as to whether or not the intermediate versions of the design are suited to the requirements and whether or not to continue forward or to perform rework of an earlier phase (Figure 2-6(b)). Clearly there are advantages to considering the impact of a design solution of a component on the overall product as early as possible before continuing to flesh out the design in greater detail.

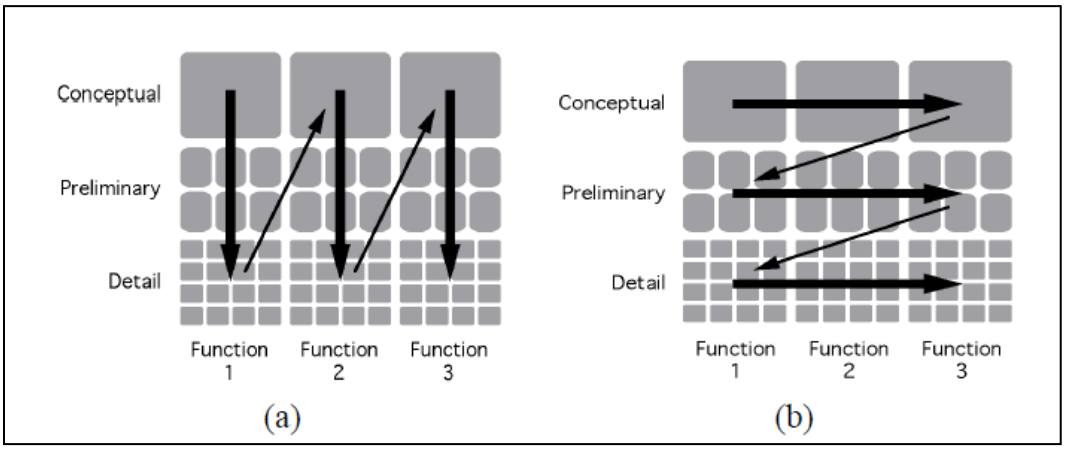


Figure 2-6 (a) Incremental and (b) progressive approaches to design (Safoutin 2003)

2.5 The effect of uncertainty in engineering design

There are several definitions of the term uncertainty mainly due to the fact that different types, causes and sources of uncertainty co-exist (Thunnissen 2004).

Perhaps one of the main distinctions regarding uncertainty in engineering design is between the lack of knowledge (epistemic) and stochastic (aleatory) uncertainty (Oberkampf, Helton et al. 2004). Epistemic uncertainty is derived from ignorance or incomplete information. Some epistemic uncertainty is reducible, for instance, via further studies, measurements or expert consultation. Stochastic uncertainty describes the inherent variation associated with a system or environment such as dimensional variation in a production system, the variation in design task duration, or an unexpected change in product requirements originating from the customer. This aleatory uncertainty cannot be reduced through more information or knowledge acquisition during the design process. Although gaining knowledge about variability may allow its influence to be mitigated through the design of systems that are adaptable, robust, flexible, etc. (Chalupnik, Wynn et al. 2009). In the engineering design process, it seems intuitive that the influence of variability may be managed through frequent information exchange between dependent activities, and one of the purposes of the analysis in this thesis is to determine the conditions under which this is most effective.

Drawing on the literature (Browning 1999; Grebici, Wynn et al. 2008), many components of uncertainty applicable to engineering design processes have been identified. However, many of these are related to the interdependence of design activities and to the lack of synchronicity of developing information, and as such we can classify them as ‘derived uncertainty’.

Fundamentally, if we view PD as a process of generating information that reduces uncertainty (Browning and Ramesh 2007), the rate of progress of the information developed by a task in PD is affected by how it reduces epistemic uncertainty. Furthermore, since by definition epistemic uncertainty in a task is reduced by increasing knowledge, it is argued here that progress in a task is a function of the amount of work done in the task and the input information it has received. This is evident since the means by which knowledge to complete a task is increased is through the information received and the work done to develop this information through analysis, testing, and research.

The aleatory uncertainty of information developed in a task and in the input information is always present and its influence must be taken into consideration. The other components of uncertainty occur as the design process unfolds based on the interactions of the elements participating in the process. In the product development model proposed in this thesis, these 'derived' aspects of uncertainty should be emergent in a process model that incorporates the dynamics of the interactions.

Two components of uncertainty within the design process were identified for inclusion in a model of the engineering design process (Suss, Grebici et al. 2010). Each of these aspects has influence on the iterative behaviour of design tasks.

- **Imprecision:** Aughenbaugh, Matthew et al. (2006) define precision as the gap between certainty and a decision maker's present state of information, i.e., the information currently available for decision-making. As design decisions are made, imprecision tends to be progressively reduced until a final, precise value is determined (Antonsson and Otto 1995). The need for greater precision justifies application of more sophisticated design tools and methods, and this may require progressively greater effort.
- **Instability:** More unstable descriptions are more likely to change and instability may be increased by events which increase the likelihood that rework is required. This component of uncertainty accounts for the gap between certainty and a state of precise information.

The term 'process progress' is similar to the term 'work state' (Carrascosa, Eppinger et al. 1998); these terms refer to the proportion of task completion. Wood, Antonsson et al. (1990) propose that as a design progresses, the level of design imprecision is reduced, although a degree of stochastic uncertainty usually remains. Having a high level of confidence in design information means that the information is detailed, accurate, robust, well-understood, and physically realistic (O'Donovan, Clarkson et al. 2003). Also, progress in a design process refers to a

process that achieves a succession of improvements or refinements on the way toward the final outcome (Safoutin 2003).

Hykin and Laming (1975) refer to work progress as a process in which the level of uncertainty in the artefact is reduced as the design progresses. We incorporated this reasoning in our model in the way in which we calculate the variation in uncertainty with the state of progress in a task. Since the reduction in epistemic uncertainty in a task is due to the learning or discovery of information required to complete the task, we would expect to find it to exhibit a slower rate of change at the beginning and end of the task (Ritter and Schooler 2002) as in a sigmoid or S-shape function. According to Carrascosa, Eppinger et al. (1998) the S-shape is the best shape function to represent task completion as it embodies a growth phenomenon similar to learning.

Here, a type of sigmoid curve called the Gompertz function is used to model the reduction in epistemic uncertainty as a function of work progress because of its flexibility in allowing for different rates of approach to the lower and upper asymptotes at activity start and end. The work progress curve can be transformed into an uncertainty reduction curve by subtracting the Gompertz function from its upper asymptote (discussed further in Chapter 3).

2.6 Quality in product development

ISO 9000 defines quality as "the degree to which a set of inherent characteristics fulfills requirements." The American Society for Quality defines quality in technical usage as "the characteristic of a product or service that bear on its ability to satisfy stated or implied needs."

Under the conventional paradigm, higher quality in PD can be achieved only at the expense of increased development expenditures and longer cycle times (Harter, Krishnan et al. 2000). However, an alternate view is that quality, cost, and cycle time in PD are complementary, i.e., improvements in quality directly relate to improved cycle time and productivity (Nandakumar, Datar et al. 1993). These improvements in quality are thought to arise from reduced defects and

rework in a mature[1] product development process (Harter, Krishnan et al. 2000) and better knowledge sharing, acquisition, integration, and application in new PD processes (Jing and Yang 2009).

2.7 Concurrent engineering

Concurrent engineering is one of the most influential, recent methods used in PD for reducing the span time of the process (Krishnan 1996). Concurrent engineering actually refers to several concepts: the idea of increased functional interaction in early activities; the overlapping of activities that normally are sequential; and the parallel execution of interdependent activities that require deliverables from each other. In the context of coordination theory, concurrent engineering is a coordination mechanism that can help to manage the producer/consumer dependency.

Research work on simple generic PD processes has studied various aspects of concurrent engineering in the effort to determine the parameters that lead to successful PD with reduced span time. This research has led to insights about the relation of the rate of upstream evolution of information and the degree of downstream sensitivity to change for the overlapping of activities (Figure 2-7 and Figure 2-8) and the importance of timely information exchange (Ha and Porteus 1995; Krishnan, Eppinger et al. 1997). This model states that "if the evolution of the upstream information is such that large changes were to happen near the completion of the upstream activity, then it would be difficult to reduce the lead time by overlapping, . . . , low sensitivity describes the case when substantial changes in the exchanged information value used by the downstream activity can be accommodated quickly by the downstream activity." The concept of evolution of information used in Krishnan's model is roughly equivalent to the rate of reduction of uncertainty discussed in the previous section. If uncertainty is

[1] Process maturity is defined by CMMI (Capability Maturity Model Integration) by the Software Engineering Institute of Carnegie Mellon University, http://www.sei.cmu.edu/cmmi/research/.

reduced quickly, the information generated by the task evolves quickly towards its final value.

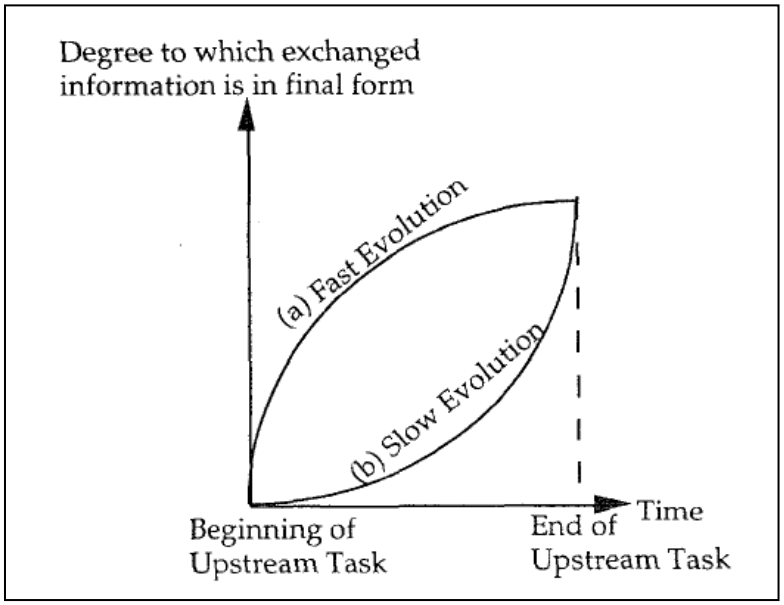


Figure 2-7 Upstream information evolution at the fast and slow extremes (Krishnan 1996).

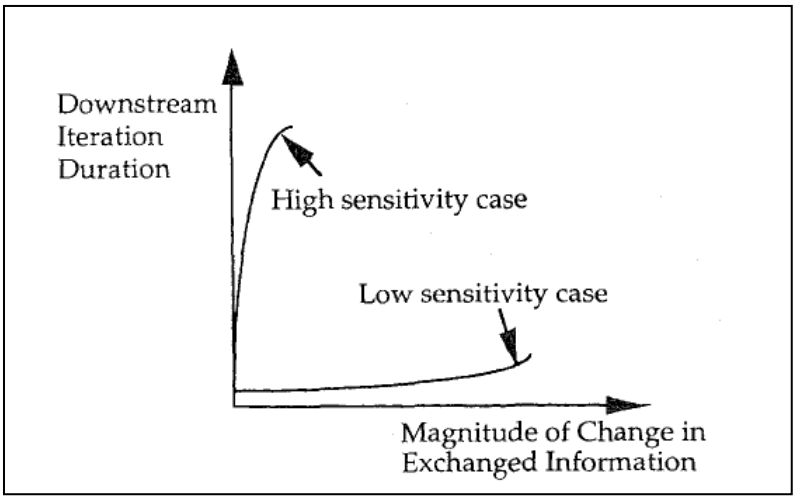


Figure 2-8 Downstream sensitivity at the high and low extremes (Krishnan 1996).

Other work has analyzed the relationship of functional interaction and overlapping of activities on effort and total span time (Bhuiyan, Gerwin et al. 2004). Studies have focused on the mechanisms of iteration and rework that occur

in PD as a result of the interdependencies of design activities (Smith and Eppinger 1997).

2.8 Complexity of engineering design processes

In the context of product development and engineering design processes, complexity is best defined in terms of systems theory. A system is defined as a collection of elements that act and interact together toward the accomplishment of some logical end (Schmidt and Taylor 1970). By this definition a product development process is one type of system where the elements are the tasks, subtasks, resources, and developing information.

A complex system is a system composed of "a large number of parts that interact in non-simple ways, …[such that] given the properties of the parts and the laws of their interactions, it is not a trivial matter to infer the properties of the whole" (Simon 1969). A product is complex when it has many interconnected parts, there are many dependencies between the parts, and these dependencies interact in numerous forms of relationships. The design process of a complex product is itself complex in that there are many activities to be performed that are interconnected through many and various dependencies of information.

The management of a complex process is challenging because of the difficulty of being able to predict what levers available to managers can produce the desired effects in the overall process. For example, if a project is behind schedule the act of adding resources to the activities that are late is often the response of a project manager. However, studies have shown that the addition of more people may add to the workload of the people already working on an activity since they have to instruct the newer people on what needs to be done while at the same time the new people are not yet effective (Black and Repenning 2001; Ford and Sterman 2003). Thus, adding more resources can initially cause a delay and if the activity is a provider of information to other activities this delay is likely to lead to ripples of delay throughout the project which may lead to eventual project failure. Thus, without a view of the interaction between elements of the system or a 'system

view' management actions may inadvertently exacerbate the problem or push a schedule problem into another category, such as cost or performance.

2.8.1 Design structure matrix (DSM)

PD processes can be more easily understood by applying some of the methods used in analyzing other complex systems. A useful system modeling tool that has been applied to PD is the design structure matrix (DSM) (also known as decision structure matrix, dependency structure matrix, dependency source matrix, and dependency structure method). It is similar to other tools such as dependency map, precedence matrix, reachability matrix, N2 diagram, and others.

The DSM is a visual representation that simply and concisely highlights interactions and facilitates predicting their implications in complex systems. With the use of matrix manipulation techniques DSM can also be used to perform several types of analyses of a system. Whereas these matrix tools have been used for many types of systems and analyses (Danilovic and Browning 2007; Collins, Yassine et al. 2009; Karniel and Y.Reich 2009), here, we are concerned with using an activity based DSM for a complex PD process.

Figure 2-9 shows an example of an activity based DSM for a process made up of 7 tasks or activities arranged in order of execution. The marks in the cells indicate the dependencies between each pair of tasks. Reading along the row for task 1 for example, the DSM shows that task 1 provides output to task 2, task 3, task 5, and task 6. Reading down column 3, the DSM shows that task 3 receives input from task 1, task 2 and task 5. Similarly, task 6 receives input from task 1, task 4 and task 5.

Therefore, one can see that marks above the diagonal represent feed forward relationships and marks below the diagonal represent feedback. If the tasks in the figure represent those in an engineering design process, then the matrix shows the flow of information from one task to another. Some of the tasks can be those of analysis, some can be testing, and some can be decisions. Iteration loops can be observed where there are tasks that are reciprocally dependent such as tasks 2 and

task 3, task 5 and task 3, and task 7 and task 2. There are loops of dependency such as 2-3-4-5-2 by virtue of the interdependence between tasks 2 and 3 and 2 and 5 and the sequential dependency of task 4 on task 2 and task 5 on task 3.

For a PD process, DSM is useful in visualizing the process and the information flow. Contrast this view with the typical flow diagram for the same process shown in Figure 2-10. It is much more difficult to visualize the dependency relationships between the tasks. The process DSM is useful for this reason in managing interfaces, analyzing and improving processes, highlighting iteration and rework, and as an aid in knowledge management.

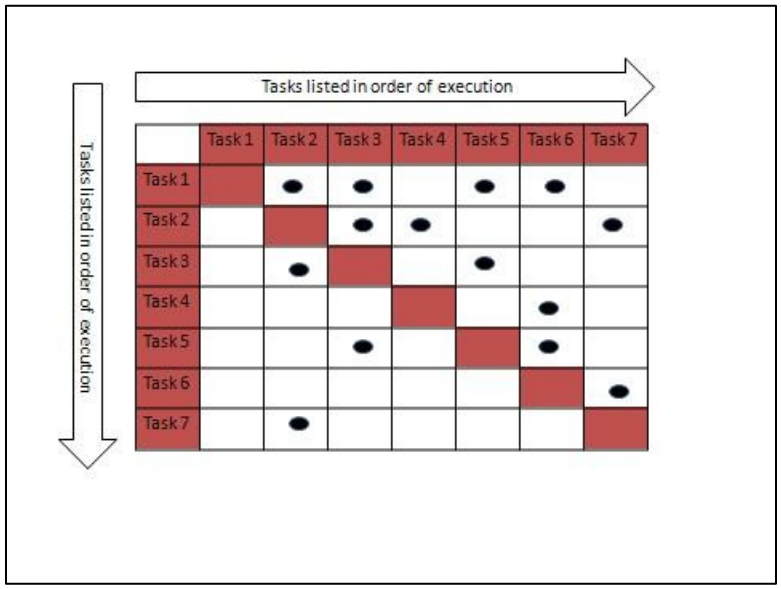


Figure 2-9 An example of an activity based DSM

The marks in the cells of the DSM only indicate that there is a dependency between a pair of tasks. However, numerical DSMs have been used for various purposes such as analyzing cost, schedule, and risk, where numbers have been put in each cell representing relative strength of the dependency.

The dependency types illustrated in Figure 2-1 are easily identified in a task-task DSM. The pooled tasks, also called independent in that they do not need any input from other tasks, are easily identifiable as those with no marks in the column of the matrix corresponding to the task (task 1 in Figure 2-9). If a group of tasks in

the project are sequentially dependent, the order for most efficient execution of these tasks is easily determined by sorting the rows and columns in the matrix so that all the non-zero elements are in the upper triangle. This indicates the sequence in which tasks should be executed so that the information required by each task is available for it before starting.

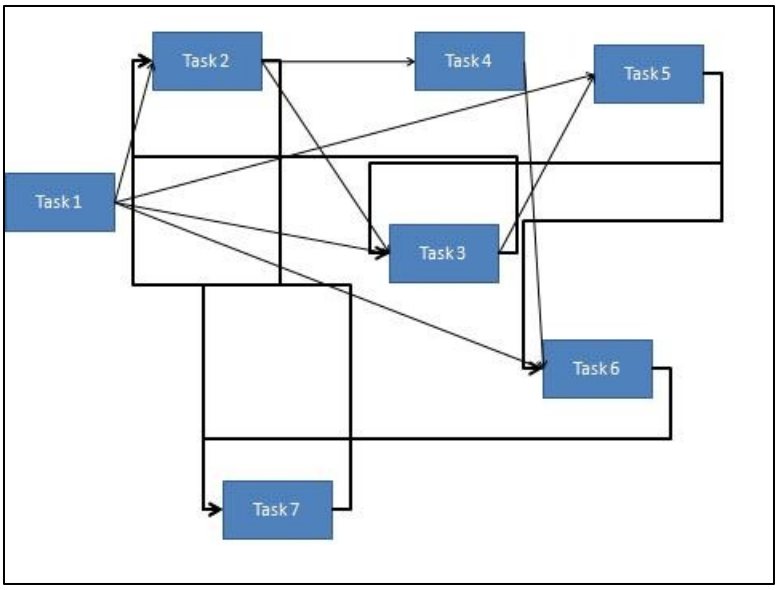


Figure 2-10 Flow diagram for the process in Figure 2-9

If some tasks in a project are reciprocally dependent, matrix manipulation can group the tasks involved in the same reciprocally dependent cycle into a cluster or block around the diagonal of the matrix. The task ordering, thus determined, indicates the most efficient sequence of tasks to minimize the process length, i.e., reduce the number of tasks not reciprocally dependent in between the sequence of tasks that are reciprocally dependent. In this way, tasks that are sequentially dependent or independent can be performed either before or after the block of reciprocally dependent tasks (Steward 1991).

In practice, the DSM of a PD process is quite sparse since generally there are many tasks that have no direct links to many other tasks. An analysis of dependencies between tasks can highlight the critical loops of reciprocally

dependent tasks and focus process improvement for greater coordination of the work of these tasks.

2.8.2 Simulation

A model or abstraction of a system we wish to study can be created in which aspects of the system are represented by simplified elements. If the system's state is time varying, a model can be built that incorporates the dynamic behaviour of the system with the use of simulation. The technique of simulation can be used to allow the various elements in a system to be represented within their bounds while preserving the interconnections between the elements.

If there is more than one behaviour possible for some elements in the system, and we can infer the likely behaviour, we can use a non-deterministic or stochastic model to provide the relationship of the behaviour of these elements with time. Simulation of a particular scenario using a stochastic model can be performed using computers to generate performance characteristics of the system under various conditions. Statistical data can then be gathered to predict how, on average, a particular system behaves under the particular tested conditions.

It is the complex combination of events that occur, brought about by the response of the various elements of the system and their interconnections, that make the understanding of product development difficult. In the next chapter the model built to achieve this understanding is described.

Simulation models of the information processing in PD have been developed (Levitt, Thomsen et al. 1999; Bhuiyan 2001). This work highlights the power of simulation in studying overall project performance as a function of various input parameters. In these simulations, the specific behaviour of actors was stochastically modeled with probabilities at decision points based on expected behaviour. The probability distributions of the likely behaviour of actors in the modeled process were hypothesized based on observations of behaviour in existing organizations engaged in developing products. These choices determine the behaviour of actors as the simulation unfolds for variations in events such as

in quality, decisions, coordination time, communication patterns, delegation of authority, as well as the frequency of occurrence of exceptions. The simulation generates emergent, aggregate project behaviour as actors carry out direct work, coordination work, and rework.

2.8.3 System dynamics models of PD

Analysis of PD using system dynamic principles to model the stocks (backlog) and flows of workload is a subject of research predicting the potential instability of PD where the process does not converge and where the workload can actually increase as time goes on (Ford and Sterman 1998; Mihm, Loch et al. 2003). The delays in information transmission and the rework caused by changes in interdependent processes are among the causes of oscillations in workload. A mathematical analysis of workload in PD using the DSM and system dynamics applied to a case study in the automotive industry showed the value of this approach in providing operational insights that explicitly captured the fundamental characteristics of a development process (Yassine, Joglekar et al. 2003). This analysis showed that accelerating the synchronization frequency for the specific tasks that contribute the most to slow convergence of the development process was more effective than accelerating the frequency of synchronization for all the tasks. The use of system dynamic methods to analyze collaborative PD systems points out the effects of problematic areas and possible coordination mechanisms that can be used to mitigate them.

2.9 Communication and system level integration

One type of collaborative PD is a series of multi-phase design processes linked via the original equipment manufacturer (OEM) which is acting as the system-level integrator of the various sub-systems. The role of a system level integrator is to ensure that the design work being done by development teams on the various subsystems functions together correctly as a product. The system level integrator transmits requirements to and receives information and requests for information from local level development teams (Yassine and Braha 2003). Communication

directly between local level development teams can also occur and is often informal and unscheduled. The rate at which this information is received, processed and transmitted between the system level and the local teams is an important element in the rate of progress of the overall design process (Mihm, Loch et al. 2003; Yassine, Joglekar et al. 2003). Drawbacks of this scheme are the tendency for communication channels to become overloaded and thereby cause delays in the progress of the project and decisions being taken by default by actors who lack requisite information usually obtained through hierarchical channels.

Information management in large scale engineering design is difficult and challenging for a variety of reasons as described by Eckert, Clarkson et al. (2001): diversity of channels, scale, variety of perspectives, and uncertainty. Evidence of a multitude of influences on the lack of coordination that are manifested in an apparent communication problem is discussed in a study by Maier, Kreimeyer et al. (2008). There, we see that the interaction of communication with planning and product complexity leads to wasted effort and rework in the case of automotive product development. Apparently, there is more to communication than information flow.

2.10 Models of product development

Wynn (2007) has developed a useful framework to organize the wide range of published models of the design and development process. The models are classified as using abstract approaches, procedural approaches, or analytical approaches to modeling and/or improving the design process.

Abstract approaches describe the design process at a high level of abstraction. This research is relevant to a broad range of situations, but does not offer specific guidance useful for process improvement. The abstract models can provide useful insight however, since they are concerned with the iteration which is ubiquitous in design.

Procedural approaches are more concrete in nature and focus on a specific aspect of the design process. They are less general than abstract approaches and can be

more relevant to practical situations. These are further classified by Wynn (2007) into design-focused approaches in support of the generation of better products by the application of prescriptive models and methods to the design process (Pahl and Beitz 1996), and project-focused approaches which advocate methods to support or improve management of the design project, project portfolio or company (Pugh 1995; Hales and Gooch 2004).

Analytical approaches are used to describe particular instances of design processes. They consist of two parts: a modeling framework used to describe aspects of a process; and techniques, procedures or computer tools which make use of the representation of the process to support investigation or improvements to that process. These approaches aim to provide insights and advice that may be used to plan a design process and to guide daily decisions made by design managers. Wynn (2007) goes further to classify models as task-network, queuing, multi-agent, and systems dynamics. He also describes the models as being activity, information, or actor focused.

The model used in this dissertation falls under the analytical approach in this classification of product development models in that it attempts to provide insights that may be used to plan, analyze and thereby improve a product development process with a particular level of uncertainty and task decomposition. As described in chapter 3, the model represents a task network with queuing, and is information focused with activity and actor focused components.

3 Description of the Model

In this chapter the creation of a product development model is described in detail. To begin with, precise definitions of the terms product development, process, model, system and simulation are given.

3.1 Definitions

3.1.1 Product development (PD)

Product development is an undertaking comprised of the various multi-functional activities done between defining a technology or market opportunity and starting production. The goal of PD is to create the 'recipe' for producing a product (Reinertsen 1999). Thus, the output of product development is not products, which are physical objects, but rather information. The distinction between information and objects is important. Information is the product of PD and it is what is being 'manufactured' in the sense that it is partially developed information that flows from task to task as the information 'product' is being made in the PD process. Thus, in an analogy to the analysis of material flows in a manufacturing processes, we analyze the information flow in the PD process with the caveat that a PD process has some important differences. In manufacturing, flawlessly doing the same thing twice in a row is a success; on the other hand, flawlessly doing the same thing twice in a row in PD is a failure since duplicating a recipe adds no value. Whereas many business processes seek an identical result repeatedly, PD seeks to do something new, once; PD involves creativity and innovation and is nonlinear and iterative (Kline 1985).

The information required by and exchanged between design tasks in PD can take different forms. However, here information flow is considered to consist of the following types of information:

- design information which can be specified directly, e.g., materials and geometry,

- performance information which is a consequence of design information, e.g., fatigue life and weight, and
- requirements which may constrain design or performance information, such as the requirements that the geometry of a component has to fulfil to meet the overall performance of the whole product.

This follows the approach of others that have worked on analysis of information exchange in design processes (Krishnan, Eppinger et al. 1997; O'Donovan, Clarkson et al. 2003; Bhuiyan, Gerwin et al. 2004).

3.1.2 Process

A process is "an organized group of related activities that work together to create a result of value" (Hammer 2001). It is a sequence of interdependent and linked procedures which at every stage consume one or more resources to convert inputs into outputs[2]. The work on any project follows a process.

Product development (PD) is a complex process that is inherently iterative and has significant uncertainty (Browning, Fricke et al. 2006). In PD there is uncertainty about what the initial requirements leads to in terms of technical requirements, about whether or not a proposed design solution performs as required under all conditions, about which activities should be done in order to validate the design solution, how much time and effort performing these activities are required, etc.

Unlike typical business or manufacturing processes, PD involves iterations of invention or innovation and validation, and as such it is not a linear series of prescribed steps that if followed infallibly lead to the required result. However, there is some repeatable structure or pattern to the process that has been observed, as described in 2.3 above. Any work done to produce a result has a process (the way the work gets done), though perhaps not a process model (an abstract description of the way the work could or should get done).

[2] http://www.businessdictionary.com/definition/process.html

Processes could be regarded and treated as systems in order to be able to engineer them purposefully and intelligently, facilitated by useful models (Pajerek 2000).

3.1.3 Model

A model is an abstract representation of reality that is built, verified, analyzed and manipulated to increase understanding of that reality. Models can be tacit mental models or be codified (Ford and Sterman 1998). "All models are wrong, but some can be useful" (Box 1979). A useful model can provide insights usually only available through costly experience, and it is helpful in making predictions and testing hypotheses about the effects of contemplated actions in the real world, where such actions are too disruptive or costly to try (Browning, Fricke et al. 2006). Here, we are interested in a model that can help represent, understand, engineer, manage, and improve complex PD processes.

3.1.4 System

A system can be defined as a collection of elements that act and interact together toward the accomplishment of some logical end (Schmidt and Taylor 1970). Elements in systems possess certain characteristics or attributes that can take on logical or numerical values. Typically, the activities of these elements interact in time and cause changes in the state of the system. The state of a system is in principle defined by the values of the collection of all of the variables that describe the system at a particular instance. For example, if the system that we were observing was a bicycle, we might say that its state could be defined by the rotational velocity of the wheels or the vector defining the velocity of its center of mass. We could also add the temperature of the bicycle or its position on a map into a description of its state. Its state could be further described by adding variables describing the viscosity of the lubricant in the wheel bearings or the number and type of each of the microscopic impurities in this lubricant. A complete description of the state of this system would include many other variables on the microscopic, and smaller scales.

In practice, the state of a system is modeled by defining variables whose values can approximate the state of the system at any time relative to the objectives of the study. The collection of elements that comprise a system of one study might only be a subset of the overall system of another. Examples of systems are machines, biological organisms, ecosystems, companies, hospital emergency wards, etc. A basic tenet of systems engineering is the interconnectedness between the various elements of the system and the necessity to consider these connections when seeking to make changes to the system's operation or design.

Thus, we recognize that there is a process being performed when developing products and we regard this process as a system of elements that interact together to create the information needed to produce the product. The elements in this product development system are the resources, tasks, and created information. Presented in this thesis is a useful model that describes the behavior of a product development system in sufficient fidelity to predict the dynamics of the flow of information and resource use such that the span time and effort of a development project may be calculated for a given scenario.

3.2 The product development process as a system

A PD system is complex and its state changes with time. The changes of state variables are often stepwise. For example, the receipt of information in a particular task can trigger a change in the state of the task from idle to working; the making of a decision can trigger a beginning of work in a series of tasks; the completion of one task triggers the *event* that a resource stops working on one task and begins another. Since these changes happen in very short time frames, they can be modeled as if they occurred instantaneously and that the system changes state in discrete steps triggered by specific events. Some other changes in state of the system are triggered more gradually; for instance, the completion of a task occurs when the amount of work done has reached a certain value which is approached in a piecewise continuous fashion over time. The task completion event may trigger a stepwise change in another variable.

In complex PD systems, it is evident that there is not always the same consequence or new state evolving from any given state. For example, in one particular instance a development team may require 1000 man hours to complete a specific analysis task of the aerodynamic performance of a wing design. Under the same conditions, but on another occasion, the task completion may not occur at 1000 man hours since the same team may require a different number of man-hours to complete the task. This may be due to many factors, such as subpar or exceptional performance of one of the members of the team for a certain period of time that affects the performance of the whole team, or some imprecise information being received about the design parameters of the product, which is later corrected, but causes some rework of the analysis to be done. The PD system as we define it does not include what occurs outside the boundaries of the system (for example, the personal activities of the personnel participating in a PD process), but may affect their performance within the system, and thereby, the performance of the system. Since we cannot be sure of the precise consequent state of the system given its current state, we classify the PD system as *stochastic*.

We can however make some assumptions about the probability of the consequent state of the system based on observations of PD processes or other relevant processes. For example, we may observe that under normal conditions, the time required to perform the above mentioned analysis requires about 1000 man-hours. We may also have observed that the analysis never requires less than 750 man-hours and never more than 1250 man-hours. Without any further information we can model the effort required to perform this analysis at any instance with the use of a triangular probability density function (PDF).

In general, a PDF describes the relative likelihood for a given value of a random variable to occur. In our example, the effort in man-hours to perform the analysis in question is a random number. We cannot predict precisely at any instance how long it will take to perform the analysis, but we can say what are the minimum, maximum, and most likely amounts of time required. With this information and the use of the triangular PDF we can choose a value from a sample of the

triangular PDF for each particular instance the model of the analysis process is simulated. In effect, we use the inverse PDF to obtain a value for a particular trial or simulation of the analysis because we have the knowledge that over many trials, the trials follow the triangular PDF with the particular minimum, median, and maximum values we had specified initially. The inverse PDF, also known as the inverse probability integral transform or inverse transformation method, is a basic method for generating numbers at random from any probability distribution (Devroye 1986).

Due to observations of a particular behaviour we can create a stochastic model of how a state variable in a system changes as a result of the passage of time or an event. Then, by allowing the modeled system to evolve from one state to the next under these assumptions of probability, and repeating this process many times, we can observe the statistics of how the modeled system behaves and draw conclusions about how this behaviour is affected by various conditions.

3.2.1 Dynamic systems and simulation

Systems whose state changes with time are called dynamic systems. If we were to know the value of all of the variables defining the state of a system at a particular time, and the evolution rules by which each of the variables changes from one state to the next, we may be able to predict the state of the system as it evolves from state to state until some point in the future.

The evolution rules for state variables in a PD system are not linear with time, but are greatly influenced by the values of other state variables at the current state and in previous states. This is because of the interdependencies between tasks in a PD process. So, in order to predict how the state variables of such a system evolve with time we need to simultaneously solve for the evolution of the state variables as time progresses. Furthermore, as described in the previous section, since the consequent state of the PD system given its current state is stochastic, we must solve this system multiple times in order to get a statistical sample of the results we wish to examine. This sample must be sufficiently large to give results within

the confidence limits we impose (a more detailed description of confidence limits is given in section 4.1).

Simulation is a technique by which a calculation of the state variables can be done starting with an initial state. From the initial values, the evolution rules of the system (which must be previously derived as part of the model of the system) are used to calculate the value of the state variables of interest in the next state. For anything, but the simplest of systems, this calculation is laborious and is one of the reasons that non-linear systems are often simplified into linear systems for which closed form equations can be formulated and solved. With the use of computers this technique can be automated and the changes in state of non-linear stochastic systems can be simulated over long periods of virtual time in reasonable amounts of real time.

Simulation methodology is similar to deductive theory development in that outcomes follow directly from assumptions made, but without the constraint of analytic tractability (Harrison, Lin et al. 2007). It also resembles inductive reasoning or empirical analysis in that relationships among variables may be inferred from analyzing output data (Carley and Lin 1997), even though this data is obtained from simulations rather than real world observations. Simulation can show that the process modeled can produce certain types of behaviour (Lant and Mezias 1992), can discover unexpected consequences of the interaction of simple processes (Carroll and Harrison 1994), can explain the mechanisms causing observed behaviour (Mark 2002), and can be used to suggest a better mode of operation of a process.

Of course, the question of how the simulation relates to real world behaviour must be addressed. This is referred to as model grounding, and there are several possibilities discussed in the literature (Harrison, Lin et al. 2007). The model's processes could be based on empirical observation, either the functional forms or the parameter settings or both. In many cases empirical estimates are not available, but empirical work can still provide much information for model

construction, and variations and sensitivity analysis can be used to examine the robustness of the results. Empirical grounding can also be established through the results of the simulation, either through comparison with observations of the real systems, or to serve as a basis for subsequent observations of these systems. Simulation can also be a valuable research tool even if the outcomes cannot be assessed empirically. This is a form of discovery and is characteristic of much theoretical work in both the natural and social sciences (for example, the prediction of the existence of the neutrino by W. Pauli in 1931 using theoretical methods, although there was no realistic means of observing this particle at the time).

3.2.2 Discrete event system models

A discrete event model describes a system with a high level of abstraction and with a continuous time base. In these models, during a bounded time span, only a finite number of relevant events occur. These events can cause the state of the model of the system to change, whereas in between events, the state of the modeled system does not change. Thus, in these models, time 'jumps' from one discrete event to the next. This is unlike continuous models where the state of the system may change continuously over time.

For many systems a discrete event model is appropriate for realistic representation of the system's behaviour. An example of a simple system depicted in Figure 3-1 can be used to illustrate the important concepts. At a physical level the system consists of a cashier serving arriving customers one at a time. Customers wait in a line (queue) if the cashier is not available (serving another customer). In the model, the state of the system is described by the state of the queue and the state of the cashier. The queuing rule is first-in-first-out (FIFO) and individual customers are assumed not to have any distinguishing features (such as what or how many items they actually buy).

The state of the queue is modeled simply by its length, i.e., the number of customers in the queue, an integer. The cashier can be in a busy or idle state. The

dynamics of the system is determined by: the arrival pattern of customers as described by the PDF of their inter-arrival time; a PDF describing the time required by the cashier to serve a customer; and the logical sequence of customers progressing through the system under different conditions. For instance, if a customer arrives and the cashier is idle, the processing of the customer's order begins immediately; otherwise, the customer waits at the head of the queue until the cashier is not busy. If there are other customers waiting in the queue, the new customer gets behind the last person in the queue and advances in the queue until the customer arrives at the head of the queue. Thus, the evolution rules of the system from state to state can be described as logical expressions.

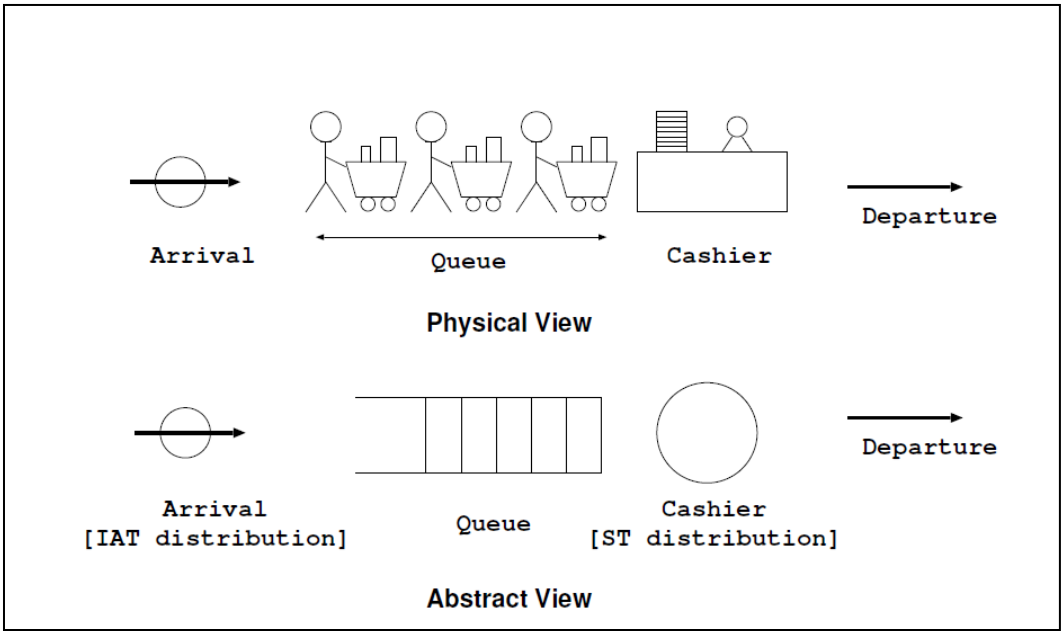


Figure 3-1. A single server queuing system

A simulation model has a *static structure* and a *dynamic structure*. The static structure specifies the possible states of the model. The dynamic structure specifies how the state changes over time. The static structure is usually described as a collection of objects and their attributes. There are different approaches, known as *world views*, for representing the dynamic structure of a model. The following concepts are at the basis of the different world views (Cota and Sargent 1992).

- An *activity* is the state of an object over an interval.
- An *event* is a change of object state, occurring at an instant, that initiates an activity not permitted to occur prior to that instant.
- An event is *determined* if the condition for event occurrence depends exclusively on system time. Otherwise, the event is *contingent* (dependent on system conditions).
- An *object activity* is the state of an object between two events describing successive state changes *for that object*. Other events may occur, related to state changes of other objects.
- A *process* is the succession of states of an object over a time span. This is equivalent to the contiguous succession of one or more object activities.

In the event scheduling world view, a model describes for each of the events the event's effect on the state and on the future behaviour of the system. This is achieved by scheduling new events into the future.

An event scheduling simulation uses two (global) data structures. One contains the state variables declared in the model. The other contains scheduled event notices in an event list, ordered by increasing time and priority. When scheduled, events are added to the queue at their appropriate time. Priorities are used to choose between events occurring at the same time (collisions). The state variables may be augmented by additional performance variables for calculation of minima, maxima, mean, standard deviation, etc. of state variables and combinations of them. Event scheduling operates by ordering (according to increasing time) scheduled events in the event list and iteratively removing and processing the head of that list until the list becomes empty. The event time of the event notice is used to advance the simulation time. Depending on the event type of the event notice, the appropriate event notice is invoked. This routine may modify the system's state and schedule new events into the future by placing event notices in the event list. As an example, part of the evolution of the state and event list during a typical event scheduling simulation of the cashier/queue model is shown in Figure 3-2.

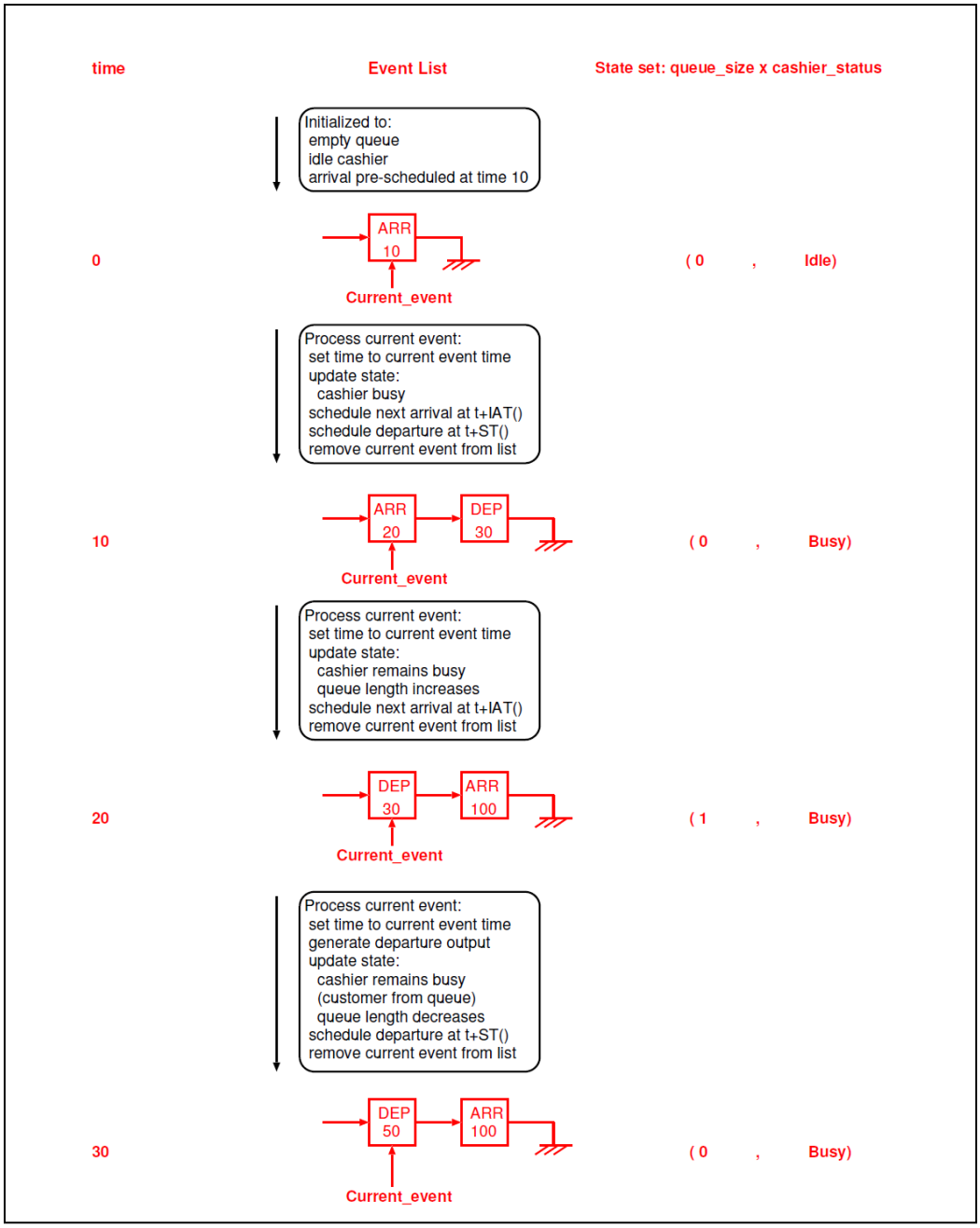


Figure 3-2. The event scheduling simulation for the single server queuing system in Figure 3-1 (Vangheluwe 2008)

An event is described by the time at which it occurs and by a type, indicating the logic that is used to simulate that event. In a discrete event model, the event logic is parameterized, such that the occurrence of one event during the simulation can create another event of type chosen during the course of the original event itself.

Typically, events are scheduled dynamically as the simulation proceeds. This means that the occurrence of an event can create another event at a later time if, for example, the value of one of the state variables set by a third event reaches a certain value.

Single-threaded simulation engines based on instantaneous events have just one current event. In contrast, multi-threaded simulation engines and simulation engines supporting an interval-based event model may have multiple current events. A processing thread in a simulation usually determines the sequence of events simulating the behaviour of a component or an actor in the model. For this reason, the processing thread in a simulation is also referred to as an *entity*. In the simulation model developed here, there are entities that represent units of information and entities that represent design tasks. Each of these entities follow a unique execution thread in the simulation and will be described in detail in the next sections.

In order to simulate a process that requires a resource, a processing thread must associate itself with a resource object for the period of time required by the process. In a multi-thread simulation, since a resource cannot simultaneously perform two processes, the instance of the resource object must be exclusively associated with one processing thread for the required period of time. Henceforth, we refer to this as a processing thread 'seizing' a resource. Once the process time has ended, the processing thread 'releases' the resource, thereby making it available for other processes. If a processing thread or entity must seize a resource before any other processing steps and the resource is occupied, it waits. As illustrated in the single server example, in order to manage multiple entities waiting for a resource, the concept of a queue is employed. Each entity is given a priority attribute according to the queue rules and the entity with the top priority is the one that seizes the resource when it is next available. The priority attribute is set according to the rules for the queue (such as first-in-first-out, or last-in-first-out for example) in accordance to the situation being modeled.

3.3 The model of the product development system

In the previous sections we have seen that the PD process can be thought of as a system that is dynamic and stochastic, and that it can change state in a stepwise fashion with the occurrence of discrete events. As described in 3.1.1 the fundamental dependency between tasks in PD is the information generated by some tasks that is required by others.

In PD, the process is completed when all of the tasks in the process have been completed. For a given level of effort and a given number and choice of resources, the manner in which the work is organized among the resources is what determines the amount of time required to complete the process. It is the "relationships among elements of a system that give systems their added value... the greatest leverage in system architecting is at the interfaces" (Rechtin 1991). In a process with dependencies between tasks, for a given level of effort and a given number and choice of resources, the span time of a total process can be reduced only if the dependencies between the tasks are better managed. Since the dependencies between tasks in PD are information, improvements in the span time of the process can be achieved by improving the information flow between tasks.

In order to meet the objective stated in chapter 1 "develop a model of PD that can be used to capture the impact of various coordination mechanisms on reduction of span time under all potential scenarios," we want to capture the details of the information flow in a PD project that are affected by various coordination mechanisms. The model we build should consider the fundamental dependencies between tasks and capture the impact of the coordination mechanisms used to manage these dependencies on PD process span time. These coordination mechanisms, outlined in Table 2-2, deal with shared resources, various consumer/producer dependencies (prerequisite, transfer, usability), simultaneity, and task/subtask. Efforts to coordinate the PD process can be judged according to how they facilitate the information flow between tasks that must be accomplished to complete the process. It follows then that we must model the details of the

information flow between individual tasks and capture how the coordination mechanisms employed affect this flow in order to meet the modeling objective mentioned above.

Information develops during the work carried out in each task, and the attributes of this information with regard to quality, precision, and stability evolve as the task progresses[3]. Interim information that is communicated carries with it these information attributes that are related to the state of the work in the task that created it. This information, when used by other tasks has an effect on their progress. If information is imprecise or unstable and if it is used as a basis for further work or decisions in the process inappropriately, the arrival of contradictory information may trigger rework of some or all of the work done based on this updated information.

Other researchers that have studied concurrent engineering with models have considered the importance of the attributes of interim information, calling it the rate of evolution of information (Krishnan 1996; Bhuiyan, Gerwin et al. 2004), maturity of information (O'Donovan, Clarkson et al. 2003), ambiguity and uncertainty of information (Terwiesch, Loch et al. 2002). Here, we wish to consider more detailed aspects of information exchange in the process in order to study the characteristics of the PD process that may influence them. As such, a more detailed model of the information exchange in PD process is required.

A task based approach for process modelling is used in this work. This choice was made because different tasks may be used to perform the same function and each task is represented as a logical sequence of events through which cyclic iterations can be made. Such a model allows multiple tasks to be defined, and offers a method of representing the appropriate context in which each task may be used by decomposing the tasks into subtasks. The task based model allows the link between information flow characteristics, functions, and objectives, e.g., the fulfilment of product performance parameters.

---

[3] Quality, precision, and stability are introduced in sections 2.5 and 2.6.

An overview of the model is first described in the following section. Further information in increasing levels of detail can be found in the subsequent sections of this chapter.

3.3.1 Overview of the model

The PD process is modeled as a series of tasks that are performed by finite resources representing development teams[4]. Each of these tasks is executed in a discrete event simulation to generate information that is required to complete the 'recipe' for producing the product[5]. The performance of a task requires effort by the resources assigned to it and this is simulated by the exclusive association of the processing thread representing the technical work of the task with a specific resource for the required period of time, i.e. the processing thread or '*work* entity' seizes the resource. Discrete units of information, each of which is a unique processing thread (called an information entity), are created in the discrete event simulation as effort is expended in each task. The number of tasks in a project and a probability distribution defining the amount of effort required by each assigned development team to perform its task are inputs for the particular scenario being modeled.

The tasks in the modeled PD process are grouped into phases. Once the tasks comprising a phase are completed, a design review takes place in which the work completed until that point is evaluated and a decision is made whether to continue to the next phase or to redo previous phases (Figure 3-3). This decision is modeled in the discrete event simulation as a Bernoulli trial[6]. The probability of a positive decision is based on the simulated performance data generated by the

---

[4] We use the word 'task' interchangeably with the word 'activity' that is used by other authors to describe the various steps that are taken in a process

[5] See section 3.1.1 for further description of the product development process and section 3.2.2 for further description of discrete event system models.

[6] A Bernoulli trial is a single experiment whose outcome is random and can have either of two possible outcomes; one outcome that can be called success and the other outcome failure. The probability of success can, for example, be ½ in a fair coin toss, or can be set as a given probability p.

execution of the tasks in the previous phases. The number of phases and design reviews is an input for the particular scenario being modeled.

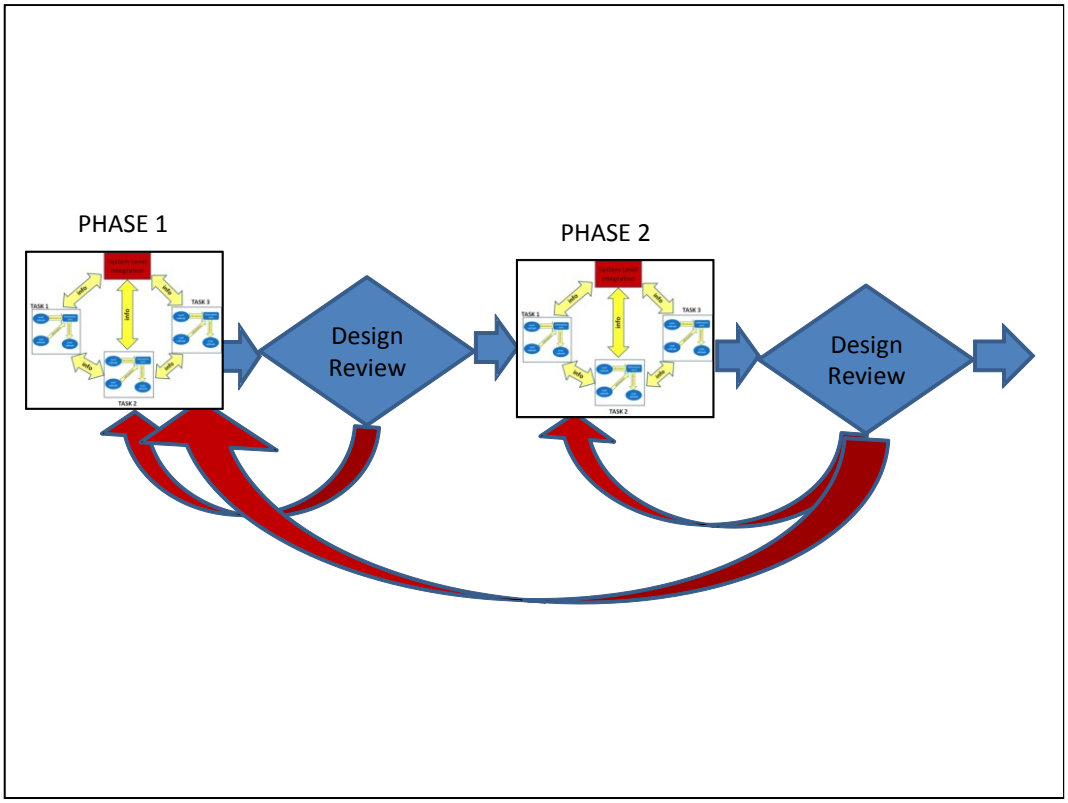


Figure 3-3 The process model showing design review decisions at the end of each phase

Groups of discrete units of information (information entities) representing the information developing in each task are periodically prepared for communication. This requires effort on the part of the resource performing the task. The frequency of preparation and communication of information is an important parameter used in the model. Once prepared, the information is sent to other tasks.

The time required for information to travel to its destination is composed of transmission time and latency time. Transmission time is related to the type of communication method used, and latency is attributable to delays in getting information through the layers of the organization to the people requiring it. The actual amount of time required in each instance of communication is determined randomly based on unique normal distributions for each type of delay and for each task.

As will be detailed further in the next sections, the number of information entities created in one task that are to be sent to another task is related to the dependency strength between that pair of tasks. The strength of the dependency between each pair of tasks is an input for a process being modeled. In practice, these dependency values are based on evaluations of the initial uncertainty of the output of one task and sensitivity to changes in input of the second task for each pair of tasks.

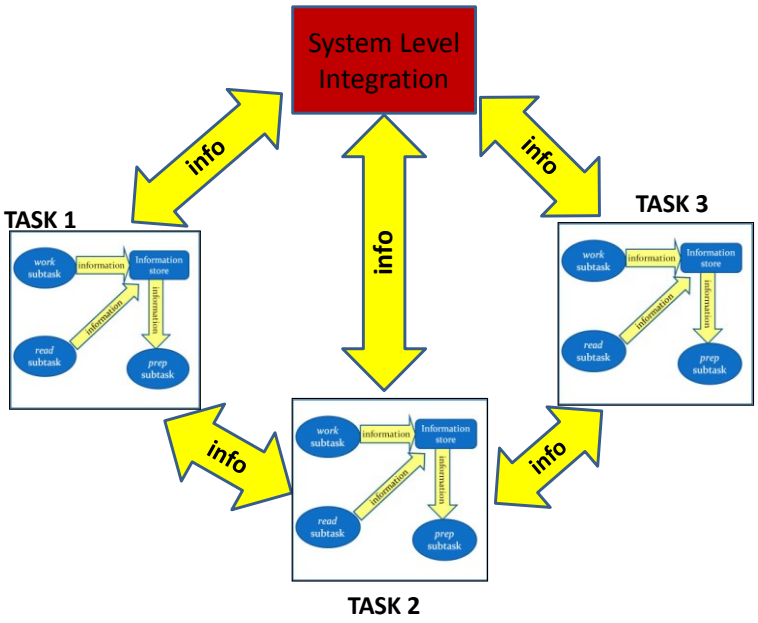


Figure 3-4 The model of a phase of the product development system

The system level integrator function[7] is modeled by first routing the information being communicated by tasks to a process step that reads and evaluates the information for its quality (Figure 3-4). This process step is assigned a resource representing the system integrator. As will be detailed further in the following sections, insufficient quality of information results in information being rejected.

[7] See section 2.9 for further description of the system integrator function.

In the model, input information received from other tasks is read and interpreted. This requires effort on the part of the resources performing the task and takes place in the model when individual information entities seize the resource associated with the task receiving the information.

The information entities sent by a development team each have attributes for epistemic and aleatory uncertainty. Epistemic uncertainty reduces as the state of the task sending the information progresses; however, a level of aleatory uncertainty is always present[8]. Iteration in a task is dynamically generated in the model based on the uncertainty of successive communications of input information received from other tasks. Input parameters define the relationship of epistemic and aleatory uncertainty of information generated by a task as a function of its state of progress. Thus each information entity has a unique value of epistemic and aleatory uncertainty based on the state of progress of the task creating it. Rework in a task is dynamically generated in the model based on the attributes of uncertainty of successive communications of input information received from other tasks.

The model is designed to permit study of the effects of various potential improvements to the information flow of a PD process. These potential improvements can take the form of a change in the architecture of the process, or as a change in the way the execution of a specific process architecture is managed. The process architecture that can be varied is the choice of decomposition, the number and relative size of the tasks and phases in the process, the sequencing of tasks, and the choice of resources assigned to each task of the process. These changes can be reflected in the model by the input values of the number of tasks and phases, the strength of the dependency relationships between tasks, the amount of effort required to execute each task, the uncertainty reduction profiles, the magnitude of stochastic uncertainty in each task, and the degree of overlapping between each pair of tasks.

---

[8] See section 2.5 for further description of uncertainty in engineering design processes.

The improvements in the way the execution a particular process is managed can be examined with changes to the frequency of communication between dependent tasks, changes to the average time required to prepare information for communication in each task such as the implementation of methods to facilitate the production of reports and drawings, reduction in the delays due to communication transmission, reduction in delays due to organizational impediments such as slow internal mail distribution or unnecessary routing of documents, the reduction of information volume unnecessarily sent to project participants, reduction in the time required to read, comprehend, and interpret information by project participants through the use of richer communication channels such as closer proximity of development teams working on highly interdependent tasks, and better management of critical resources such as system level integrators.

In the sections that follow, the operation of the model is further detailed. Input parameters for modeling scenarios are summarized in Table 3-3 and symbols used in the model are summarized in Table 3-4 at the end of this chapter.

3.3.2 The task model

The amount of technical work accomplished and the amount of input information received in each of the tasks in the process are state variables of interest in our model. We plan to investigate how the evolution of these variables influences the aggregate output variables, process span time and effort. We therefore must define the completion of the process by the achievement of final values of these state variables; that is, we define the state variable of technical work done relative to the amount of work required to be done to complete the task, and the input information received relative to the total amount of input information required to be received from all the other tasks. If a task is able to bring these two state variables to unity, the task can be considered complete. If all the tasks in a phase are complete, then the phase is complete, and if all phases in a project are complete, the project is complete. The effort of the process can therefore be calculated as the sum of the effort expended in each of the tasks in all of the

phases, and the span time can be calculated from the start time of the initial phase to the completion time of the final phase.

Effort in the model is accumulated when a resource is occupied for a period of time either doing technical work or communication. Project participants must spend time to prepare information for communication to others by writing reports, preparing presentations, coordinating and attending meetings, etc. Additionally, when receiving information project participants must spend time to read, comprehend and interpret it. Since project participants cannot do technical work while they are occupied with communication work, communication effort has significant impact on the progress of the technical work and the progress of the process.

In the model, we therefore split a task into three subtasks: *work* which is comprised of the technical work required to complete the task; *read* which is the time spent by the resource performing the work required to read, interpret and comprehend incoming information; and *prepare* which is the time spent by the resource in preparing information developed in the task for communication (Figure 3-5).

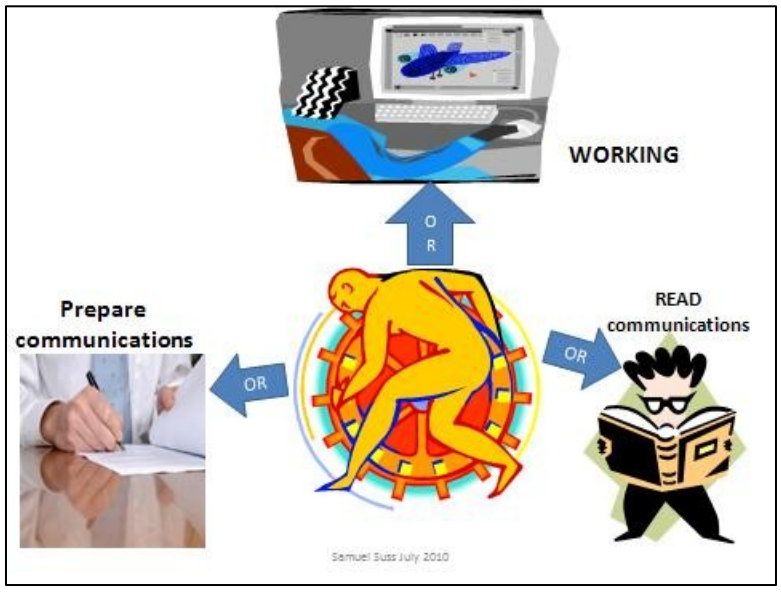


Figure 3-5 Task model broken up into work and read and prepare communication subtasks

We model each of these subtasks as a stochastic process so that the amount of time a resource needs to perform each of them is chosen from an inverse triangular PDF[9]. The total effort initially required to complete the technical work in a task is determined once the task begins, but the effort required to read or prepare each unit of information is determined randomly at the instant of each information entity's arrival by using an inverse PDF. Information communicated to a particular project team performing a task must queue if the project team is performing technical work or occupied with processing other information. Similarly, technical work to be performed by a project team must queue if the project team is occupied with processing information. As we shall see later, the total effort required to complete the technical work in a task can increase during the performance of the task due to rework that may be generated.

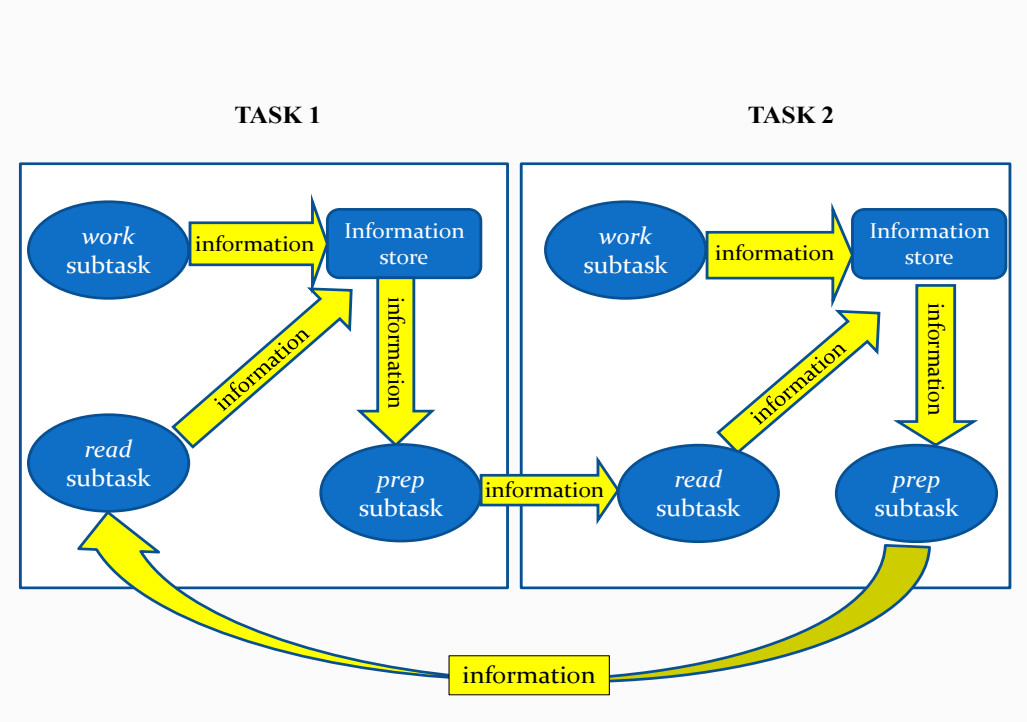


Figure 3-6 Links between tasks via information exchange

[9] Inverse PDF is defined in section 3.2

When execution of a task begins, the task entity seizes the allocated resource to perform *work*. The task entity releases the resource associated with the task periodically to perform communication work or because it must wait for further information. We call the period between the time the task entity seizes the resource and releases it the *work cycle*. As shown in Figure 3-7, the timeline of the task is broken into work cycles of period Δt [10] between which there are periods of communication.

At the beginning of each new work cycle the uncertainty attributes of the input information received in the previous communication period are evaluated. From these attributes the average input uncertainty of information received by each task from each other task in the project are evaluated. The change in uncertainty levels of input information drives the logic of design iteration rework in the model (rework is described in section 3.3.10 and 3.3.11).

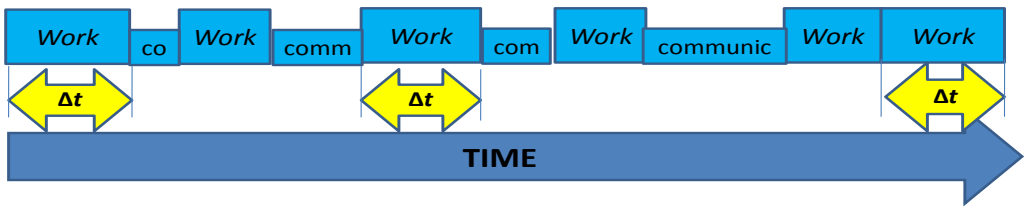


Figure 3-7 Work subtask done in discrete chunks of time or cycles between which communication is done

3.3.3 The information exchange matrix

Now recall that in order to complete a task we also required that the input information received by a task must be complete. In the following paragraphs we show how can this be determined for any PD process.

[10] Δt is an input value specified for each task in the process being modeled. See Table 3-3 for a complete list of input variables.

3.3.3.1 Dependency strength between tasks in PD

We first characterize the information dependency between each pair of tasks *i* and *j* as made up of the initial uncertainty in information input to task *j* from task *i* and of the sensitivity of task *j* to changes in this information input (Yassine, Falkenburg et al. 1999). The sensitivity of a task *j* to changes in information input from task *i* is a measure of how the final results of task *j* are affected by changes to information input from task *i*. The initial uncertainty in information input to task *j* from task *i* is the possible deviation of an initial estimate from the actual value. Task *i*'s information to task *j* is highly uncertain if the team working on task *j* is incapable of guessing a value or range of values for the output of task *i*.

Yassine, Falkenburg et al. (1999) constructed the discrete subjective measurement scales for sensitivity and uncertainty in a design context shown in Table 3-1 and Table 3-2 respectively. A different set of attribute levels and descriptions can be tailored for a specific design situation; however, those shown here illustrate the meanings that are attached to each discrete level of each attribute in a structured interview process with experts in the specific design situation and are used in the model developed here.

Now, we define a project's information dependencies using a dependency structure matrix (DSM) **D** whose elements are defined as follows:

$$Dij = Uij * Sij \tag{3-1}$$

where Uij is the initial uncertainty of the information input from task *i* to development team *j*, and Sij is the sensitivity of task *j* to changes in information input from task *i*. Using values of 0, 1, 2, and 3 to represent the levels of uncertainty and sensitivity of none, low, medium and high for Uij and Sij (Tables 3-1 and 3-2), the elements Dij can take on values of 0 through 9.

Table 3-1 A constructed attribute for sensitivity *S* of task *j* to changes in information from task *i*

Attribute level	Description of attribute level
0	Weak. Changes in the information produced by task *i* are practically irrelevant.
1	Not vital. A major part of task *j* can be performed with initial estimates of information from task *i* without danger of significant rework.
2	Vital. Task *j* can be started with initial estimates of information from task *i*, but substantial rework may be required if final values vary by more than expected.
3	Extremely vital. Any change in estimated information from task *i* requires rework for anything done in task *j*.

Table 3-2 A constructed attribute for the initial uncertainty *U* of information required from task *i* by task *j*

Attribute level	Description of attribute level
0	Definite. Information produced by task *i* is relatively certain.
1	Confident. Information from task *i* can be identified as highly probable.
2	Uncertain. An interval of values of information from task *i*, can be identified, but there is no way to conclude which value is more likely.
3	Extremely uncertain. It is not possible to identify any limits on the uncertainty of the information from task *i*.

Dij is referred to as the dependency strength of task *i* on task *j*. It is defined as the product of Uij and Sij because the overall strength of a dependency link depends directly on the level of uncertainty, where higher uncertainty means higher dependency, and on the level of sensitivity, where higher sensitivity means higher dependency.

3.3.3.2 Relation of the dependency between tasks to information exchange

Now, we further hypothesize that the total amount of information that must be communicated between interdependent tasks is directly proportional to the dependency strength Dij. If the PD project is split into differentiated tasks the information that must flow between each pair of tasks is related to the way in which the tasks are split up. If there is no dependency between a pair of tasks, there is no need for any information to be exchanged between them. If the tasks are dependent, there needs to be information exchanged.

This can be understood by considering that, if there is greater uncertainty in an activity's output, it is likely that more estimates of the output information need to be generated and communicated to downstream activities before the design activity is completed. Similarly, if there is greater sensitivity to another activity's output, it is likely that more information needs to be transferred before the linked activities arrive at a jointly satisfactory solution. For each increased level of uncertainty, the effect of sensitivity is magnified and vice versa; so, we assume that the effects of these characteristics are multiplicative as indicated in Equation 3-1.

Imagine that for a given product development project there is a certain total amount of information that must be generated. Now, consider that this project is decomposed so that the work required is split among *NPT* development teams, each of which performs a differentiated task. We consider that in order to perform the work, there are dependencies between the tasks that must be managed and that the dependencies are due to the information generated by some tasks that is required by others.

The information generated by each task that is required by dependent tasks is in the form of values of design parameters, performance requirements, or design performance. If we consider that these values are each contained in 'pieces of information' or discrete information units, then we can estimate the number of these discrete units that must be communicated by each task to every other task in

the task decomposition. Consider that task *j* is dependent on task *i* for information and that task *i* is likewise dependent of task *j* for information. The number of discrete units of information required to be communicated from *i* to *j* is proportional to the number of dependent items assuming that each unit contains information at least about one dependent item such as a single design parameter or requirement.

Now, recall that uncertainty is the difference between the information required to accomplish a task and the information currently residing with the actor charged with performing it (Galbraith 1977). That uncertainty of a task affects the communication requirements in performing the task can be understood from the following example. Consider as a task, the development of an aircraft subsystem, e.g., a previous example, a landing gear assembly. The shape, weight, and power requirements of the subsystem may have an effect on other components of the aircraft. For example, the geometric shape of the landing gear housing has an effect on the exterior contour of the aircraft, which has an effect on the aerodynamic loads on the aircraft. These loads have an influence on the structural design of the aircraft which has an effect on its weight. The weight of the aircraft in turn has an effect on the landing gear design and in particular the structural design and size. Thus, the design of the landing gear is reciprocally dependent or interdependent with the design of the exterior contour of the aircraft.

This interdependence is managed in aircraft design by a progressive process that generates intermediate outcomes[11]. The process begins in a preliminary phase where the overall size and exterior shape of the aircraft is designed based on the major requirements of the aircraft mission profile (payload, range, cruising speed). Using estimates for the designs of the subsystems of the aircraft that have to be contained within this exterior shape, aerodynamic design, stability, and other flight science considerations enable an exterior model of the aircraft to be created and optimized. In the subsequent phase, a more detailed design of the major

[11] See section 2.4 for a discussion of progressive processes and iteration in engineering design.

subsystems and components can proceed based on the requirements imposed by the external shape acting now as an envelope for the geometry of these subsystems, and imposing a requirement on weight.

The estimates of the designs of aircraft subsystems and components that are used in the first phase of the aircraft design are based on the knowledge that the organization designing the aircraft has of these subsystems. This is generally based on previous designs of similar systems, either from experience or trade studies. Uncertainty is present if the organization is unable to find the relevant information on particular subsystems or if the technology the designers wish to employ in one or several subsystems is new and there is little or no experience with it.

If the design of some components have uncertainty, it is likely that the initial parameter estimates require modifications and a rebalancing is required as the development of each component progresses. This requires more communication between the development teams. Alternatively, if all the subsystems and components have been successfully developed before and there are only minor changes to their design, it is likely that the initial estimates of their characteristics do not require much change and the communication load is lower. Therefore, the number of communications that is required is also a function of the initial uncertainty.

Finally, there is the issue of sensitivity. How sensitive is the work of task *j* to changes in the information provided by task *i*? If a parameter value generated by task *i* is subject to change, but task *j* is able to complete its work with the earlier estimates of this parameter and its output is affected only slightly by changes in the value of this parameter, the dependency of task *j* on this 'piece of information' from task *i* is low. Thus, for each level of uncertainty of a parameter from task *i* the number of communications required to be received by task *j* is modified by the sensitivity of task *j*.

Thus, we reason that for each 'piece of information' or value of a design parameter, *n*, the number of communications from task *i* to task *j* is proportional to the product of the initial uncertainty in this information U_{nij} and the sensitivity of task *j* to changes in the value of this parameter S_{nij}. Therefore, the total number of information units NC_{ij} that must be communicated from task *i* to task *j* can be expressed as:

$$NC_{ij} = \mathbf{ANINT}\left[C_L \sum_{n=1}^{P_{ij}} U_{nij} \times S_{nij}\right] \qquad \forall\, i,j = 1,2,\ldots NPT, i \neq j$$

3-2

ANINT is the operator that finds the nearest integer of the quantity in the brackets. C_L is a scaling factor which relates the number of information units that must be communicated to the numerical scale chosen to quantify the levels of uncertainty and sensitivity. P_{ij} is the number of parameters for which task *j* requires values from task *i* and NPT is the number of tasks.

If we equate:

$$D_{ij} = \sum_{n=1}^{P_{ij}} U_{nij} \times S_{nij}$$

3-3

then

$$NC_{ij} = \mathbf{ANINT}\ \left[C_L \times D_{ij}\right] \qquad \forall\, i,j = 1,2,\ldots NPT, i \neq j$$

3-4

Using a similar approach the levels of initial uncertainty and sensitivity for each dependent parameter value between a pair of tasks can be established and a square *NPT* x *NPT* matrix **NC** with elements NC_{ij} describing the number of units of information that must be communicated from task *i* to task *j* can be formed.

It has been observed in research conducted in organizations that conduct product development that the number of communications between participants in product

development projects are greater between people performing activities with greater dependency (Allen 2007). This was the case reported regardless of other factors such as physical distance between the participants or departmental membership. The results were obtained by measuring the number of communications or messages that were exchanged between people. This result supports the basis for the development of the equations to populate matrix **NC.**

The scaling factor C_L in equations 3-2 and 3-4 can be calculated by relating the total average effort in technical work to the total average effort in communication work per phase in the PD project being studied. The total effort in communication work C_w for a phase of a project with *NPT* tasks is given by the sum of the effort in preparing each unit of information in the *prepare* subtask multiplied by the number of units of information to be sent by each task plus the sum of the effort in reading each unit of information in the *read* subtask multiplied by the number of units to be received by each task. This turns out to be expressed as:

$$C_w = \sum_{k=1}^{NPT} \sum_{j=1}^{NPT} \left[\bar{P}_k \times NC_{kj}\right] + \left[\bar{R}_k \times NC_{jk}\right] \quad \text{3-5}$$

where the average effort in preparing each outgoing unit of information by task *k,* $\bar{P}_k$, is calculated from the minimum, median, and maximum parameters of the triangular PDF of the k^{th} *prepare* subtask (stored in matrix **PR**) as follows:

$$\bar{P}_k = {}^1\!/_3 \times \left(PR_{k,1} + PR_{k,2} + PR_{k,3}\right) \quad \text{3-6}$$

and the average effort in reading each unit of incoming information by task *k,* $\bar{R}_k$, is calculated from the minimum, median, and maximum parameters of the triangular PDF of the k^{th} *read* subtask (stored in matrix **RD**) as follows:

$$\bar{R}_k = {}^1\!/_3 \times \left(RD_{k,1} + RD_{k,2} + RD_{k,3}\right) \quad \text{3-7}$$

and where *NPT* is the number of tasks in each phase of the PD project.

Now, equating the average communication effort divided by the average technical effort to an input parameter $\boldsymbol{\alpha}$ and using equation 3-4, we obtain the following value for C_L :

$$C_L = \frac{\boldsymbol{\alpha} \sum_{k=1}^{NPT} \overline{WK}_k}{\sum_{k=1}^{NPT} \sum_{j=1}^{NPT} \left(\bar{P}_k \times D_{kj} + \bar{R}_k \times D_{jk} \right)}$$

3-8

where the average technical work of task *k*, $\overline{WK}_k$, is calculated using the minimum, median, and maximum parameters of the triangular PDF of the k^{th} *work* subtask (stored in matrix **WK**) as follows:

$$\overline{WK}_k = {}^1/_3 \times \left(WK_{k,1} + WK_{k,2} + WK_{k,3} \right)$$

3-9

It has been observed by practitioners in aerospace companies whom we have interviewed that the effort expended in communication is typically equal to the effort in the technical aspects of the work in the project[12]. Therefore we carried out simulation experiments with $\boldsymbol{\alpha}$ = 1, and then examined the sensitivity of the results when $\boldsymbol{\alpha}$ was varied about this value[13].

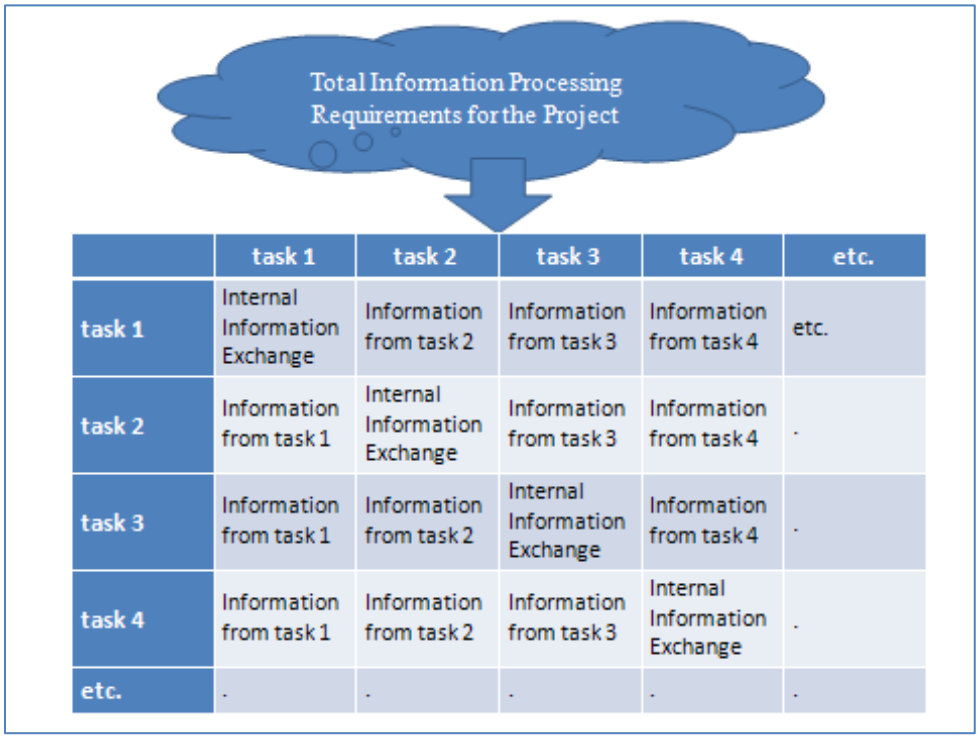
3.3.4 Diagonal elements of the information exchange matrix

Equation 3-4 describes the off-diagonal elements of matrix **NC** in terms of the dependency of information of one task on another, but does not provide any measure of the amount of communication between people working on the same task. In the model we consider that a task converts input information into output information with the performance of value-added work, and that no matter how many tasks there are in the decomposition, the information processing requirements are constant for a given value of initial uncertainty. The

[12] See Chapter 5.
[13] See chapter 4.

decomposition of the task simply serves to allocate the total information processing work among the tasks according to their dependency relationships as indicated in the previous section. Therefore, whatever information processing requirements that are not taken up by those between different tasks must remain within each task itself (Figure 3-8).

Further details on how the total information requirements in a PD project are related to the model variables are given in section 4.2.



	task 1	task 2	task 3	task 4	etc.
task 1	Internal Information Exchange	Information from task 2	Information from task 3	Information from task 4	etc.
task 2	Information from task 1	Internal Information Exchange	Information from task 3	Information from task 4	.
task 3	Information from task 1	Information from task 2	Internal Information Exchange	Information from task 4	.
task 4	Information from task 1	Information from task 2	Information from task 3	Internal Information Exchange	.
etc.	.	.	.	.	.

Figure 3-8 Schematic view of the information exchange matrix

If there are fewer tasks in the decomposition, a larger proportion of the information exchange is carried out internally in a task within the development team performing the work. The value of a diagonal element in the information exchange matrix is a measure of the information that the development team performing the task must gain from its own internal interaction in order to produce its output. In other words, it is the uncertainty gap that is not filled by input information from the other tasks in the project.

In practice, the values of each of the diagonal elements of **NC** can be estimated with equation 3-2, so that:

$$NC_{ii} = C_L \sum_{n=1}^{P_{ii}} U_{nii} \times S_{nii} \qquad \forall\, i = 1,2, \ldots NPT$$

3-10

3.3.5 Information flow in the model

As described earlier, each task in the process decomposition is composed of three subtasks: *work, read, and prepare* and the state variables of the model change with the occurrence of events. In the model, the simulation begins with a processing thread for each task. Each thread, simulating the sequence of events of a task 'entity' performs its series of steps according to the logic of the programmed algorithms. Each step is added to the pending event set (see section 3.2.2) and is processed according to its position on this list.

At the beginning of the processing of a task, the entity for each task *i* (henceforth called the task entity) triggers:

1. the calculation of the initial value of total technical work required in this task, WD_i , from the inverse triangular PDF for the task[14];
2. the creation of additional entities equal in number to the sum of the values of elements in row *i* of matrix **NC**. These entities or processing threads (henceforth called information entities) each represent a piece of information in the model. Each of these new entities is assigned an attribute called *PID* with the identity of the task from which it came and an attribute called *TO* specifying the identity of the task to which it will be sent, e.g., an entity sent from task 2 and addressed to task 4 has value of *PID*=2 and *TO*=4. This latter attribute is assigned according to the values of **NC** so that the correct number of entities NC_{ij} are addressed to each

[14] See Table 3-4 for a description of all variables employed in the model. As described in section 3.3.10, WD_i can vary from this initial value due to the generation of rework.

task j from task i. The TO attributes are assigned in random order since in the general case we do not know in which order the information is developed by a task. These information entities each then follow the logic explained below.

Each task entity waits in a queue until the state of the resource representing the associated development team is 'available'. (At the onset of the simulation, the state of the resource of each task is 'available'). Once 'available' and the task entity is at the head of the queue, the state of this resource is set to 'unavailable', i.e., the task entity 'seizes' the resource. The state variable for the work accomplished in task i, called WT_i, increases at the rate of one unit per hour until the task entity releases the resource. An interim state variable DT_i , which keeps track of the number of hours of effort expended in the current work cycle in task i, also increases at this rate. The next event in this processing thread occurs as a result of the logic in the algorithm. For example, whichever of the following events occurs first triggers the subsequent events in this processing thread.

1. The span time of task i exceeds the time allotted, SCH_i. Work in task i stops, i.e., the task entity 'releases' the resource. This processing thread continues to a series of task end events described in subsequent sections.
2. The amount of work done in a task exceeds the starve condition where insufficient input information has been received to continue the technical work in the task. This triggers the suspend work event. The task entity waits for subsequent events before continuing to process anything further. Further details of the starve condition and logic applied for subsequent events are given below.
3. The interim state variable DT_i in the task exceeds the value of model input variable Δt_i and one of the queue sizes for this resource, which holds

waiting information entities, exceeds a minimum number[15]. This triggers the event that the work in this task is suspended and DT_i is set to zero. The task entity waits in a queue until the state of the resource representing the associated development team is 'available.'

Each information entity associated with task *i* follows a unique processing thread under conditions imposed on it during the simulation that is summarized as follows:

1. creation initiated by the task entity,
2. assignment of attributes *TO* and *PID*,
3. pause until the event triggered by achieving a state of progress in the *work* subtask,
4. assignment of attributes for quality (*qlty*), epistemic uncertainty (*eps*), and aleatory uncertainty (*phi*) based on the state of the task at that instant,
5. queue and seize the resource associated with task *i* for the *prepare* subtask,
6. pause for the time representing the *prepare* subtask operation,
7. release the resource associated with task *i*,
8. pause for the time representing the random transmission delay,
9. queue and seize the integrator resource,
10. pause for the time representing the integrator operation,
11. release the integrator resource,
12. return to the originating task *prepare* queue if quality rejected, **or**
13. pause for the time representing the random transmission delay,
14. wait for the time representing the latency delay,
15. queue and seize the resource assigned to the task associated with the *TO* attribute,

[15] An input variable setting the minimum number of information entities in the queue for the resource of each task to perform the *read* subtask (QMR_i) or the *prepare* subtask (QMP_i) is used for this (see Table 3-3)

16. pause for the time representing the *read* subtask operation in the addressee task,
17. release the resource,
18. trigger the update of state variables for uncertainty, input information,
19. end the processing thread.

Information entities with address attributes *TO* of value *j* where $j \neq i$ are intended to model information communicated to other tasks in the project. Here, although we assume that information develops uniformly with progress made in the technical work of the task, information is not communicated as soon as it develops, but rather is prepared for communication and transmitted periodically as one would expect in practice. In the model, therefore, the information entities created wait until an event triggers their release to the *prepare* queue (step 3 above). This event occurs each time the ratio WT_i/WD_i reaches an integer multiple of the modeling input CI_i which defines the communication interval as a fraction of the nominal span time of task *i*[16]. Thus, when the following condition is true, the n_{th} batch of information entities is released to the *prepare* queue of task *i*:

$$\frac{WT_i}{WD_i} \geq n \times CI_i \qquad \forall\, n = 1, 2, 3, .., 1/CI_i$$

3-11

Information communicated internally within the team performing task *i* (an information entity from task *i* with an address attribute *TO* of value *i*) is communicated without delay, uniformly in relation to the progress made in the technical work of this task. This assumption is made because members of the same development team are generally located close to each other and engage in almost continuous informal communication about their common work. Thus, the

[16] Recall that the value of WD_i can increase during the simulation as a result of rework generated (see section 3.3.10).

release of each information entity destined for internal communication to the *prepare* queue takes place with equal intervals of WT_i/WD_i.

'Releasing' an information entity for communication means that it enters the queue for the resource associated with task i for the *prepare* subtask. Prior to entering the queue the attributes (described in further detail in the next section) for quality, epistemic uncertainty and aleatory uncertainty are assigned to the information entity based on the state of its originating task at that moment. When an information entity reaches the head of the *prepare* queue, it seizes the resource for the time period chosen randomly from the specified inverse triangular PDF.

3.3.6 Attributes of information entities

Each information entity has attributes carrying its quality, epistemic and aleatory uncertainty (refer to section 2.5). These are set when the information entity is released for communication based on the state of the sending task at that instant.

3.3.6.1 Uncertainty attributes

As discussed in section 2.5 the epistemic uncertainty of information generated by a task is related to the state of the task at the time it is generated. We assume that this epistemic uncertainty reduces as the state of the task progresses. This occurs because of the work accomplished on the task and because of the input information received by the development team performing the task. Since the scale for uncertainty is arbitrary we can only determine its value at any time relative to the initial estimate done in the manner described in Table 3-2. For different tasks, the reduction in epistemic uncertainty as progress is made depends on the nature of the task itself, whether it is a task that rapidly reaches lower uncertainty as work is done or not. The reduction in uncertainty can be thought of in terms of lack of precision in the value of a design parameter. Precision may reach a high value, i.e., be more precise, early and then improve more slowly to its final value, or in another task, the precision of a design parameter may improve only slowly until the task is completed.

In order to take the nature of the task's evolution into consideration in our model, we postulate a functional relationship that is sufficiently flexible to account for the type of variation encountered in practice. To this end we use a Gompertz function which yields an S-shaped curve that allows for different rates of approach to the lower and upper asymptotes at the start and end of a task.

Thus, the epistemic uncertainty of a task is related to the task state with the following equation:

$$\epsilon_{ti} = 1 - e^{\left(-b_i e^{(-c_i S_{ti})}\right)}$$

3-12

where ϵ_{ti} is the epistemic part of uncertainty at time *t* divided by the initial value of epistemic uncertainty for task *i,* and S_{ti} is the state of progress of task *i* at time *t*. Equation 3-12 is the Gompertz function subtracted from its upper asymptote (here equal to 1) and where b_i, c_i are input coefficients that define the shape of the curve. This function has some of the characteristics of an S-shape with the steepness of reduction of uncertainty controlled by the choice of the coefficients b_i, c_i for each task *i* (Figure 3-9). This calculation of the ratio of epistemic uncertainty reduction to its initial value is sufficient for our purposes since we are concerned in our model only with the changes to uncertainty during the progress of the tasks in the process.

The state of progress of a task at any time *t* is assumed to be a linear combination of the achieved technical work fraction and the received input information fraction at that time.

$$S_{ti} = \frac{1}{2}\left(\frac{WT_i}{WD_i} + \frac{I_{ti}}{\gamma_i}\right)$$

3-13

where WT_i is the amount of technical work done until time *t* by task *i*; WD_i is the total amount of technical work required to be done by this task (initially determined from the triangular PDF at the onset of the task, but as described in section 3.3.9, can increase when there is design iteration rework generated during

the simulation); I_{ti} is the amount of input information received from all other tasks until time *t*; and γ_i is the total amount of input information required to be received by task *i* from all other tasks (derived from matrix **NC**). The state of progress in equation 3-13 is calculated for each information release .

Note that the two terms within the brackets in equation 3-13 are the fractions of completion of technical work and input information respectively and are both less than or equal to one at all times during the simulation of the process.

This relationship for the state of progress of the task is used because it incorporates the two aspects of a task's progress which describes its 'nearness' to completion. Simply relating the state of a task to the amount of technical work that has been accomplished is insufficient because there may be required rework of some of the work already done. However, the combination of the proportions of work done and received input information is a better measure of the state of progress of a task. The reasoning for this is that we expect that there will be less or smaller amounts of design iteration required if a greater proportion of the input information has been received[17]. While technical work alone is insufficient to measure the progress of a task, using the linear combination of the technical work fraction and the input information fraction is the simplest relation that incorporates the effects of both with equal weight.

The aleatory or stochastic part of uncertainty of information generated by a task is modeled as follows:

$$\varphi_{ti} = \epsilon_{ti} \times m_i \times \mathbf{UNIF}(0,1) \qquad \text{3-14}$$

where φ_{ti} is the ratio of the aleatory part of uncertainty at time *t* to the initial uncertainty of task *i*; m_i is a scaling factor; and $\mathbf{UNIF}(0,1)$ is a sample chosen from the uniform probability distribution between 0 and 1. Since the total

[17] Recall from section 2.4 that design iteration implies rework or refinement of activities to account for changes in their inputs.

uncertainty U is the sum of φ_{ti} and ϵ_{ti}, m_i represents the initial magnitude of the aleatory part of uncertainty of task *i* (Figure 3-9).

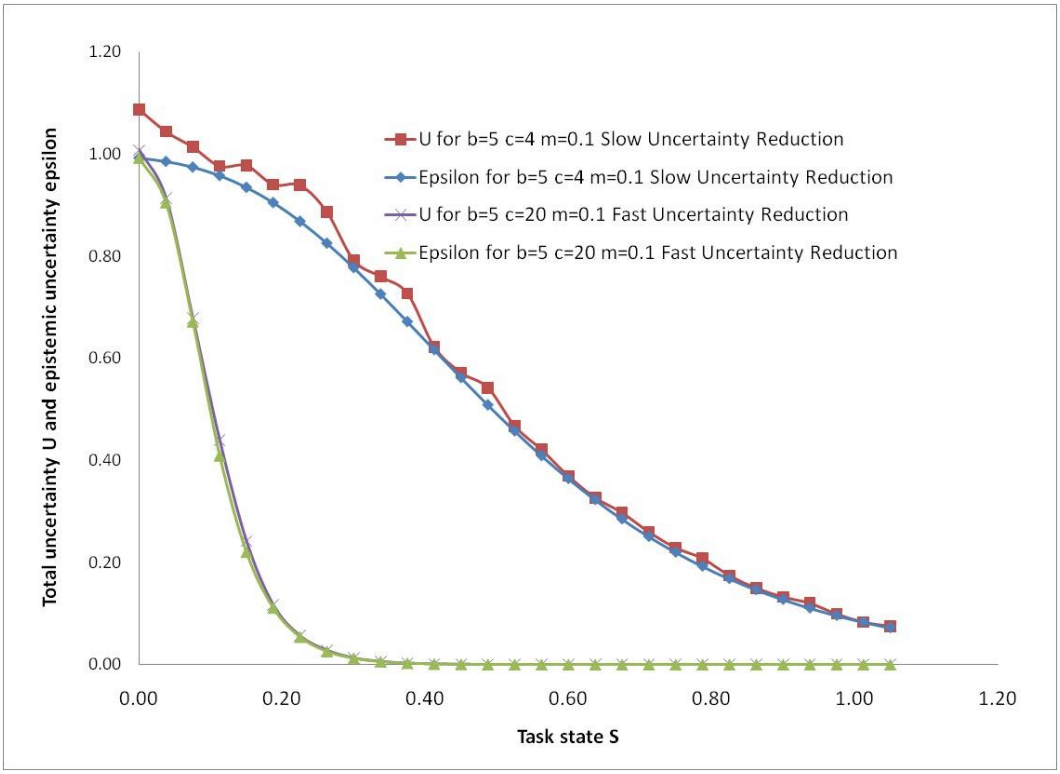


Figure 3-9 Instantiations of the model of uncertainty as a function of task state using the Gompertz Function

3.3.6.2 Quality attribute

The quality of a unit of information produced by each task is modeled to represent the conformance of the work to the system level requirements. The quality attribute Q_{ti} of a unit of information created at time *t* in task *i* is also modeled with a Gompertz function, but with different coefficients than those used for the reduction in epistemic uncertainty as follows:

$$Q_{ti} = e^{\left(-B_i e^{(-C_i S_{ti})}\right)} \quad \text{3-15}$$

where the task state of progress S_{ti} is defined in equation 3-13, and where B_i, C_i are input coefficients that define the shape of the curve (different from the coefficients used in equation 3-12 for the epistemic uncertainty). The quality

attribute is the fraction of achievement of the actual quality of the unit of information to the quality required, and it is related to S_{ti} as defined in equation 3-13 because I_{ti}/γ_i is a measure of completeness of the incoming information and WT_i/WD_i is a measure of the completeness of outgoing information. When each information entity is evaluated by the system level task (integrator resource), and its quality is insufficient, it is rejected, where the information entity is not passed to other tasks, but is returned to its originating task. As the state of this task progresses, the quality of this information entity improves according to equation 3-15 evaluated at the new state of progress of the task.

3.3.7 The starve condition

The performance of tasks in PD requires information from other tasks according to the dependency relationships described in section 3.3.3. Although there are initial estimated values of input information that are used to begin the work, without updated, more precise information, a task is 'starved' and the work cannot usefully continue until more input information is received.

In order to incorporate this idea into the model, we calculate a stochastically determined starve condition SC_{ti} as follows:

$$SC_{ti} = WD_i \times \mathbf{MAX}[\mathbf{NORM}(\mu_{ti}, \sigma_i), TRHD_i] \tag{3-16}$$

where **MAX** is the maximum value of the expressions separated by the comma in the square brackets, **NORM** is the inverse normal PDF with mean value μ_{ti}

$$\mu_{ti} = \frac{I_{ti}}{\gamma_i} \tag{3-17}$$

and standard deviation σ_i. The mean value μ_{ti} goes from 0 and 1 during the course of the execution of the task and the value of the standard deviation is an input parameter to the simulation chosen for each task at a value between 0 and 1. $TRHD_i$ is an input parameter to the simulation representing the fraction of *work*

that can be expected to be done before there is any input information to the tasks other than the preliminary information available before tasks begin.

The logic used to determine the starve condition states that it is most likely that a task is starved for input information when the proportion of technical work done exceeds the proportion of required input information received. For the general case this seems to be a reasonable assumption and the stochastic nature of the relation allows for the potential effects of variations from the mean in each specific instance.

When the value of WT_i exceeds SC_{ti}, work in task i stops with the release of the resource by the task entity. The task entity then waits until an information entity from another task is *read* by task i. This sequence simulates the starve condition and stoppage of work. The waiting time is accumulated and recorded as an output statistic called *starve time*. Once new information from another task is received, the task entity enters the queue for the resource associated with task i to continue to work, and a new starve condition is chosen according to equation 3-16.

In the case of reciprocal dependency, such a starve condition can lead to stoppage of all work in the process or deadlock, since if one task cannot proceed without information, it cannot develop information required by other tasks, which in turn can reach their own starve condition (recall that information develops according to progress in the *work* subtask, as described in section 3.3.5). Then, these tasks cannot develop the information required by the first task, leading to a situation where all tasks are stopped.

In practice, if such a deadlock condition occurs, the manager of the PD project must intervene to enable the restarting of work on tasks that have reached a starve condition and whose information is required to enable other tasks to progress. This is done, once a deadlock is discovered, by restarting work on one of the deadlocked tasks with an estimate or assumption about the missing information, sufficient to enable the generation of information required by other tasks and restart their work.

In the model, to simulate this managerial intervention, the algorithms described in the following sections were implemented. They differ in the way in which the deadlock is discovered.

3.3.7.1 Managerial intervention type A in a deadlocked condition

In this scheme, a deadlock is discovered between one pair of tasks that have stopped. In practice, this requires a close monitoring of the status of each task in a process and an understanding of the interdependencies between tasks. The manager, realizing that a pair of tasks are stopped and that they are blocked because they are waiting for information from each other, takes the step of restarting one of the tasks with an assumption about missing information sufficient to allow work to continue for a period of time. The algorithm in the simulation follows the logic:

1) If task entity i is in the starve condition, i.e., $WT_i > SC_{ti}$, check if another task k is starved in the project;
2) If statement 1) is true, check if task i requires information from another task k;
3) If statement 1) is true, check if task k requires information from task i;
4) If statements 2) and 3) are true, continue work in task i for another work cycle and choose another starve condition according to equation 3-16;
5) If any statement 1), 2), or 3) is false, for all k =1,2,..NPT, $k \neq i$, task i continues waiting for more information.

This logic unblocks the deadlock by allowing task i to do more work, enabling it to generate more information and thus allow other dependent tasks to become unblocked. This simulates one kind of managerial intervention, called type A, that occurs in a PD project when a pair of tasks are stopped due to a deadlock.

3.3.7.2 Managerial intervention type B in a deadlocked condition

Another type of managerial intervention is one which requires less knowledge of the process. Rather than intervene when *one* pair of tasks in the process is blocked, this intervention only takes place when the entire process is blocked.

Here, it may only become apparent that there is a problem in information exchange if all of the tasks in a project are deadlocked, and only then does management intervene to push one of the tasks forward. This scheme was also set up as an optional algorithm when condition 1) above is the only one checked, but in this case work does not continue in task *i* unless **all** of the tasks in the project are in the starve condition. This managerial intervention is called type B.

The consequences of the managerial decision to proceed regardless of the state of the input information are manifested as follows: work proceeds which allows further information to be generated by the task; this information is of higher uncertainty because the state of the task is lower (as described in section 3.3.6.1) since the amount of input information received is lower; this results in higher values of uncertainty in dependent tasks receiving the information which increases the likelihood of rework due to design iteration or failure in the design review. The likelihood of rework depends on the rate of reduction of uncertainty in the task.

3.3.8 System level integration

After each information entity completes the time period where it is 'served' by the resource from its associated task (representing the *prepare* subtask), the information entity releases the resource. Then, if the information entity is addressed to another task in the process, it pauses for a period representing a transmission delay; this delay is a random number chosen from an inverse normal PDF with modeling input values for its mean value and standard deviation $PRD_{i,1}$ and $PRD_{i,2}$ for each task *i* (see Table 3-3). The entity then waits to seize a system level integrator (refer to section 2.9). Here again, there is a queue for this resource.

Once the particular information entity has reached the head of the integrator queue, and the resource becomes available, this entity seizes one unit of the integrator resource for a random period of simulated time selected from an inverse normal PDF with (modeling input) mean value *MINT* and standard deviation

SINT. This simulates the time required for an integrator to evaluate a unit of information. Recall that each information entity carries with it an attribute representing the quality of this unit of information from the point of view of its suitability in meeting the requirements of the product (see previous section). Once the information entity has completed the simulated integrator process, a Bernoulli trial is made with probability of success equal to the value of the entity's quality attribute. If this trial fails, the information entity is sent back to the originating task. Otherwise, it continues to the task indicated in its address attribute after waiting for a stochastic period of time representing another transmission delay. This time period is chosen in the simulation for each information entity from an inverse normal PDF with modeling inputs for the mean (*ATD*) and standard deviation (*STD*).

The capacity of the integrator resource is flexible and can be increased based on the amount of work in its queue. This feature was included in the model because it was of interest to examine the effects of this resource constraint. Insufficient integrator resource capacity acts as an impediment to all tasks in the process because of the centrality of the integrator in the process network. An example of this feature in the model is given in section 4.6.7.

As each additional information entity 'enters' the queue for the integrator, its expected waiting time for an integrator resource is calculated as follows:

$$avintwait = \frac{\mathbf{NQ}(integrator\ queue) \times MINT}{(number\ of\ integrators)_t}$$

where $\mathbf{NQ}(integrator\ queue)$ is the length of the integrator queue at that moment. If the average waiting time is longer than a modeling input *IQ*, additional integrator resource capacity is added. Alternatively, if the average waiting time is less than 80% of the value of *IQ*, resource capacity is reduced. These actions replicate managerial actions typically taken to eliminate bottlenecks in critical activities in a project.

Other schemes to manage this resource, such as triggering the addition of integrator resources when $(avintwait + mint)$ multiplied by the number of entities remaining to process is greater than the remaining time to complete the phase. This more adaptive scheme is typical of project management monitoring the process and making adjustments to resource capacity, but requires a knowledge of the integrator workload. The value of *IQDecide* (a modeling input) signals the choice of which of these schemes to employ in a simulation.

Information entities waiting for the integrator resource may leave the queue and proceed directly to the final addressee when an entity's waiting time exceeds a value determined by a random number selected from an inverse normal PDF with (modeling input) mean value *MW* and standard deviation *SW*. Thus, each time an information entity enters the queue the maximum time it waits is individually determined. When information bypasses the integrator quality check, there is an increased chance of design version rework as is described in section 3.3.11.

3.3.9 Information reception in the model

Each information entity that is created follows a unique series of steps that are determined by the values of variables that change during the simulation. Some of these changes occur as the result of events initiated by other information entities. Each information entity is assigned address attributes (i.e., one attribute with the identity of the task from which it originated and one attribute with the identity of the task to which it will be sent) as well as attributes for quality, epistemic, and aleatory uncertainty which are determined by values of task state variables as described above.

Once an information entity completes the simulated integrator process and transmission delay, it processes another time delay representing the latency time within the receiving organization for the information to reach the addressee development team. This time period is chosen in the simulation for each information entity from an inverse normal PDF with modeling inputs for the mean ($LATM_i$) and standard deviation ($LATS_i$) for each task *i*.

The information entity then waits to seize the resource associated with the addressee task. Once it has seized the resource for the stochastically determined period of time for the *read* subtask, the resource is released. Subsequently, one of the following processing paths is followed:

1. If the entity has been returned to its originating task because it had been rejected by the integrator process, it is sent to the *prepare* queue for the resource once again. This simulates the rejection of information by the system level integrator, its return to the originating task, and its reissuance after the task has achieved a more advanced state. Once the information entity re-enters the *prepare* queue, its quality attribute is reset according to the new task state. This simulates the re-release of this unit of information when the task has achieved a more advanced state of progress. After performing the *prepare* process the entity once again goes to the integrator process.
2. If the entity has not been rejected in the integrator process, the processing thread associated with this entity ends, i.e., the entity is disposed of from the simulation, after the following state variables are assigned:
    i. I_{ti} - the number of units of input information *read* by task *i* from all other tasks (used in equation 3-13 to calculate the task state) since the start of the phase (the subscript *t* denotes that this value is accumulated until time *t* in the simulation)
    ii. ϵ_{ik}^{m} – accumulates the sum of the epistemic uncertainty attribute of each unit of information *read* by task *i* from task *k* between *work* cycle *m-1* and work cycle *m* of task *i*
    iii. U_{ik}^{m} – accumulates the sum of the epistemic uncertainty attribute plus the aleatory uncertainty of each unit of information *read* by task *i* from task *k* between *work* cycle *m-1* and work cycle *m* of task *i*
    iv. N_{ik}^{m} – accumulates the number of units of information *read* by task *i* from task *k* between *work* cycle *m-1* and work cycle *m* of task *i*.

3.3.10 Rework due to design iteration

Design iteration implies rework or refinement of activities to account for changes in their inputs as discussed in section 2.4. Interim information provided to a task is subject to change. The change is a result of ongoing work done in the task where the information originated and changes to the work of other dependent tasks that propagate through the system via the information flow between tasks. In the model these changes are reflected in the uncertainty attributes of information entities.

If input uncertainty is continuously decreasing, it is likely that none of the work done by the task requires rework since each updated piece of information is within the range of the precision of the previous piece. However, if there is an increase in input uncertainty, it is likely that rework will be required. This has to do with the perception of the precision in the received information. In practice, the development team performing a task takes the imprecision of input information into consideration when performing subsequent technical work, and does not make efforts further than the level of precision warranted. Only if the precision of subsequently received input information is below the precision of information received earlier is there rework required.

Now, the amount of required rework can be derived from the amount of work done between the current time and the time when the uncertainty of the input information was as high as in the latest received information. Essentially, we say that the portion of the work done with the wrong understanding of the precision of input information is what has to be redone. To enable this calculation, we record the epistemic and total uncertainty of information received by task *i* from each other task between each *work* cycle and associate it with the cumulative amount of *work* done by task *i* at that point.

We denote φ_{kl} and ϵ_{kl} as the aleatory and epistemic attributes of information entity l originating from task *k*. When information is *read* by the receiving task *i,*

these attributes are used to calculate the average total input uncertainty $\bar{U}_{ik}^{m}$ and the average input value of $\bar{\epsilon}_{ik}^{m}$ from task *k* between *work* cycle *m-1* and *m* in task *i*:

$$\bar{U}_{ik}^{m} = \left(\sum_{l=1}^{N_{ik}^{m}} (\varphi_{kl}^{m} + \epsilon_{kl}^{m}) \right) \; / \; N_{ik}^{m}$$

3-18

$$\bar{\epsilon}_{ik}^{\;m} = \left(\sum_{l=1}^{N_{ik}^{m}} (\epsilon_{kl}^{m}) \right) \; / \; N_{ik}^{m}$$

3-19

where N_{ik}^{m} is the number of information entities arriving at task *i* from task *k* between *work* cycles *m-1* and *m*.

Prior to commencing *work* cycle *m*, the uncertainty $\bar{U}_{ik}^{m}$ is compared to $\bar{U}_{ik}^{m-1}$. If it is higher, the amount of incremental rework that is required in task *i* due to the increase in uncertainty of information received from task *k* is calculated as follows:

$$[\Delta WD_i]_k = (WT_i^m - WT_i^p) \times \frac{NC_{ki}}{\gamma_i}$$

3-20

where WT_i^p is the amount of *work* done in task *i* until the beginning of a previous work cycle *p* such that the value of $\bar{\epsilon}_{ik}^{\;p}$ was greater than $\bar{U}_{ik}^{m}$. This is further illustrated in Figure 3-10.

If there is an increase in uncertainty in information received from other tasks, the total amount of rework needed to be done at the start of work cycle *m* is the sum of the incremental rework from equation 3-20 for each *k*. Thus, a new value of WD_i is calculated at the beginning of each work cycle as:

$$WD_i = \sum_{\substack{k=1 \\ k \neq i}}^{NPT} [\Delta WD_i]_k + WD_i^{old}$$

3-21

The total amount of rework generated corresponds to the weighted contribution of each task whose output information has increased in uncertainty.

The changes to all of the WD_i during a simulation are accumulated and recorded as an output statistic called *churn*. The term churn has been defined in the context of engineering design as rework due to redoing a task when making an informal incremental change (Bhuiyan, Gerwin et al. 2004) or as a scenario where the total number of problems being solved (or progress being made) does not reduce (increase) monotonically as the project evolves over time (Yassine, Joglekar et al. 2003). Here, churn is generated in a task when the uncertainty of interim input information is not monotonically decreasing and this occurs because of the sequence of events affecting information received by dependent tasks. Churn is generated because a decrease in the precision of input information implies that some of what was already done in the task must be redone.

Note that churn is quite different from design version rework, which is generated when there is an unsuccessful evaluation of intermediate outcomes of a progressive process. This type of rework is discussed in the next section.

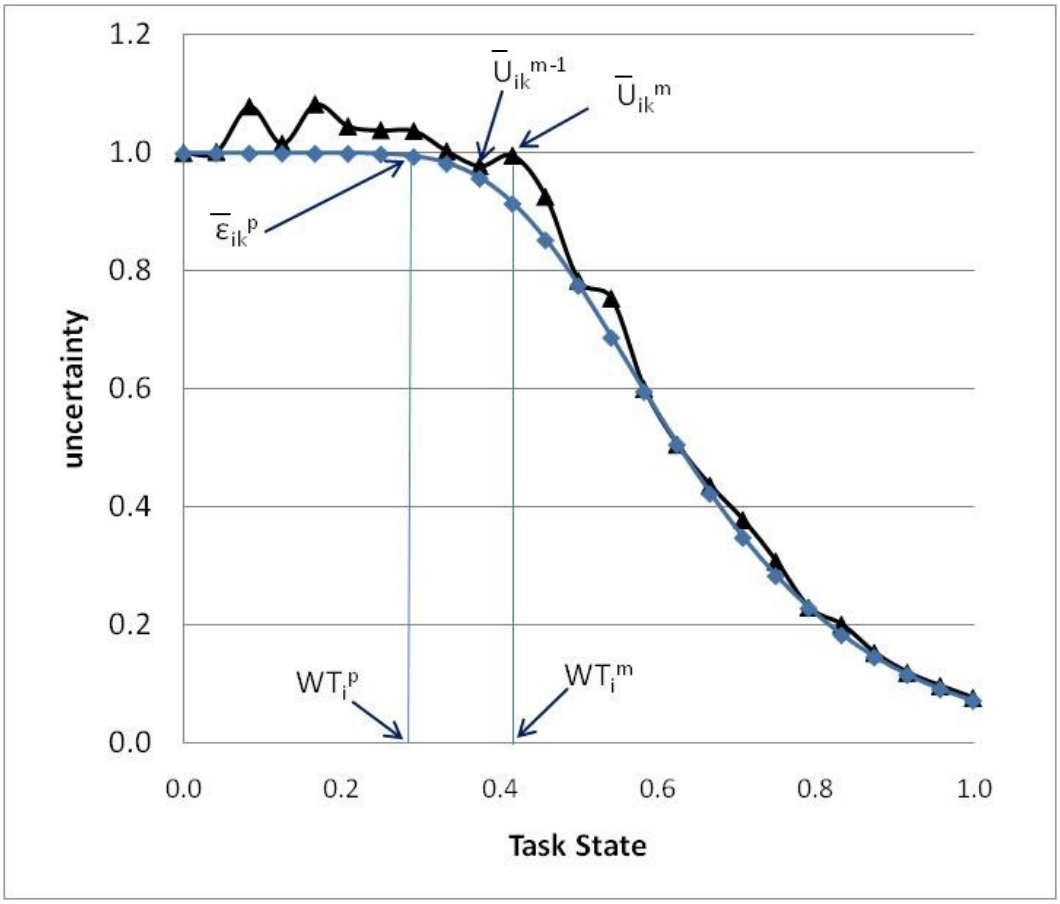


Figure 3-10 Illustration of the variables used in the calculation of rework in equations 3-18, 3-19, and 3-20.

3.3.11 Design version rework

In previous sections the concept of a progressive process that generates intermediate outcomes was introduced as an explanation for the partitioning of engineering design into phases. This process approach allows for a design review in which the intermediate outcome of a phase is evaluated and a decision can be made as to whether or not the intermediate versions of the design are suited to the requirements. This results in a decision on whether or not to continue forward or to perform rework of an earlier phase or design version rework. This type of rework, classified as a feedback iteration, results in one or several entire phases being redone, but with the benefit of the knowledge and uncertainty reduction obtained by the experience of the intermediate outcomes.

In the model we simulate this type of rework and feedback using the intermediate results of the simulation of previous phases. Once tasks in a phase are completed the probability of design version rework is governed by:

i. the certainty of information of each task at the end of the phase
ii. the percentage of completion of information exchange of each task,
iii. the percentage of completion of work of each task, and
iv. the percentage of completion of system level oversight.

A task may be complete when its allocated or scheduled span time is over, or because the *work* subtask is completed and all of the input information it required was *read*. The fraction of certainty in the last input information received by task *i*, called CRT_i is a measure of how likely it is that the task has completed its work with finalized information. This is calculated as:

$$CRT_i = \frac{1}{\gamma_i} \sum_{\substack{k=1 \\ k \neq i}}^{NPT} \left[\frac{(1 - \bar{\epsilon}_{ik}^{f}) \times NC_{ki}}{(e^{(-b_k e^{-c_k})})} \right]$$

3-22

The numerator in this equation is one minus the weighted epistemic uncertainty of information received at task *i* between its second to last and its last work cycle *f* from task *k*. The denominator is one minus the epistemic uncertainty of information of task *k* at a state of one.

Several ways for calculating the probability governing the success in a design review were considered. One scheme was to calculate the following value when task *i* ends in each phase of a simulated project as follows:

$$DVTEST_i = \mathbf{MIN} \left[\frac{1}{2} (CC_i \times CRT_i + WT_i / WD_i), SO_i \right]$$

3-23

where **MIN** returns the minimum of the two quantities separated by a comma within the square brackets, and

$$CC_i = \left(\frac{I_i + C_i}{\sum_{j=1}^{NPT} NC_{ij}} \right)$$

3-24

C_i is the number of information entities *read* at task *i* originating from task *i* (i.e., internal information) and I_i is the final number of information entities from other tasks *read* by task *i*. The fraction in equation 3-24 thus represents the amount of input information *read* by task *i* divided by the amount of input information it should have *read* (according to the interdependencies between tasks in the project as described in sections 3.3.3 and 3.3.4).

The term SO_i in equation 3-23 is the number of information entities sent by task *i* that went through the system level integrator process divided by the number of information entities that were sent by this task to the integrator. This fraction is a measure of how much of the information generated by the task was successfully verified by the system level integrator during the phase (Sections 3.3.8 and 2.9 gave details about the system level integrator in PD projects).

Thus, equation 3-23 relates the probability of success of a task in the design review to the minimum of:

i. the linear combination of the fraction of input information read by the task weighted by its certainty and the fraction of work completed
ii. the fraction of information generated by the task that is successfully verified by the system integrator.

A Bernoulli trial with probability $DVTEST_i$ simulates the design review of the work completed in the design phase for each task *i* in the phase. Successfully passing all of these trials allows the task entities to continue to the next phase in the project. Since each task that must be reworked generates new information that is required by other dependent tasks, a failure in one trial triggers design version rework in all the tasks. If there were earlier phases, the Bernoulli trials are redone for each of the earlier phases with their associated probabilities. Thus, a failure in

a design review may cause the project to return to earlier phases to begin again from that point.

This scheme, while intuitively attractive, creates a bias towards more design version rework when there are more tasks. This comes about because the probability of success in the design review using this scheme is the product of the probabilities of success of each of the tasks, and since these probabilities are each less than one, the probability of continuing to the next phase without design version rework decreases with an increase in the number of tasks. In effect, successfully completing a design review using this scheme amounts to *NPT* successes in *NPT* independent Bernoulli trials with probability $DVTEST_i$. If each of the $DVTEST_i$ are equal to a fraction p, this design review process is a binomial distribution giving the probability of success in a design review as p^{NPT}.

In order to avoid this bias, an alternate scheme for calculating the probability of completing the design review successfully was considered. Here, rather than performing a Bernoulli trial for $DVTEST_i$ for each task, the minimum value for each of CRT_i, CC_i, SO_i, and the final value of WT_i/WD_i are calculated and one Bernoulli trial for the minimum of each is performed. A failure in one trial triggers design version rework in all the tasks. As in the first scheme, if there were earlier phases and design version rework is triggered, these Bernoulli trials are redone for each of the earlier phases with their associated probabilities. Thus, a failure in a design review may cause the project to return to earlier phases to begin again from that point. In this way, failure in a design review is not biased towards a larger number of tasks in a phase, but the essential feature of failure in one task (in this case the one with the lowest performance) causing design version rework of all tasks in a phase is retained.

3.3.12 Feedback due to design version rework

Design version rework is the feedback type of iteration discussed in section 2.4. Each time an earlier phase is reworked, the benefit of feedback and learning is

incorporated into the model with the reduction of the technical work and of uncertainty in each task.

In practice, rework of a task in its entirety requires less effort than the original time because of several factors. First, there is a reduction of the effort required to perform the technical work processes, also known as first order learning or learning-by-doing. This is the well known learning curve effect that has been widely studied (Adler and Clark 1991; Von-Hippel and Tyre 1995). For this part of the reduction in effort we use the classic learning curve expression to calculate the average technical work requirement for each rework cycle $\overline{WK}_{i,DV}$, in terms of the original requirement for each task $\overline{WK}_{i,0}$, as follows:

$$\overline{WK}_{i,DV} = \overline{WK}_{i,0}(1 - LF)^{DV}$$

3-25

where DV denotes the design version rework number (subscript 0 denoting the original) and LF is the learning factor or percentage reduction in effort each time the task is repeated. Thus, we calculate WD_i, the initial *work* requirement for rework cycle DV of task i by multiplying the minimum, median, and maximum values of the original triangular PDF in equation by λ_{DV}, where

$$\lambda_{DV} = (1 - LF)^{DV}$$

3-26

thus ensuring that equation 3-25 is obeyed.

The second factor we consider in reduction of effort due to design version rework is the reduction in epistemic uncertainty. Having already performed the task along some solution path, it is likely that the epistemic uncertainty profile would reduce more rapidly each time a task is reworked. Here, we propose to model this by changing the parameters of the Gompertz function for epistemic uncertainty, b_i and c_i , for each rework cycle.

Let S_0 and S_1 be two values of the state of a task in the process and let the epistemic uncertainty of a task in the first design version be ϵ_0 and epistemic uncertainty of this task the DV_{th} design version (the DV_{th} rework cycle) be ϵ_{DV}:

$$\epsilon_0(S) = 1 - e^{\left(-b_0 e^{(-c_0 S)}\right)} \quad \text{and} \quad \epsilon_{DV}(S) = 1 - e^{\left(-b_{DV} e^{(-c_{DV} S)}\right)}$$

Then, we propose that

$$\epsilon_{DV}(S_0) = \lambda_{DV} \times \epsilon_0(S_0) \quad \text{and} \quad \epsilon_{DV}(S_1) = \lambda_{DV} \times \epsilon_0(S_1)$$

3-27

After some manipulation, we arrive at the following result for b_{DV} and c_{DV} :

$$c_{DV} = \frac{\ln(-\ln(1 - \lambda_{DV} \times \epsilon_0(S_0))) - \ln(-\ln(1 - \lambda_{DV} \times \epsilon_0(S_1)))}{S_1 - S_0}$$

3-28

$$b_{DV} = e^{c_{DV} S_0} \times \ln(1 - \lambda_{DV} \times \epsilon_0(S_0))$$

3-29

Results for Gompertz coefficients b=30 and c=6 in Figure 3-11 show how the epistemic uncertainty profile reduces more rapidly for each subsequent bout of design version rework with a learning factor LF of 0.2 in equation 3-25. In these cases the match points chosen were S_0 =0.4 and S_1=0.9 because they were found to best preserve the S shape of the curve at the start and end of a phase.

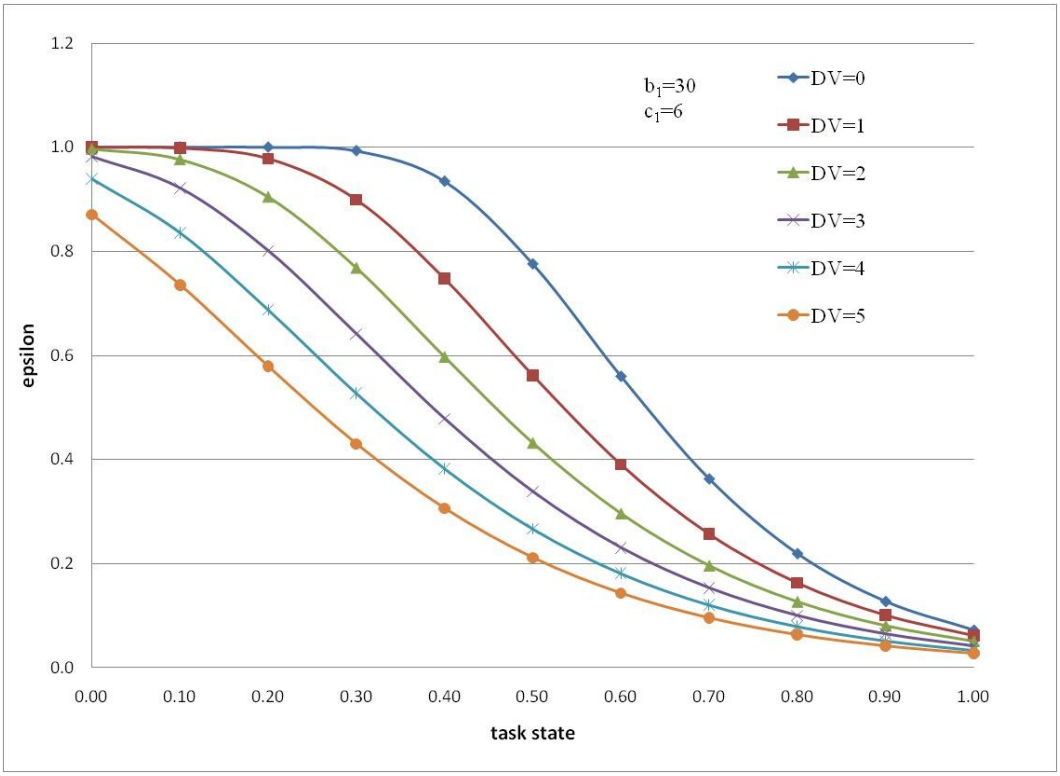


Figure 3-11 Variation of epistemic uncertainty with task state for each subsequent design version rework

3.3.13 The computer program

Arena© by Rockwell Automation Technologies Inc. was chosen as the software to construct the simulation model because of its facility of ready-made modules or software constructs that create, hold in queue, route entities, and seize release and schedule resources. Together with this facility, the software also has the flexibility of allowing programming with lower level commands and development of custom made modules for specific routines that recur in a simulation model. This was of particular use in this application where it was required to scale up the model with multiple tasks and phases.

The simulation is based on task modules of identical logical structure that are re-instantiated in a model according to the number of tasks in the project being studied. The tasks are combined into phases that are executed recursively as required for either rework or subsequent phases.

Table 3-3 Input parameters defining scenarios for the PD model[18]

Input Variable	Description
α	The ratio of total effort in communication to total technical effort in the modeled scenario (introduced in section 3.3.3.2)
ATD,STD	Mean value and standard deviation of the normal PDF for the transmission time of an information entity from the system level integrator to the final addressee
b, c	Vector defining Gompertz function values for the epistemic uncertainty for each task
B, C	Vector defining Gompertz function values for the quality for each task
BR	Vector containing the flag value for each task whether to broadcast information to all other tasks
CI	Vector defining the nominal communication interval (hours) for each task (section 3.3.5)
DeltaT	Vector defining Δt (hours) maximum work cycle time between which communications are attended to by each development team
D	Matrix defining dependency strength between each pair of tasks with element values of 0 to 9
INTMAX	System level integrator capacity
IQ	Integrator queue time threshold in hours where additional integrator resource is added
IQDecide	Flag for the type of integrator resource management scheme to use
LF	Learning factor for design version rework (equation 3-25)
LAT	Matrix defining the mean value $LATM_i$ and standard deviation $LATS_i$ of the PDF for communication latency for each task
MINT, *SINT*	Mean value and standard deviation of the normal PDF for the integrator process
MW, *SW*	Mean value and standard deviation of the normal PDF for the waiting time in the integrator queue after which information entities renege the queue
MANI	Flag value for the type of managerial intervention in a starve condition (0 for type A and 1 for type B as described in section 3.3.7)
M	Vector defining the scaling factor for aleatory uncertainty for each task
NPT	Number of tasks in each phase
OVF	Matrix defining the degree of overlap between each pair of tasks in a project

[18] Latin symbols for input variables in bold are matrices with elements in each row corresponding to the respective task

Input Variable	Description
PRD	Matrix defining the mean value and standard deviation of the normal PDF for the transmission time of an information entity from each task to the system level integrator
PR	Matrix defining the parameters of the triangular PDF for the *prep* subtask for each task
QMP	Vector defining minimum number of entities in the *prep* queue before attending to this queue
QMR	Vector defining minimum number of entities in the *read* queue before attending to this queue
RD	Matrix defining the parameters of the triangular PDF for the *read* subtask for each task
SCH	Vector defining the ratio of scheduled span time for each task to the maximum value of the *work* subtask PDF for that task.
STSD	Vector defining the standard deviation σ_i for the calculation of the starve condition for each task (equation 3-16)
TRHD	Vector defining the lower cutoff for the calculation of the starve condition for each task (equation 3-16)
TP	Number of phases in the project
WK	Matrix defining the parameters of the triangular PDF for the *work* subtask for each task

Table 3-4 Description of symbols employed in the model

Symbol	Description
$avintwait$	The expected waiting time in the integrator queue at any time
Dij	The dependency of task *j* on information output of task *i*
Uij	The initial uncertainty of the information input from task *i* to development team *j*
Sij	The sensitivity of task *j* to changes in information input from task *i*
C_L	A scaling factor which relates the number of information units that must be communicated to the numerical scale chosen to quantify the levels of uncertainty and sensitivity
NC_{ij}	The number of units of information that must be communicated from task *i* to task *j*
WD_i	The value of total technical work required in task *i*, initially calculated from the inverse triangular PDF for the task
PID	The attribute assigned to an information entity identifying the task from which it came
TO	The attribute assigned to an information entity identifying the task to which it is to be sent
WT_i	The state variable for the work accomplished in the task *i*
DT_i	An interim state variable which keeps track of the number of hours of effort expended in the current work cycle in task *i*
$qlty$	The attribute assigned to an information entity representing the quality of the information it carries
eps	The attribute assigned to an information entity representing the epistemic uncertainty of the information it carries
phi	The attribute assigned to an information entity representing the aleatory uncertainty of the information it carries
I_{ti}	The amount of input information received in task *i* from all other tasks until time *t*
S_{ti}	The state of a task *i* at any time *t*
γ_i	The total amount of input information required to be received by task *i* from all other tasks (derived from matrix **NC**)
SC_i	A stochastically determined starve condition for task *i*
ϵ_{ik}^{m}	Accumulates the sum of the epistemic uncertainty attribute of each unit of information *read* by task *i* from task *k* between work cycle *m-1* and work cycle *m*
U_{ik}^{m}	Accumulates the sum of the epistemic uncertainty attribute plus the aleatory uncertainty of each unit of information *read* by task *i* from task *k* between work cycle *m-1* and work cycle *m*
N_{ik}^{m}	Accumulates the number of units of information *read* by task *i* from task *k* between work cycle *m-1* and work cycle *m*

Table 3-5 Key assumptions in the model

	Key assumptions in the model	Rationale
1	Amount of information exchange between tasks is proportional to their dependency strength	See section 3.3.2
2	Technical work fraction and input information fraction have equal weight in determining the state of progress of a task	See section 3.3.6
3	The functional relation of uncertainty with the state of progress of a task	See section 3.3.6
4	Starve condition is normally distributed and the mean value occurs when the technical work fraction exceeds the input information fraction	See section 3.3.7
5	The criteria for churn type of rework	See section 3.3.10
6	The criteria for design version rework	See section 3.3.11
7	Model for feedback and learning with design version rework	See section 3.3.12

4 Simulation Experiments with the Model

4.1 Statistical significance of results

Recall that the PD system model is stochastic and as such its outputs are stochastic variables. The simulation of a PD process using the model described in chapter 3 ends when the tasks in the last phase of the project are all complete. This kind of simulation is classified as a terminating simulation[19]. Now, in order to obtain results of simulation experiments that have statistical significance different runs or *replications* of the simulation must be made with the same input parameters. In this way each replication is therefore a trial and the span time and effort (as well as other outputs) of each replication constitute a random variable. Each replication is carried out with different random numbers generated for the probability distributions used in the simulation, thus ensuring that the replications produce output results that are independent. By making sufficient replications of the simulation for each scenario, meaningful statistics of mean values and variances of output quantities can be obtained. In the remainder of this section a discussion of what confidence intervals for the results of simulations mean and how they have been determined is presented.

Suppose that X_1, X_2,X_3,…X_n are independent and identically distributed random instances of a measured variable with mean μ and variance σ^2; for example, X_1 can be the resultant span time of the PD project for replication 1, X_2 can be the resultant span time of the same PD project for replication 2, and so on. Our objective is to estimate the values of μ and σ^2 (to continue with the example, we wish to estimate the mean value and variance of the span time for a modeled scenario based on *n* replications of the simulation run of the scenario). The sample mean after *n* replications is

---

[19] A terminating simulation is defined as a simulation in which the model dictates specific stopping conditions as a natural reflection of how the real system actually operates.

$$\bar{X}(n) = \frac{1}{n}\sum_{i=1}^{n} X_i$$

4-1

and the sample variance is

$$\bar{S}^2(n) = \frac{1}{n-1}\sum_{i=1}^{n} [X_i - \bar{X}(n)]^2$$

4-2

We want to estimate how close the value of $\bar{X}(n)$ is to μ and how close $\bar{S}^2$ is to σ^2. To this end, we make use of the confidence interval which can provide a measure of the error in the estimation of μ when we use $\bar{X}(n)$ from equation 4-1 to calculate the mean value of the sample data. The half-width of a 95% confidence interval, also calculated from the sample data, can be used to calculate the approximate lower and upper bounds about the sample mean between which 95% of the simulation results will lie for a given number of replications, *n*.

It turns out that if the X's are independent and normally distributed random variables, a 100(1 – α) confidence interval for μ (for 95% confidence interval α=0.05) is given by

$$\bar{X}(n) \pm t_{n-1,1-\alpha/2}\sqrt{\frac{\bar{S}^2(n)}{n}}$$

4-3

where $t_{n-1,1-\alpha/2}$ is the upper 1 – α/2 critical point for the t distribution with *n* - 1 degrees of freedom (Law and Kelton 2000)[20].

In practice, we want to know how large *n* must be in order to have the 95% confidence interval be 'small enough'. Small enough for our purposes is deemed to be when the respective confidence interval is less than 5% of the calculated

[20] The t distribution is a continuous probability distribution that arises when estimating the mean of a normally distributed population in situations where the sample size is small and a population standard deviation is unknown. For a 95% confidence interval α = 0.05.

average of the both the span time and effort. That is, we want *n* to be large enough so that 95% of the time, the average value of the effort for a simulated scenario will fall within 5% of the value of the average effort calculated from the *n* previous replications of the scenario. Similarly, we also want *n* to be large enough so that 95% of the time, the average value of the span time for a simulated scenario will fall within 5% of the value of the average span time calculated from the *n* previous replications of the scenario.

The scheme used to find the required value of *n* was to calculate the 95% confidence interval during the simulation using equation 4-3 after each replication once *n* reached 49, and to compare it to the calculated average value of each of the two key output variables, span time and effort. If 5% of the average span time or effort calculated at that point in the simulation is greater than the respective confidence intervals for span time and effort calculated at that point, the simulation continues to the next replication. If the respective confidence intervals are both less than 5% of the calculated mean value of these output variables, the simulation is terminated after the next replication. The number of replications *n*, for each different scenario, sufficient to achieve this confidence interval was usually about 100.

The confidence interval so calculated is approximate only in that it is not absolutely true that the sample data is normally distributed. However, use of the standard normal theory statistical inference methods applied here is justified since in probability theory, the central limit theorem states that the mean of a sufficiently large number of independent, identically distributed random variables, each with finite mean and variance, is approximately normally distributed. Intuitively, since each replication of the simulation is essentially the sum of many individual random processes, it seems reasonable to assume that these are symmetrically distributed around a finite mean value.

4.2 Output quantities and input parameters

The output quantities of interest are the total effort and span time of the simulated PD project. Other quantities of interest are the total churn, starve time, design version rework cycles, integrator effort, average resource utilization, and average queue lengths and waiting times. For each of these output variables the simulation provides data averaged across replications, half width of a 95% confidence interval, minimum average, maximum average, minimum value, and maximum value. As explained in the previous section, the number of replications carried out for a particular scenario were sufficient to ensure that the 95% confidence interval was smaller than 5% of the average of the output variables in each case.

4.2.1 Normalization of output quantities and input parameters

Comparisons of different scenarios using the model are most usefully made when inputs and outputs are normalized. This can be understood if we want to study the effect of variation of the number of tasks or phases in a project decomposition. Since the work per task is determined for each replication by the input parameters of a triangular PDF, we can hold the average sum of work of all tasks in all phases as constant across the scenarios we want to compare. In this way, we can judge the effects of each input parameter on the performance of a simulated system that must execute a group of tasks with the same total technical effort.

The total average technical effort TWK in a project of TP phases and NPT tasks per phase is:

$$TWK = \sum_{l=1}^{TP} \left(\sum_{k=1}^{NPT} \overline{WK}_k \right)_l$$

4-4

where $\overline{WK}_k$ is the average technical work of task *k,* calculated using the minimum, median, and maximum parameters of the triangular PDF of the k^{th} *work* subtask (stored in matrix **WK**) as follows:

$$\overline{WK}_k = {}^1/_3 \times \left(WK_{k,1} + WK_{k,2} + WK_{k,3}\right)$$

4-5

The communication effort in the project can be calculated as $\boldsymbol{\alpha}$ multiplied by the technical effort (from the discussion in section 3.3). Thus, the total average effort *TE* in a project is given by:

$$TE = (1 + \boldsymbol{\alpha}) \times TWK$$

4-6

In a project scenario where all the work is performed by one resource, the average project span time is the same as the total average effort *TE*. Therefore, both the output quantities span time and effort of each simulated project are normalized by dividing them by *TE*. Other output quantities such as total churn, integrator effort, and others are also normalized using *TE*.

Input parameters were normalized to facilitate comparison across scenarios as well. For example, the work cycle Δt_i, usually expressed in hours, is normalized with the nominal span time of a task. Thus, the normalized work cycle time NDT_i is expressed as:

$$NDT_i = \frac{\Delta t_i \times TP \times NPT}{TE}$$

4-7

A summary of normalized input parameters and output quantities is given in Table 4-1 (refer to Table 3-3 and Table 3-4 for a description of the symbols used here).

Table 4-1 Normalized input and output parameters

Parameter	Definition
Normalized Effort	*NEFFORT=Effort/TE*
Normalized Span Time	*NSPAN=Span Time/TE*
Normalized Churn	*NCHURN=Churn/TE*
Normalized Integrator Effort	*NINTEFF=IntEffort/TE*
Normalized Starve Time	*NSTARVE=Starve time/TE*
Normalized Communication Interval	$NCI_i = CI_i \times TWK/TE$
Normalized nominal work cycle Δt	$NDT_i = \Delta t_i \times NPT \times TP/TE$
Normalized IQ	$NIQ = IQ \times NPT \times TP/TE$
PDF parameters for normalized latency time	$\mathbf{NLAT} = \mathbf{LAT} \times NPT \times TP/TE$

4.3 Verification and validation of the model

Verification means ensuring that the model behaves as it was intended, which means to ensure that the computerized model and its implementation are correct versus the specifications described in chapter 3. Validation means ensuring that the model's predictions reflect reality within the intended domain (Law and Kelton 2000). In this section the steps taken to accomplish verification and validation are described.

4.3.1 Model verification

The computer program was written as a series of modules or sub-programs, each of which began as a simple algorithm. More complexity was added to each module after ensuring that the module operated as intended. Visualization of the behaviour of entities and values of variables in the simulation was done by proceeding at slow motion or discrete steps and enabled verification that these developed correctly. Tests were run with simple cases to verify that the computer program produced expected results. The behaviour of the mean value, confidence intervals, minimum and maximum values of variables such as queue sizes, waiting times, effort, span time, over differing numbers of replications were observed to ensure that these were within reasonable ranges over a wide range of scenarios. Additionally, test cases were run at the boundary values of input parameters to confirm results as would be expected. For example, the scheduled

span time (defined in input vector **SCH** in Table 3-3) of individual tasks was varied from very short times to very long times to observe how the behaviour of the project span time and effort changed. It was expected that as scheduled time was increased the behaviour of the overall span time and effort would asymptotically reach a final value that did not change when **SCH** was further increased.

4.3.2 Validation of the model

Ideally the best way to validate a model of a PD process would be to compare data from an actual process with data of a simulation of the same process. The required data would be span time and effort for cases when the real project was carried out with variation of the variables being studied in the simulation such as the interval between information exchanges, scheduled span times of tasks, etc. The variation of the real process output data with changes in the key variables would be compared to those predicted by the simulation.

In practice, this is not feasible. Historical data of span time and effort of PD projects, if available, would not have more than one instance of conditions for the same set of tasks. This is the case for PD since no project is the same as another. Furthermore, while the data for effort may be available for each project, it is not likely that span time data would reflect comparable conditions from one project to another nor those assumed in the model. This is because PD projects are often subject to delays due to conditions external to the project and to the system modeled. Often a project may be delayed because of a business situation such as market conditions or financial constraints, or delays may be imposed due to prioritization of other projects at certain times during the product development cycle.

Therefore, validation is based on comparison of the results predicted by the model with a combination of those predicted by other models with data from real processes that are available for specific conditions, i.e., relevant results from similar studies, and with the comparison of the specific behaviors incorporated in

the model's logic with those that have been reported to take place in PD, i.e., validation based on expert knowledge (Kleijen 1999)). These results were described to practitioners of PD in the aerospace industry in a series of interviews that were conducted to analyze the coordination mechanisms in use and to place them into the context of the framework presented here (further details of these interviews are given in chapter 5). Finally, experience and intuition based on the author's years of practice in industry were used to hypothesize how certain components of the system operate.

In the remainder of this chapter results of simulation experiments are shown and compared to results of models and data collected in previous studies. The results shown here also serve to illustrate the operation of the model and how it can explain the mechanisms that drive the PD process.

4.4 Modelling uncertainty, sensitivity and information evolution

As discussed in section 3.3.3 we model the dependency of information between each pair of tasks as the product of the initial uncertainty and the sensitivity of the 'downstream' task to the information provided by the 'upstream' task. The quotation marks are used here because in the general case of reciprocal dependence and sequencing of tasks such that they begin simultaneously, these terms mean only that we refer to the information providing task and the information receiving task. The information dependency between tasks is described in matrix form, where each element D_{ij} of the matrix represents the dependency of task j on information provided by task i. Therefore, for a given value of initial uncertainty, the more sensitive the downstream task is to changes in the information provided by the upstream task, the higher the value of the corresponding element in matrix **D**.

Examples of this matrix for a process with 5 tasks is shown in Table 4-2 and Table 4-3 for different dependency relationships between the tasks. The first row in Table 4-2 shows that task 1 provides information to tasks 2, 3, 4, and 5 where for each of these tasks, the product of the initial uncertainty (valued on an integer

scale of 0 to 3) and the sensitivity (valued on an integer scale of 0 to 3) of the input information from task 1 are high. Similarly, the third row shows that tasks 1 and 2 are not dependent on task 3 for any information and that task 4 and task 5 are dependent upon task 3, but that the dependency is low. The sensitivity of task 4, for example, to changes in information from task 3 is either very low and the uncertainty is high, or vice-versa. In either case, the dependency is low and according to our model, there are fewer information updates required from task 3 by task 4 and 5. The lower triangle of the matrix is populated with only zeros and this indicates that the dependency between the tasks is sequential. Task 2 depends only on task 1, task 3 only on task 1 and 2, etc. There is no dependency from a higher numbered task to a lower numbered task so that if the tasks were executed in sequential order there would be no downstream information required by an upstream task.

Table 4-2 A dependency matrix case (a)

TASK	1	2	3	4	5
1	9	9	9	9	9
2	0	9	9	9	9
3	0	0	9	3	3
4	0	0	0	9	9
5	0	0	0	0	9

In Table 4-3, the lower triangle of the dependency matrix is not zero indicating the presence of reciprocal dependencies. Task 1, for example, is dependent on task 3, task 4, and task 5. The dependency relations lead to the information exchange requirements for each task as indicated in equation 3-4, and as such they determine how quickly a task can be completed. This develops in the simulation according to how quickly information can be gotten to each task requiring it. If there are organizational impediments to information transfer or delays in the tasks providing the information, the task in question is delayed.

The two dependency matrices shown also have values along the diagonal. As discussed in section 3.3.4 these elements represent the uncertainty that must be resolved within the development team performing the task itself and set the

amount of information sent and received internally. In the simulation, these communications do not contribute to the uncertainty of input and rework calculations, but they occupy the resource assigned to each task in the model and they are part of the calculation that is used to form the probability of success in the simulated design review at the end of each phase.

Table 4-3 A dependency matrix case (b)

TASK	1	2	3	4	5
1	9	9	9	9	9
2	0	9	3	6	9
3	3	0	3	6	9
4	9	6	0	9	9
5	6	0	6	3	9

The reduction in uncertainty that takes place during the execution of a task (referred to as upstream information evolution in other studies of PD processes (Krishnan, Eppinger et al. 1997) or information stability and precision (Terwiesch, Loch et al. 2002)) is modeled here using a Gompertz curve as a function of the task state. Recall from section 3.3.6.1 that we defined the task state as the linear combination of the technical work fraction done and the input information fraction received. In the following, we show several models of uncertainty reduction profiles that we used in simulations to predict span time and effort. The vertical axis in these figures is the uncertainty at task state S during the execution of a task divided by the uncertainty at S=0. The total uncertainty is made up of the sum of the epistemic uncertainty ε and the aleatory uncertainty φ which are attributes of units of information created in each task in the simulation according to the equations 3-12, 3-13, and 3-14.

Figure 4-1 shows how the parameters b and c of the Gompertz function can be used to model the rate of the reduction of ε at the beginning of the task with value of b, and the approach to the lower asymptote with the value of c.

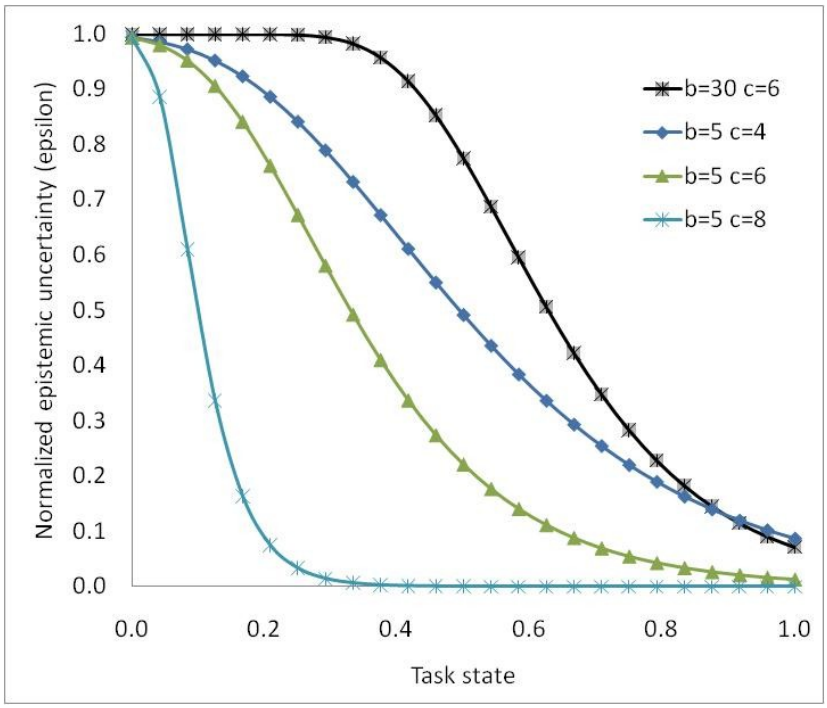


Figure 4-1 Epistemic uncertainty ε versus task state as modeled for various values of b and c

Each time that a value is set for the uncertainty attribute of an information unit, the inverse uniform PDF is used to determine the value of φ for the aleatory portion. Figures 4-2, 4-3, 4-4, and 4-5 each show a sample of these values for various values of ε. Notice how, for more rapidly reducing epistemic uncertainty, the aleatory uncertainty reduces in magnitude. However, the precise way the values fluctuate introduces the randomness in the uncertainty of information received by each task in the process modeled. Each of the cases shown in the following figures has a constant value of the magnitude of aleatory uncertainty *m* defined in equation 3-14.

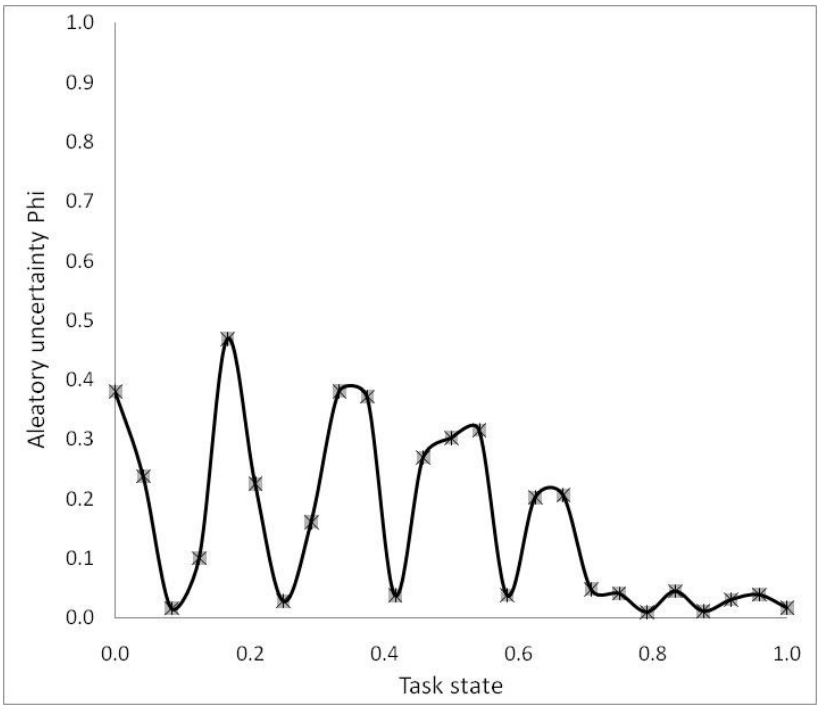


Figure 4-2 Aleatory uncertainty φ versus task state for b=30, c=6, m=0.5

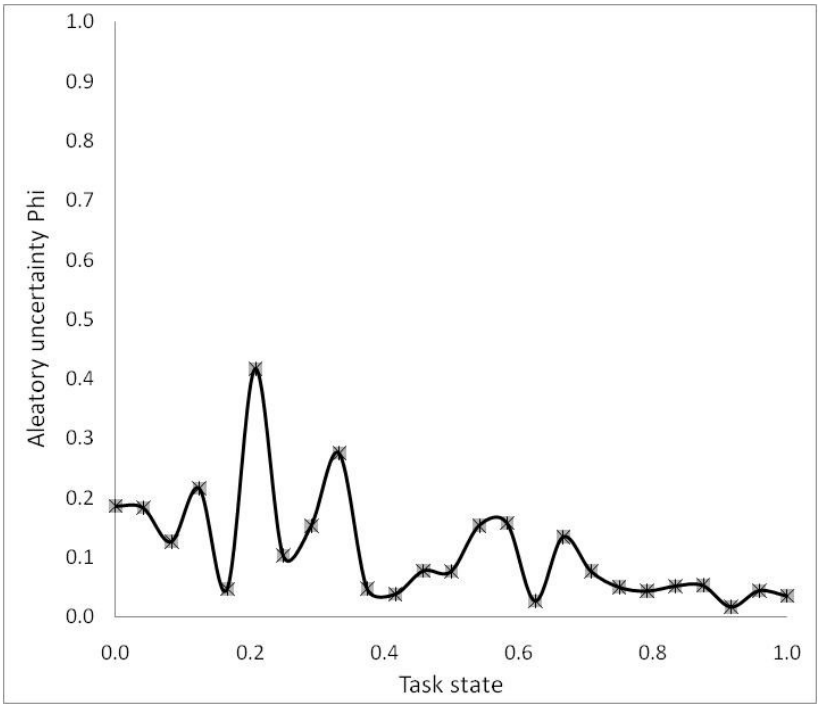


Figure 4-3 Aleatory uncertainty φ versus task state for b=5, c=4, m=0.5

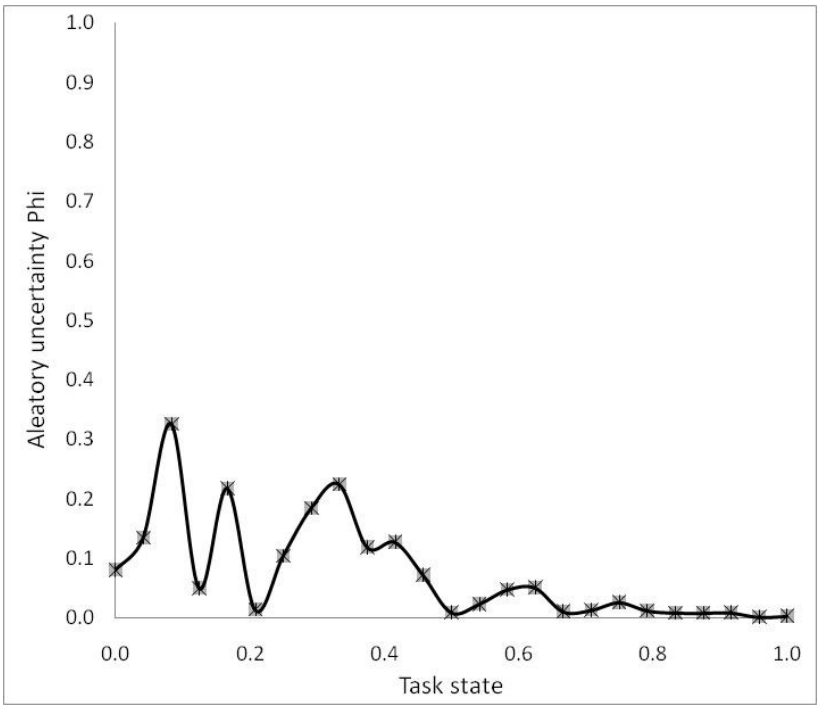


Figure 4-4 Aleatory uncertainty φ versus task state for b=5, c=6, m=0.5

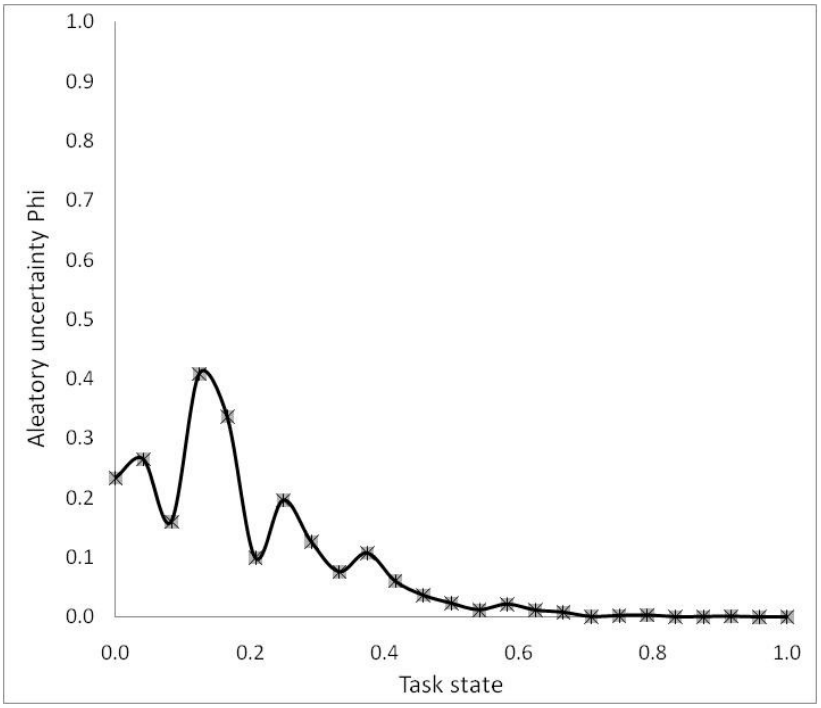


Figure 4-5 Aleatory uncertainty φ versus task state for b=5, c=8, m=0.5

The combined values of ε and φ give uncertainty reduction profiles as shown in the examples below. Here, we see how the combination of the values of *b, c,* and *m* lead to very different profiles of uncertainty reduction. In the results of simulations shown in the following sections we refer to these uncertainty reduction profiles and their influences on PD project performance. An example of a task with slow reduction in epistemic uncertainty and low magnitude of aleatory uncertainty could be the design of a cockpit windshield. The design must incorporate the requirements for visibility of landmarks by the pilot on landing and takeoff, the seat design, typical pilot heights, aerodynamic considerations, potential cockpit layouts of instrumentation, controls, and equipment, etc. These studies require time and effort, and so the design evolves slowly in precision until all of the information can be synthesized into a shape that meets the requirements. The information needed to complete the design, however, is stable in terms of the requirements and standards that must be considered, such that there is little potential for unexpected changes.

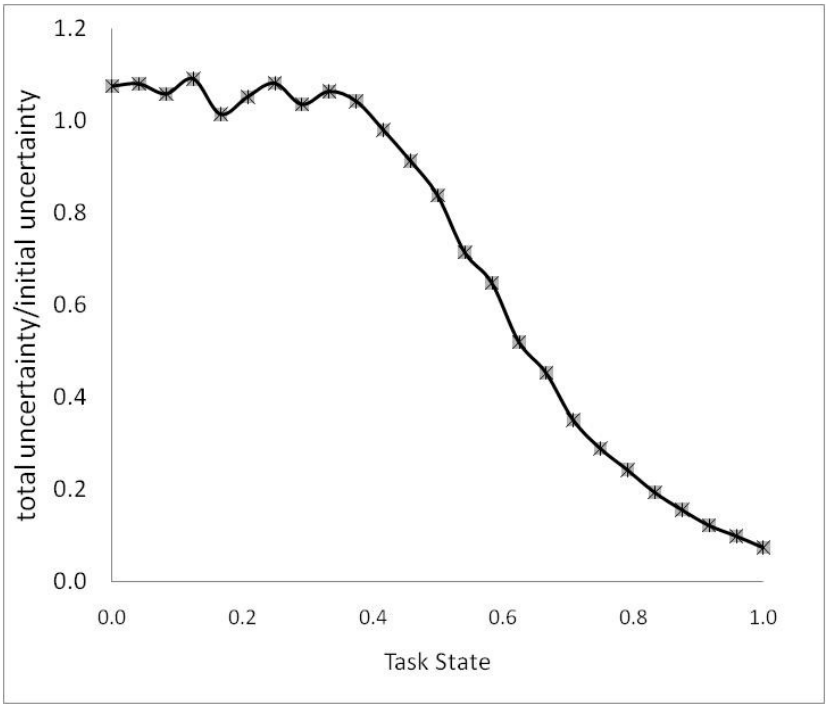


Figure 4-6 The uncertainty for a task *b*=30, *c*=6, (slow reduction in epistemic uncertainty) *m*=0.1 (low magnitude of aleatory uncertainty)

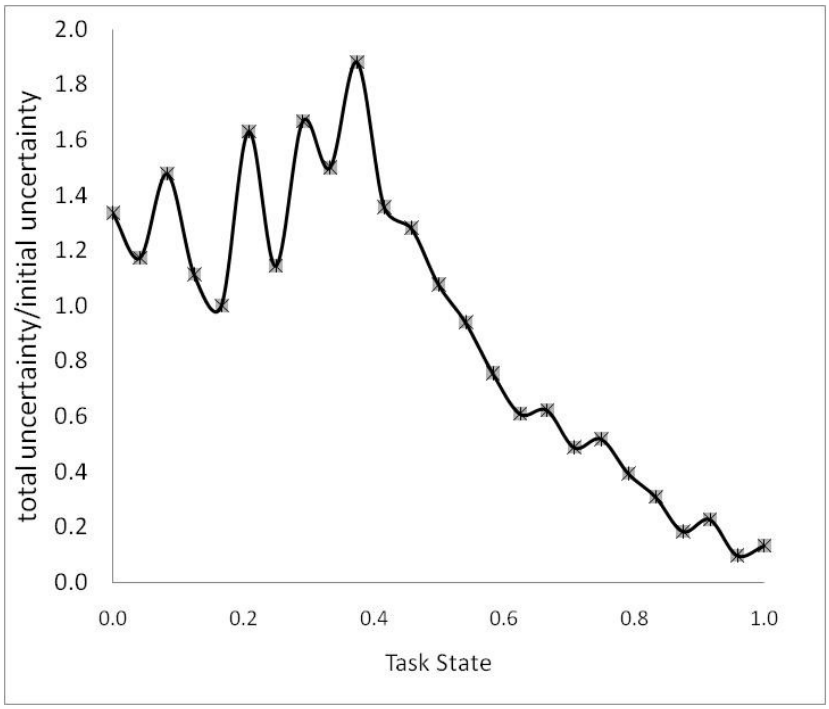


Figure 4-7 The uncertainty for a task b=30, c=6, (slow reduction in epistemic uncertainty) m=1.0 (high magnitude of aleatory uncertainty)

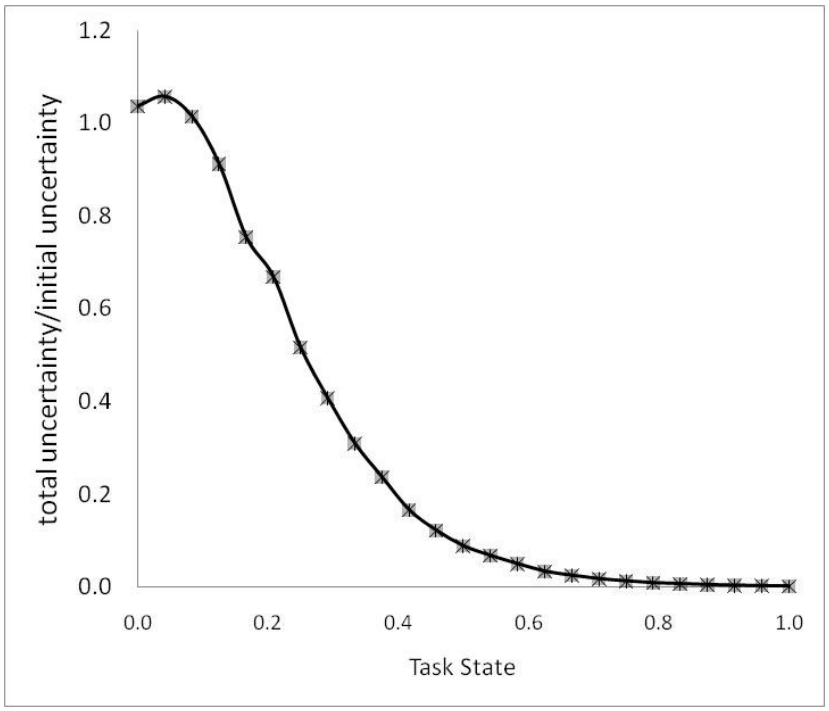


Figure 4-8 The uncertainty for a task b=5, c=8, (rapid reduction in epistemic uncertainty) m=0.1 (low magnitude of aleatory uncertainty)

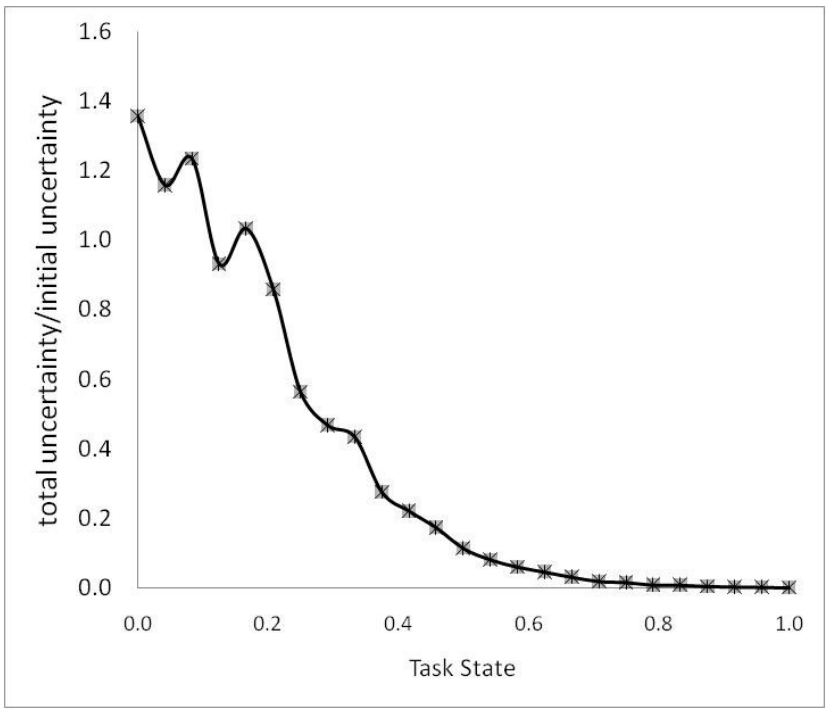


Figure 4-9 The uncertainty for a task *b*=5, *c*=8, (rapid reduction in epistemic uncertainty) *m*=0.5 (medium magnitude of aleatory uncertainty)

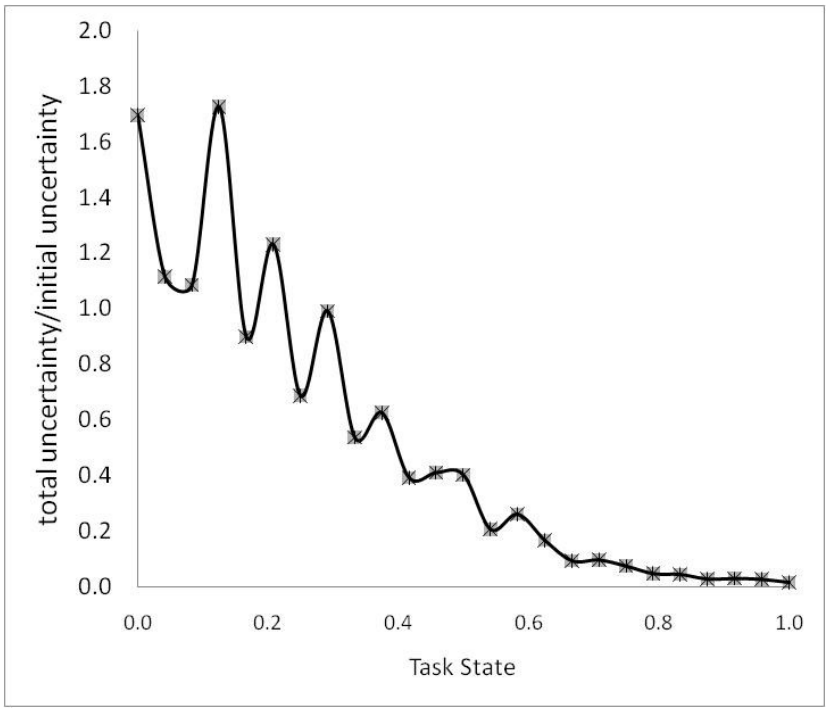


Figure 4-10 The uncertainty for a task *b*=5, *c*=6, (moderate reduction in epistemic uncertainty) *m*=1.0 (high magnitude of aleatory uncertainty)

4.5 Sequential dependency

In this section we examine simulation results for the case of sequential dependency. For example, with three sequentially dependent tasks, task 3 requires information from tasks 1 and 2; task 2 requires information only from task 1; and task 1 does not require any information from any other tasks.

Simulations were run with 5 sequentially dependent tasks in a PD project. The dependency matrix for the case of 5 tasks (*NPT*=5) with high sequential dependence is shown in Table 4-4.

Table 4-4 matrix **D** for high sequential dependency

TASK	1	2	3	4	5
1	9	9	9	9	9
2	0	9	9	9	9
3	0	0	9	9	9
4	0	0	0	9	9
5	0	0	0	0	9

The scenarios with sequential dependency were run for cases with α=1, two phases (*TP*=2) and *TWK*= 15,000 hours. The following were the input parameters for the *prepare* (PR), *read* (RD), and *work* (WK) subtasks (the 3 columns in each matrix here are the minimum, median, and maximum values in hours of the respective triangular PDFs):

$$\mathbf{PR} = \begin{bmatrix} 1 & 4 & 9 \\ 1 & 4 & 9 \\ 1 & 4 & 9 \\ 1 & 4 & 9 \\ 1 & 4 & 9 \end{bmatrix} \quad \mathbf{RD} = \begin{bmatrix} 1 & 4 & 8 \\ 1 & 4 & 8 \\ 1 & 4 & 8 \\ 1 & 4 & 8 \\ 1 & 4 & 8 \end{bmatrix} \quad \mathbf{WK} = \begin{bmatrix} 643 & 1286 & 2571 \\ 643 & 1286 & 2571 \\ 643 & 1286 & 2571 \\ 643 & 1286 & 2571 \\ 643 & 1286 & 2571 \end{bmatrix} \tag{4-8}$$

The value for C_L is calculated from equation 3-5 which results in an information exchange matrix **NC,** calculated with equation 3-4 and shown in Table 4-5. As explained in section 3.3.3, the element *i* and column *j* of this matrix are the number of information entities that should be communicated from the task *i* to the task *j* during the execution of the project.

Table 4-5 The resultant information exchange matrix **NC** for the sequential dependency scenarios

TASK	1	2	3	4	5
1	56	56	56	56	56
2	0	56	56	56	56
3	0	0	56	56	56
4	0	0	0	56	56
5	0	0	0	0	56

Other input parameters were set at the values shown in Table 4-6.

Table 4-6 Values for input parameters for scenarios with sequential dependency

Input parameter	Value
BR	(0, 0, 0, 0, 0)
B	(0, 0, 0, 0, 0)
C	(1, 1, 1, 1, 1)
INTMAX	20
LF	0.2
MINT	15
MANI	0 (signifies type A intervention in a starve condition)
MW	1200
NDT	(0.02, 0.02, 0.02, 0.02, 0.02)
NIQ	0.002
NLAT	Each row identical (0.005, 0.001)
PRD	Each row identical (0.5, 0.1)
QMP	(1, 1, 1, 1, 1)
QMR	(1, 1, 1, 1, 1)
ATD, STD	0.5, 0.1
STSD	0.4
SCH	(11, 11, 11, 11, 11)
TRHD	(0.1, 0.1, 0.1, 0.1, 0.1)

These values were chosen so that we could isolate the effects of input parameters we wished to study. Thus, for example, the value of parameters governing quality of information **B** and **C** were set as shown, effectively making the quality value unity at all times. The parameter governing the integrator resource capacity *INTMAX* was set to 20 to ensure that there would be no constraint imposed in this part of the process. The allowed span time for each task, set by **SCH**, was

sufficiently high to ensure that this would also impose no constraint in the following scenarios.

4.5.1 Effects of task overlapping and uncertainty

In this section, we examine the behaviour of the sequentially dependent PD process described above with overlapping of tasks in the process under different conditions of uncertainty. Overlapping of tasks in an engineering process can work well under conditions of sufficient communication of intermediate information between interdependent tasks and under suitable conditions of reduction of uncertainty in tasks (Bhuiyan, Thomson et al. 2006). In this section, we examine under what conditions overlapping achieves the best results in terms of span time and effort.

4.5.1.1 The effect of the profile of epistemic uncertainty reduction

In Figure 4-11 the performance of effort and span time of the PD project is shown for profiles of slow and rapid reduction in epistemic uncertainty (the upper and lower curves respectively in Figure 4-1) for sequentially dependent tasks. The data points on each individual curve show simulation results for zero, 50% and full overlap conditions. The data points are joined by a smooth curve for clarity in describing results of scenarios with the same parameters. These cases depict high values of stochastic uncertainty and frequent communication interval for all tasks (*NCI* =10%, where *NCI* is defined in Table 4-1, is the normalized interval of *work* progress in a task between communications of interim information).

For both cases shown, zero overlap yields the identical result with the longest span time and the least effort. Note that the span time of this point is less than unity. This is because communication work takes place in the *read* subtask whenever input information is received. Thus, even before the technical work begins in a downstream task with no overlap, some communication work is done. This effectively reduces the normalized span time to less than one. (Recall that the span time is normalized with the total average effort and that this includes the effort required for information exchange. Since some of the total effort in reading

input information begins prior to the start of the technical *work* in the downstream tasks, the normalized span time is less than unity even when there is no overlapping of *work*).

Following the lower curve for the case of rapid reduction in epistemic uncertainty, we see a small increase (3%) in effort and a 26% reduction in span time at the 50% overlap data point. Fully overlapping this configuration of tasks results in an additional 37% decrease in span time and another 12% increase in effort.

The upper curve for the case of slowly reducing epistemic uncertainty shows an increase in effort of 7% and a reduction in span time of 25% at the 50% overlap data point. Fully overlapping this configuration incurs a further 33% increase in effort and an additional 8% decrease in span time.

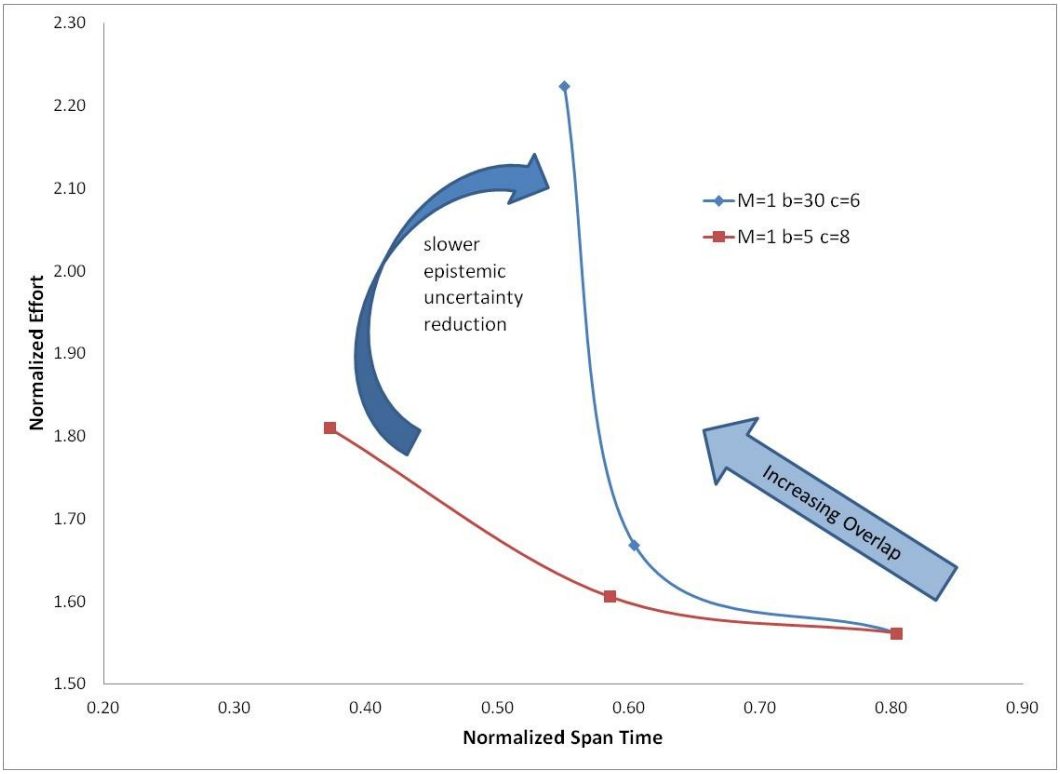


Figure 4-11 Normalized effort versus span time for two profiles of epistemic uncertainty reduction for overlapping of sequentially dependent tasks

Thus, we see that for tasks with fast reduction of epistemic uncertainty there is a substantial reduction in span time with fully overlapping tasks. This is evident even when there is a large magnitude of aleatory uncertainty and strong dependence between tasks, but as we shall see later, this does require frequent communication of interim information. This is not the case for tasks with slow reduction in epistemic uncertainty, where overlapping tasks by more than 50% has little additional benefit in span time and incurs a significant increase in effort.

Examination of Figure 4-12 shows that the reason for the span time and effort performance in these scenarios is the increasing amount of rework from progressive iteration that is generated during the simulations with higher overlap (for a more detailed discussion of rework see section 3.3.10). The sum of this rework over all tasks and phases of the simulated PD project is called *churn*. When uncertainty reduces more slowly, there is more churn generated with increasing overlap. Thus, the model predicts that when there is greater uncertainty tasks do not make progress, but spend more time reworking what was done with imprecise information. The additional effort in rework in the tasks results in increased effort to complete the project, and in a lack of progress because the additional time spent in rework increases span time.

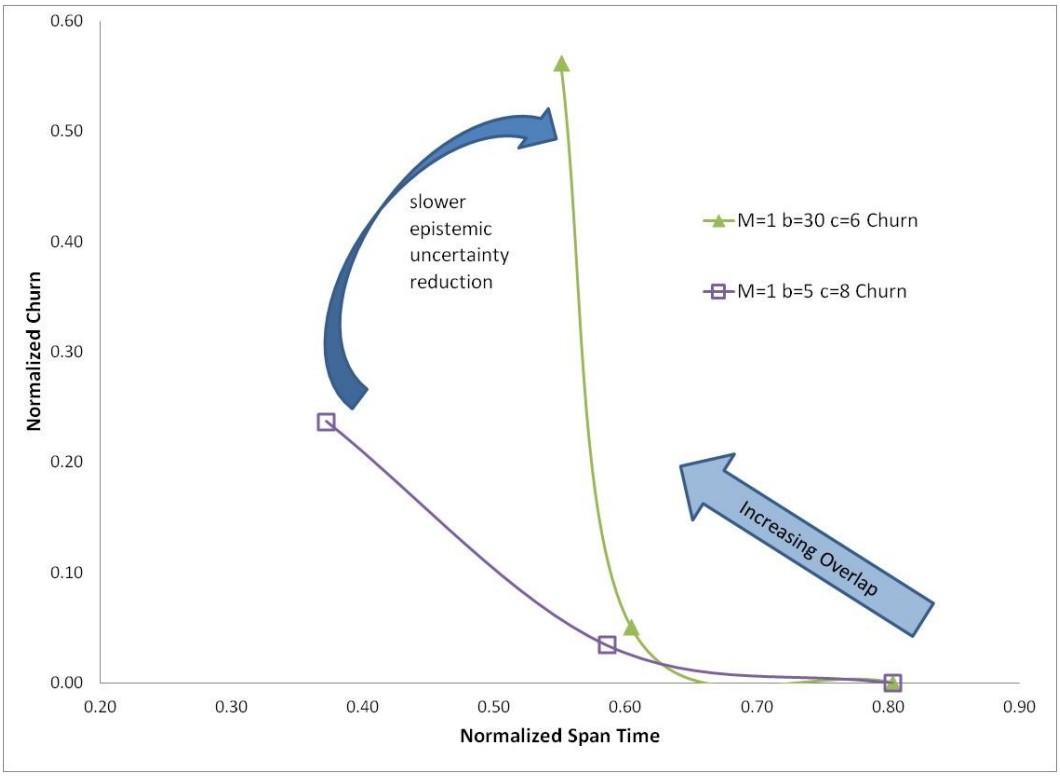


Figure 4-12 Normalized churn versus span time for two profiles of epistemic uncertainty reduction for overlapping of sequentially dependent tasks

4.5.1.2 *The effect of the magnitude of stochastic uncertainty*

The results in Figure 4-13 are for tasks with slow reduction in epistemic uncertainty (the highest curve in Figure 4-1 corresponding to b=30 and c=6) and several values of the magnitude of stochastic uncertainty m. At zero overlap, the span time and effort is the same regardless of the level of stochastic uncertainty. Overlapping at 50% results in similar reduction of span time and increase in effort for each value of m. This corresponds to a 25% reduction in span time and a 9% increase in effort.

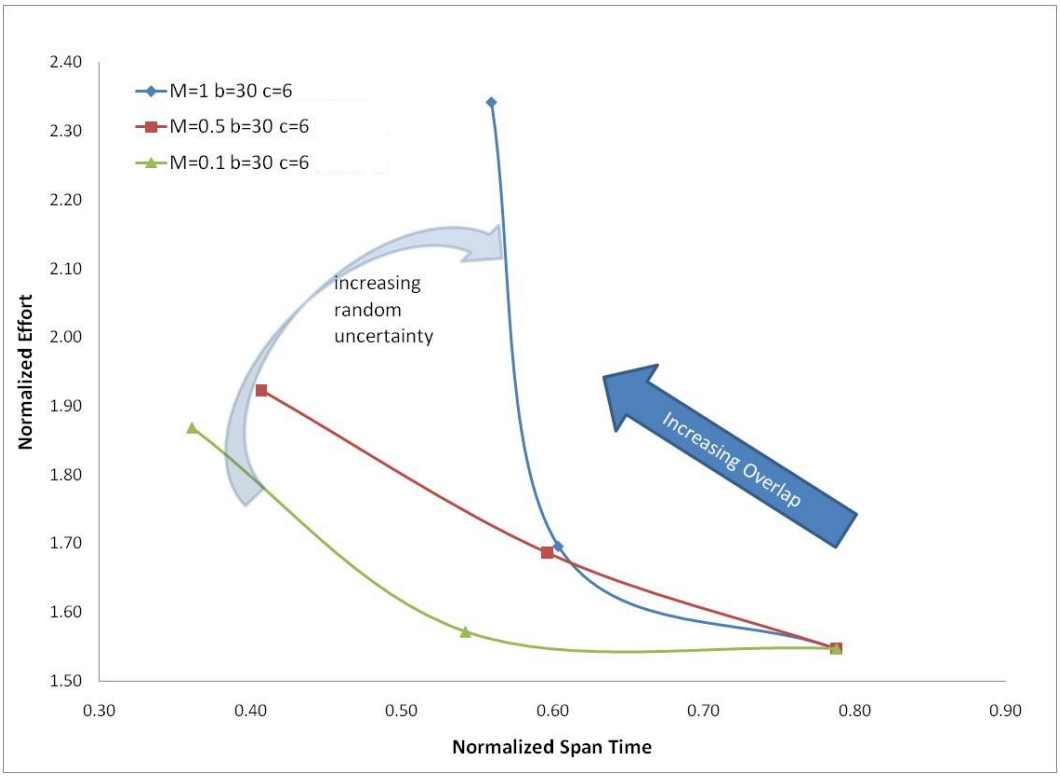


Figure 4-13 Variation of PD project performance of 5 sequentially dependent tasks with increasing overlap using the same slow reduction in epistemic uncertainty with increasing stochastic uncertainty

For fully overlapped tasks, there is a wide divergence with *m*, indicating an increasingly larger amount of effort and increasingly smaller reduction in span time when stochastic uncertainty is higher. This result is primarily due to the increasing amount of rework from progressive iteration that is generated during the simulations (for a more detailed discussion of rework see section 3.3.10). The variation of the sum of this rework over all tasks and phases of the simulated PD project (*churn*) is shown in Figure 4-14. Here, the substantial increase in churn occurring at full overlap is evident.

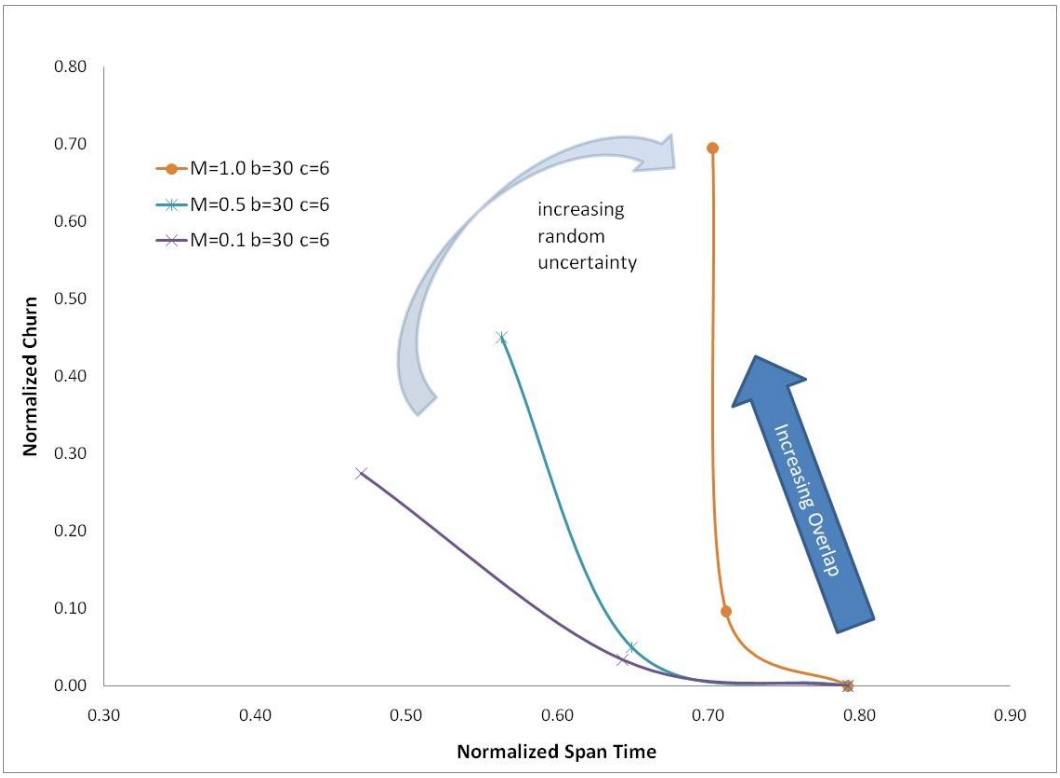


Figure 4-14 Normalized churn for the scenarios in Figure 4-13.

The criterion for starting the downstream task in the simulation is based upon the percentage of information received by this task from the upstream task. At 50% overlap the downstream task begins work when 50% of the information requirement from upstream tasks has been met. Thus, the uncertainty of the information received before the downstream task begins is relatively low, even in the case of high values of *m*, and so there is much less rework generated. Further reductions in span time are predicted with full overlap, but only in cases with relatively low values of stochastic uncertainty. If stochastic uncertainty is too high, there is much wasted effort in rework, and little benefit in span time reduction for full overlap.

These results with increased overlap are a direct outcome of the rework generated in the tasks. This is evident in Figure 4-14 which shows the value of churn or total rework generated by progressive iteration in tasks during each phase in the simulations. The increase in effort corresponds to the increase in churn when

overlap increases beyond 50%, and this results in increased span time as more time in each task is spent redoing work rather than making any progress towards completion of the project.

With zero overlap all cases of stochastic uncertainty show the same normalized span time and effort values at approximately 0.8 and 1.55 respectively.

Figure 4-15 shows the behaviour with overlap and magnitude of stochastic uncertainty when the epistemic uncertainty in the tasks reduces more quickly (the lowest curve in Figure 4-1). Here, we see than the span time reduction is higher and the increase in effort is lower at each instance of overlapping and magnitude of stochastic uncertainty.

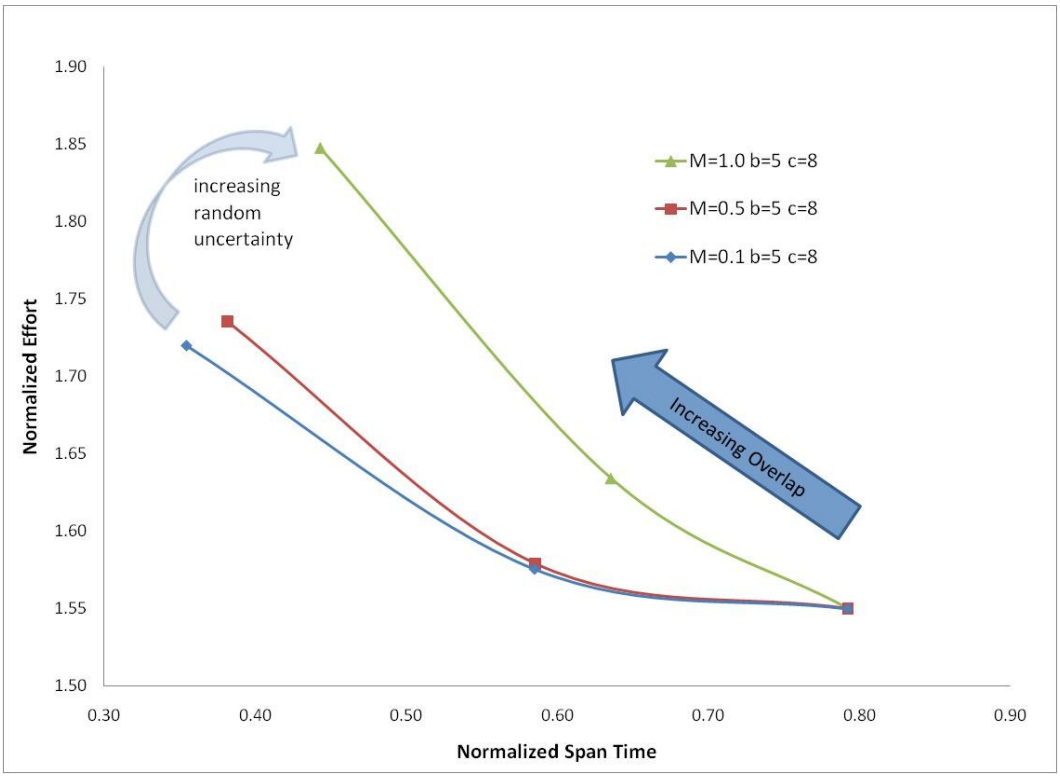


Figure 4-15 Variation of PD project performance of 5 sequentially dependent tasks with increasing overlap using the same rapid reduction in epistemic uncertainty with increasing stochastic uncertainty

Once again, this is due to the substantially lower amounts of churn generated in the simulation as can be seen by comparing the variation in churn shown in Figure

4-16 with those in Figure 4-14. Here, we see that the magnitude of churn is less than one-half of those at corresponding levels of overlap.

The behaviour of span time and effort of the simulated PD projects shows how overlapping tasks, where there is high sequential dependency between them, can be of benefit in reducing span time. The extent to which this is advantageous depends on the level of stochastic uncertainty in the tasks and on the rate of reduction of epistemic uncertainty. These simulations were performed with a communication interval of 10% of the nominal task time (*NCI*=0.10). In the next section, we look at the effect on PD project performance when this communication interval is varied.

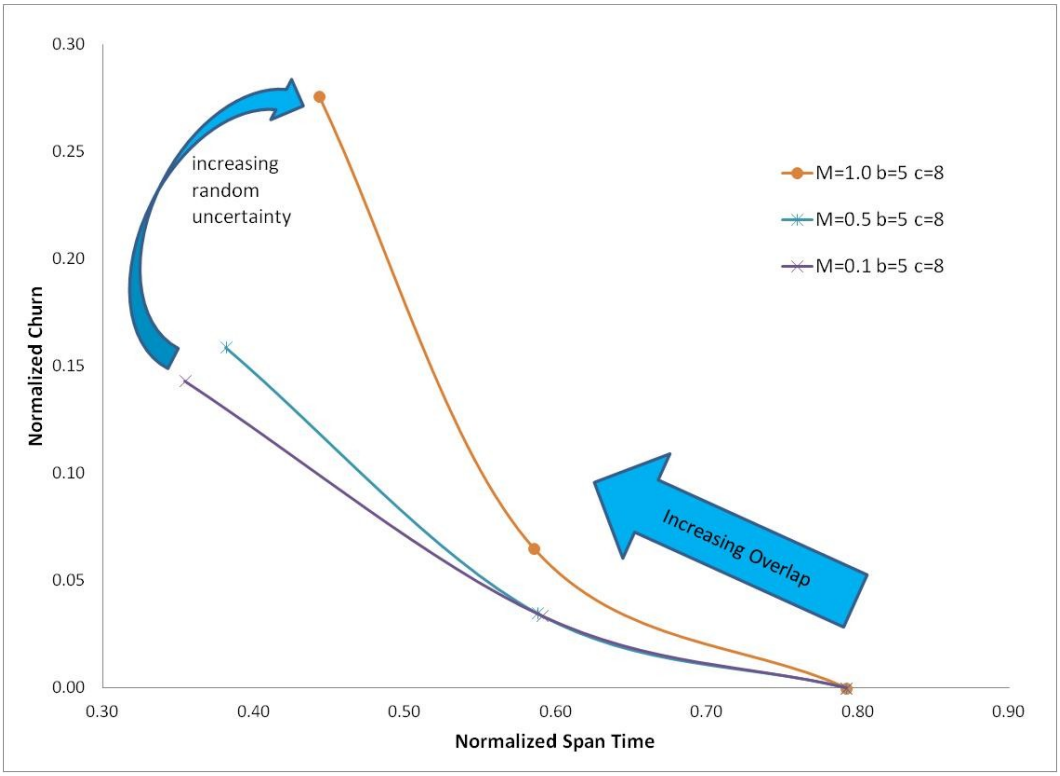


Figure 4-16 Normalized churn and span time for the scenarios shown in Figure 4-15.

The results shown here are qualitatively similar to those predicted in other research in concurrent engineering (Krishnan, Eppinger et al. 1997; Bhuiyan, Gerwin et al. 2004) using different kinds of process models. In a field study of concurrent engineering practices, Swink et al. (1996) found that companies tended

not to overlap manufacturing process planning tasks with design tasks for the development of products with a high degree of innovation or new technology in design. A high degree of innovation or new technology in design implies high uncertainty in design tasks, and therefore, this observation supports the findings here that overlapping downstream tasks is counterproductive when there is high upstream uncertainty.

Care must be taken in comparisons, however, since in the model presented here we differentiate between an initial level of uncertainty between tasks, which is used to construct a measure of the dependency strength between tasks and the information exchange requirements, and the rate at which this uncertainty reduces during the execution of the tasks. We show here, that even for high initial uncertainty between tasks, if the uncertainty reduces rapidly during task execution, overlapping with suitable frequency of communication of interim information can be an effective technique for reducing span time. We examine this relationship in the next section.

4.5.2 Effects of task overlapping and communication interval

The variation of span time with overlapping and communication interval is shown in Figure 4-17. These results are for slowly reducing epistemic uncertainty and an intermediate value of stochastic uncertainty. Here, we see the effects as we vary the value of *NCI* about the value of 0.1 used in the previous section. As can be seen in the figure, with zero overlap, the span time remains nearly constant with change in *NCI*. At 50 per cent overlap, span time rises as *NCI* is increased. With full overlap there is a minimum span time achieved with *NCI* at about 10 per cent and further increases in communication interval result in increases in span time.

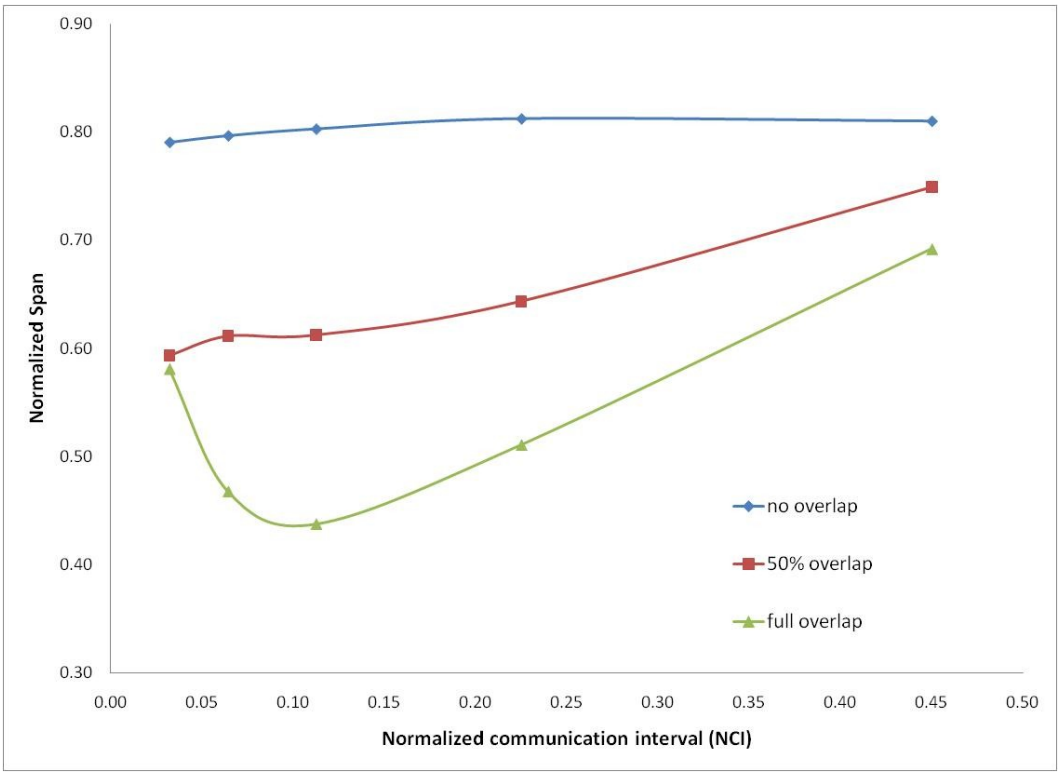


Figure 4-17 Variation of span time of 5 sequentially dependent tasks with communication interval for different amounts of task overlap. Cases shown for slow reduction in epistemic uncertainty and moderate magnitude of aleatory uncertainty (b=30, c=6, m=0.5).

Figure 4-18 shows how effort varies with communication interval. Effort is highest when *NCI* is small in the fully overlapped case, reaching a minimum at a value of *NCI* of approximately 10 per cent and then rising more gradually with increasing values of communication interval. Effort increases with more overlap.

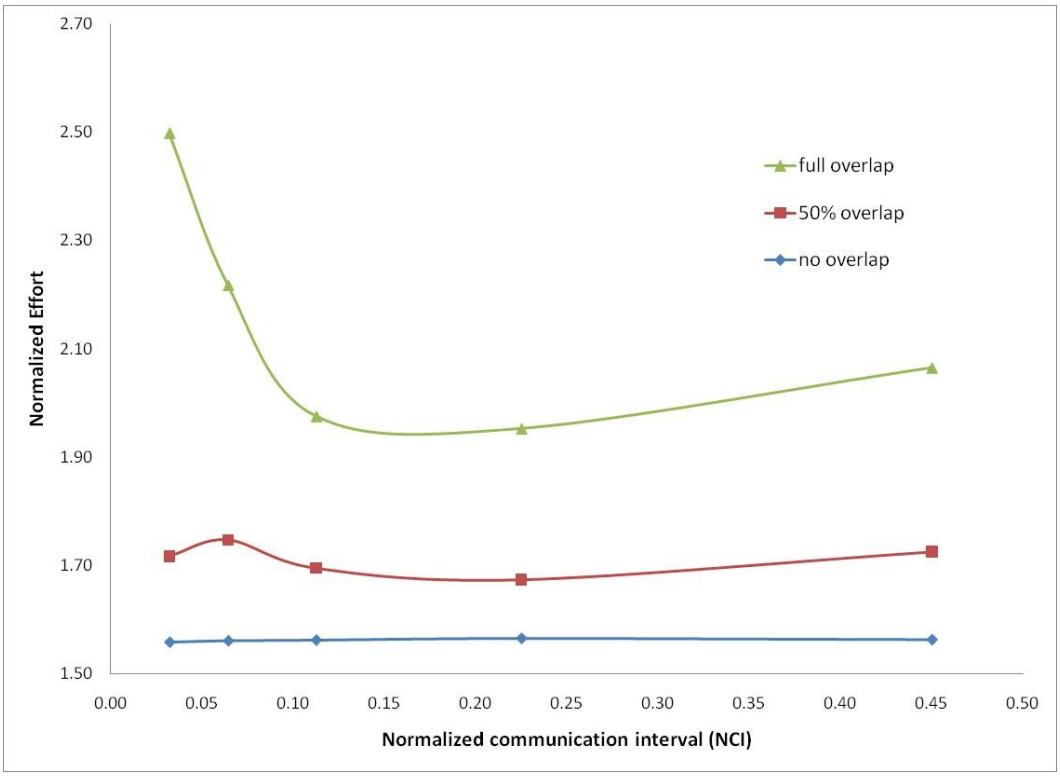


Figure 4-18 Variation of effort with communication interval for the cases shown in Figure 4-17.

The simulation results show that the higher effort and lower span time resulting with full overlap and very short *NCI* is caused by higher values of churn. When there is too frequent communication of uncertain information, tasks perform more rework. This is evident in Figure 4-19 where churn reaches a minimum at *NCI* of approximately 10 per cent and levels off substantially. When the communication is too infrequent, tasks are starved for information and span time extends largely due to tasks waiting for information. The downstream tasks wait, but there is less rework; so, the effort does not increase substantially.

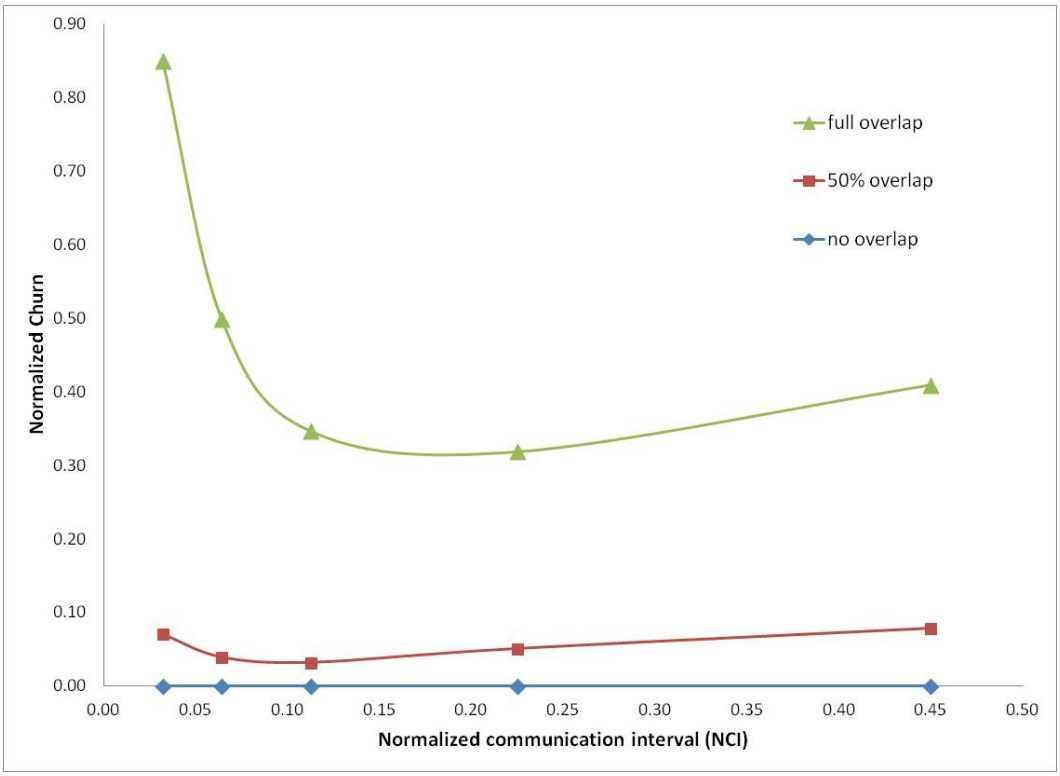


Figure 4-19 Variation of churn with *NCI* for the scenarios in Figure 4-17.

The simulation results for starve time shown in Figure 4-20 bear this out. Here, the tasks in an overlapped scenario are 'starved' for information as communication intervals increase and cannot progress for increasingly longer periods of time.

In a sequentially dependent scenario, downstream tasks must wait for information in order to perform their work. If the tasks are overlapped, there can be a reduction in span time even if dependency is high, provided that the rate of uncertainty reduction in the upstream tasks is sufficiently rapid, and that interim information is communicated with suitable frequency. Too frequent updates of information about design parameters results in the likelihood of excessive rework, whereas too infrequent updating of this information can starve the downstream tasks so that even if they begin work they are unable to continue, having gone as far as they can with the received precision of information.

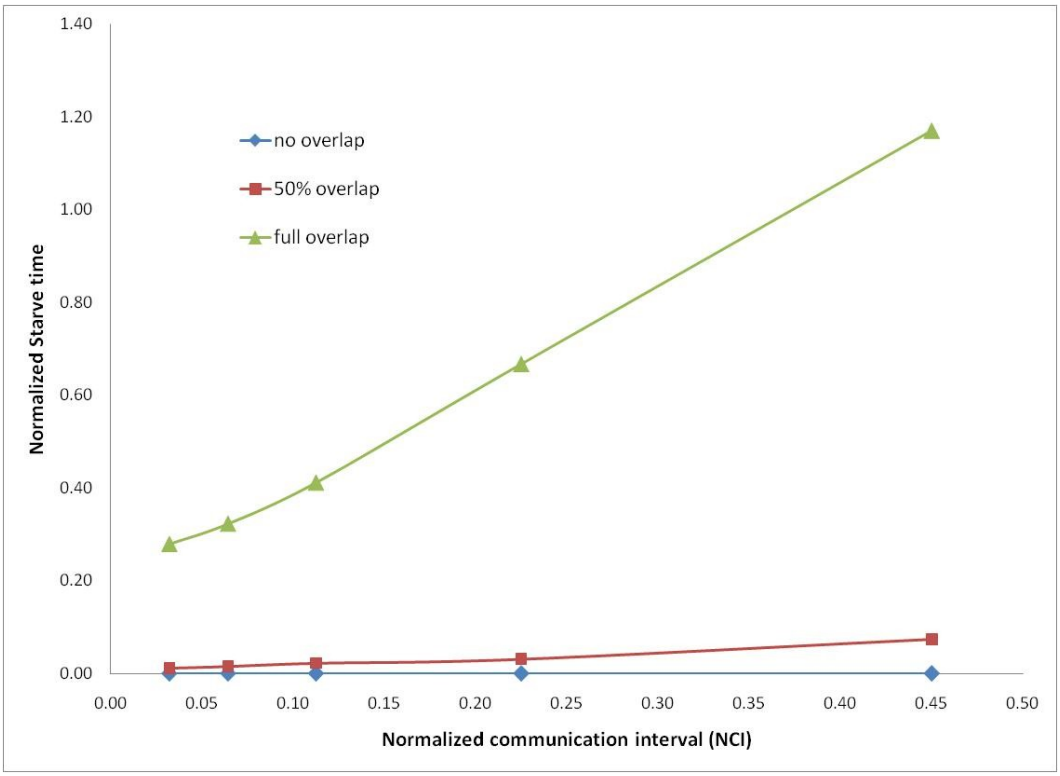


Figure 4-20 Normalized starve time versus *NCI* for the scenarios in Figure 4-17.

The simulation results shown in this section predict the optimal communication interval and the conditions under which overlapping of tasks is advantageous. The sensitivity of these results to the various parameters in the model are investigated next.

4.5.2.1 *Sensitivity of results*

Figure 4-21 shows the influence of increasing the magnitude of stochastic uncertainty m on the span time. The behaviour is similar to that shown for $m = 0.5$ in Figure 4-17, but in Figure 4-21 the value of *NCI* at which the minimum span time is reached is 'flatter', showing a range between 0.1 and approximately 0.25 with full overlap. However, in comparison to the results in Figure 4-17, the reduction of span time is smaller in Figure 4-21, reaching a minimum of 0.55 with $m = 1$ in comparison to the value of 0.44 when $m = 0.5$. The span time at very low *NCI* is also significantly higher with full overlap, reflecting the detrimental effect of too frequent communication of interim information with high stochastic

uncertainty. Similar effects are apparent with 50% overlap albeit with smaller magnitudes.

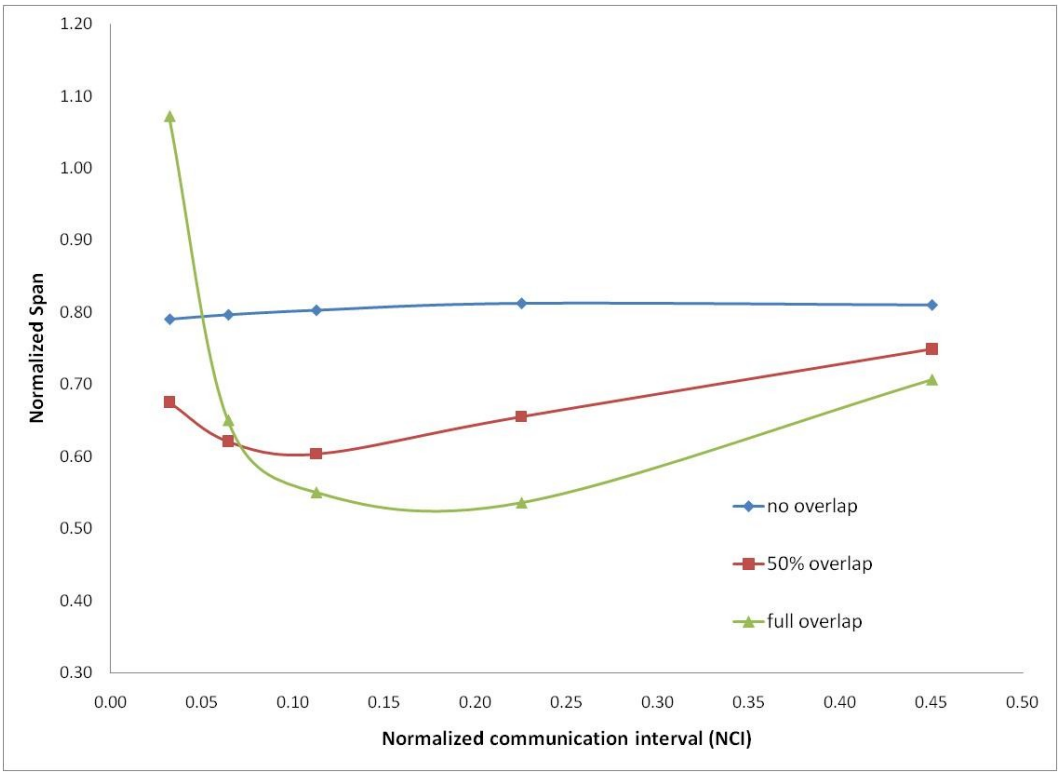


Figure 4-21 Variation of span time with communication frequency with slow evolution and high aleatory uncertainty for five tasks that are sequentially dependent (b=30, c=6, m=1.0).

In Tables 4-7 and 4-8, the actual values of normalized effort, span time, churn and starve time are shown for the m=0.5 and m=1.0 respectively with full overlap. Here, it can be seen that while the minimum span time is lower for lower stochastic uncertainty, the value of span time is approximately equal for higher *NCI*.

Table 4-7 Simulation results for the scenario in Figure 4-17

b	*c*	*m*	*neffort*	*nspan*	*nchurn*	*nstarve*	*nci*	*Overlap*
30	6	.5	2.50	0.58	0.85	0.28	0.03	Full
30	6	.5	2.22	0.47	0.50	0.32	0.06	Full
30	6	.5	1.98	0.44	0.35	0.41	0.11	Full
30	6	.5	1.95	0.51	0.32	0.67	0.23	Full
30	6	.5	2.07	0.69	0.41	1.17	0.45	Full

Table 4-8 Simulation results for the scenario in Figure 4-21.

b	*c*	*m*	*neffort*	*nspan*	*nchurn*	*nstarve*	*nci*	*Overlap*
30	6	1	3.59	1.07	1.88	0.53	0.03	Full
30	6	1	2.59	0.65	0.92	0.52	0.06	Full
30	6	1	2.22	0.55	0.56	0.56	0.11	Full
30	6	1	1.98	0.54	0.38	0.72	0.23	Full
30	6	1	2.10	0.71	0.44	1.17	0.45	Full

The values for effort are much higher at low values of *NCI* reflecting the effect of higher churn at this range, but also reach approximately the same values for higher values of communication interval.

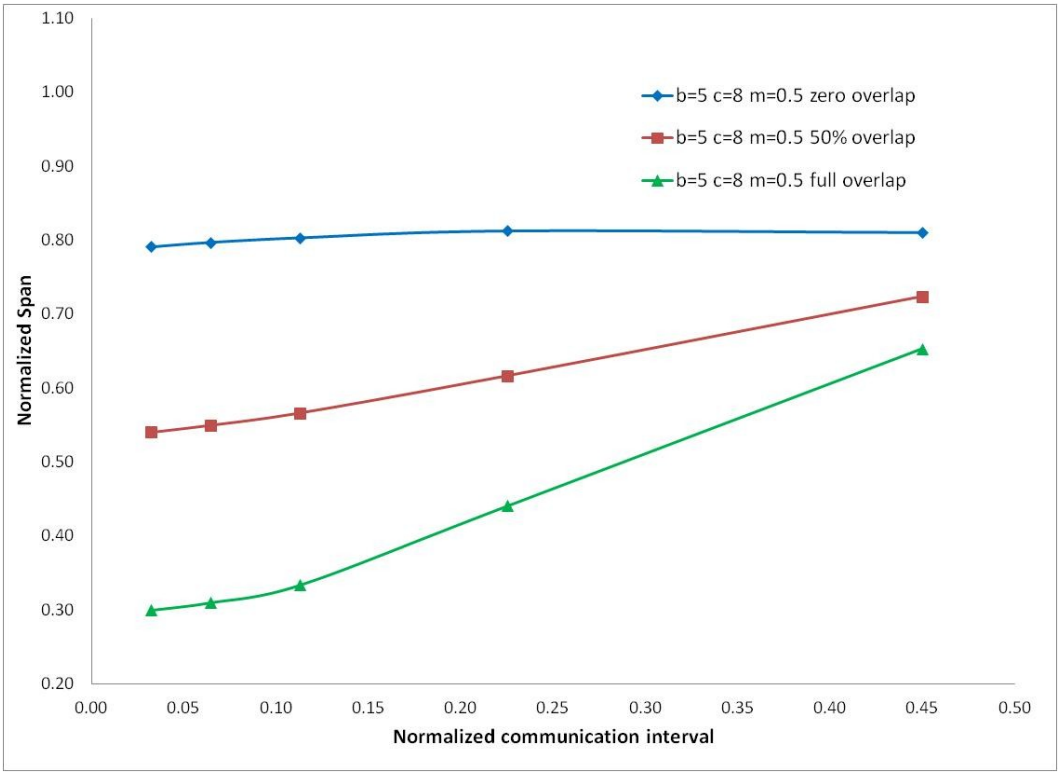


Figure 4-22 Variation of span time with communication frequency for rapid reduction in epistemic uncertainty and moderate aleatory uncertainty for five sequentially dependent tasks.

Variation of span time with communication interval for scenarios with rapid reduction in epistemic uncertainty is shown in Figure 4-22. In comparison with slow reduction of epistemic uncertainty (Figure 4-17) the following results are observed:

i. With zero overlap, the results are identical, showing constant span time of approximately 0.8.

ii. At 50% overlap, the span time is approximately 10 percent lower at small *NCI,* but rises with longer communication intervals to the same values achieved with slow reduction of epistemic uncertainty.

iii. With full overlap, the behaviour is quite different at low *NCI*; span time is not higher at low *NCI*, but rather increases gradually as communication interval is increased. The minimum span time is 30% lower with rapidly reducing epistemic uncertainty than in the scenario in Figure 4-17, and is achieved at the lowest values of *NCI*. This is consistent with the rapid reduction in epistemic uncertainty, where the final values of parameters communicated are reached very rapidly.

4.5.3 Modification of dependencies with multiple inputs (set-based coordination)

Set-based coordination was put forward as a hedging tactic that can be employed to reduce span time in situations where the cost of addressing multiple upstream task outcomes is not prohibitive (Loch and Terwiesch 2005). In essence, set-based coordination is an approach in design where engineers develop several concepts to greater resolution before choosing one to finalize (Sobek 1996). As an illustration consider the situation where a task cannot proceed without vital information from an upstream task, and it is decided to develop several alternatives, one for each set of possible results from the upstream task. This, in effect, eliminates the dependency with the upstream task, but comes at a cost of having to develop several solutions.

We looked at simulating this scenario using the model. In our 5 task sequentially dependent project, we consider the effects of developing two solutions for task 4; one we call task 4a and one task 4b. These each carry out the work of the previous task 4, but for one of two possible outcomes we conjecture that could be provided by task 3 (outcomes of task 3 are the inputs to task 4). Table 4-9 shows the resultant dependency matrix considered.

Table 4-9 The dependency matrix for a set-based coordination scenario

TASK	1	2	3	4a	4b	5
1	9	9	9	9	9	9
2	0	9	9	9	9	9
3	0	0	9	0	0	9
4a	0	0	0	9	0	9
4b	0	0	0	0	9	9
5	0	0	0	0	0	9

Thus, there is no dependency between task 3 and task 4a or task 4b and no dependency between task 4a and 4b. The **WK** matrix with the parameters for the *work* subtask PDF is the same as in equation 4-8, but with an additional row for the additional task added to the project. In this scenario, therefore, the total *work* is greater due to the additional task, and is 18,000 hours. However, we want to keep the total amount of communication work the same as previously at 15,000 hours; so, $\boldsymbol{\alpha}$ is set accordingly using equation 3-8. The resultant communication matrix **NC** is shown in Table 4-10.

Table 4-10 The communication matrix for the set-based coordination scenario

TASK	1	2	3	4a	4b	5
1	46	46	46	46	46	46
2	0	46	46	46	46	46
3	0	0	46	0	0	46
4a	0	0	0	46	0	46
4b	0	0	0	0	46	46
5	0	0	0	0	0	46

The effort versus span time for this scenario for various amounts of task overlap is shown in Figure 4-23 in comparison to the earlier dependency conditions with high uncertainty. Here, rather than compare normalized effort and span time we show non-normalized results in hours so that we can more readily compare the results for these two situations. This is because the amount of work in the project has increased in the set-based scenario and so the number with which we would

have normalized the results is not the same for both cases. These cases were run at *NCI*=10%.

Here, we can see that in the zero overlap condition the set-based scenario has a 20% lower span time than in the base case scenario and 6% additional effort. With 50% overlap, there is 17% lower span time and about 6% more effort for the set based scenario. At full overlap, the span time is about the same for each scenario, but there is 17 percent higher effort in the set based condition. At zero overlap, the span time is lower since tasks 3, 4a, and 4b begin once the information from tasks 1 and 2 has been received without any dependency between them. This, in effect, allows tasks 3, 4a, and 4b to work in parallel even in the 'zero overlap' condition. It is actually the other tasks in the project that are affected by the imposed overlapping, since they are the only tasks with dependencies. The results here show that the overlapping of tasks has more rapidly diminishing benefits with increasing overlap compared to the base case.

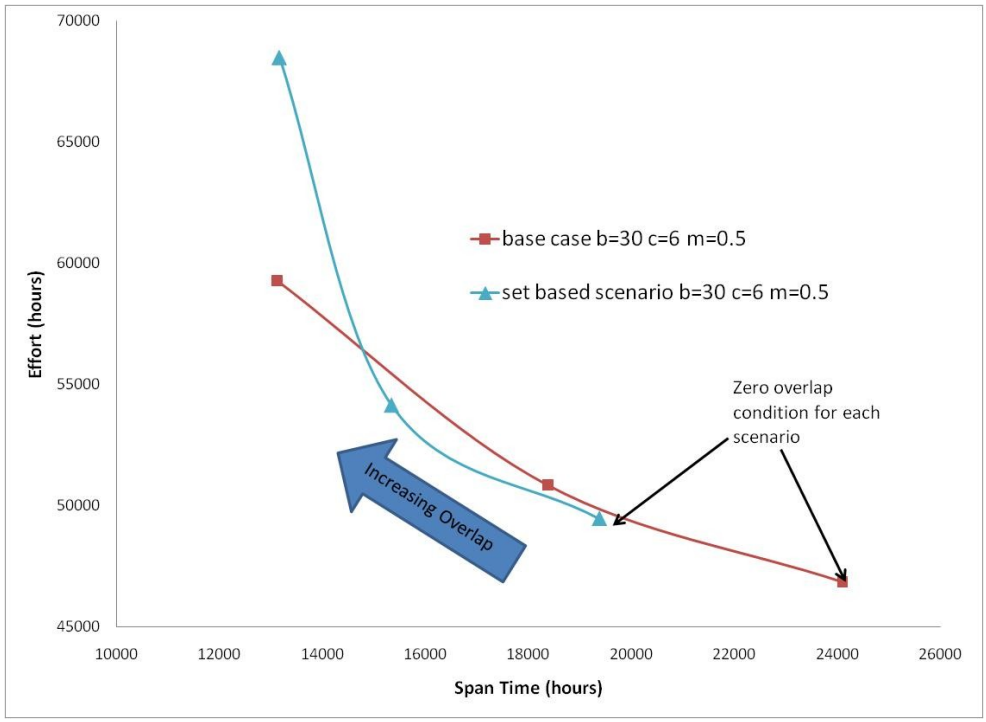


Figure 4-23 Effort and span time results with increasing overlap for a set based coordination scenario in comparison to ordinary sequential dependency conditions

This is supported by the results shown in Figure 4-24, which shows a more rapid rise in churn with overlap than the base case.

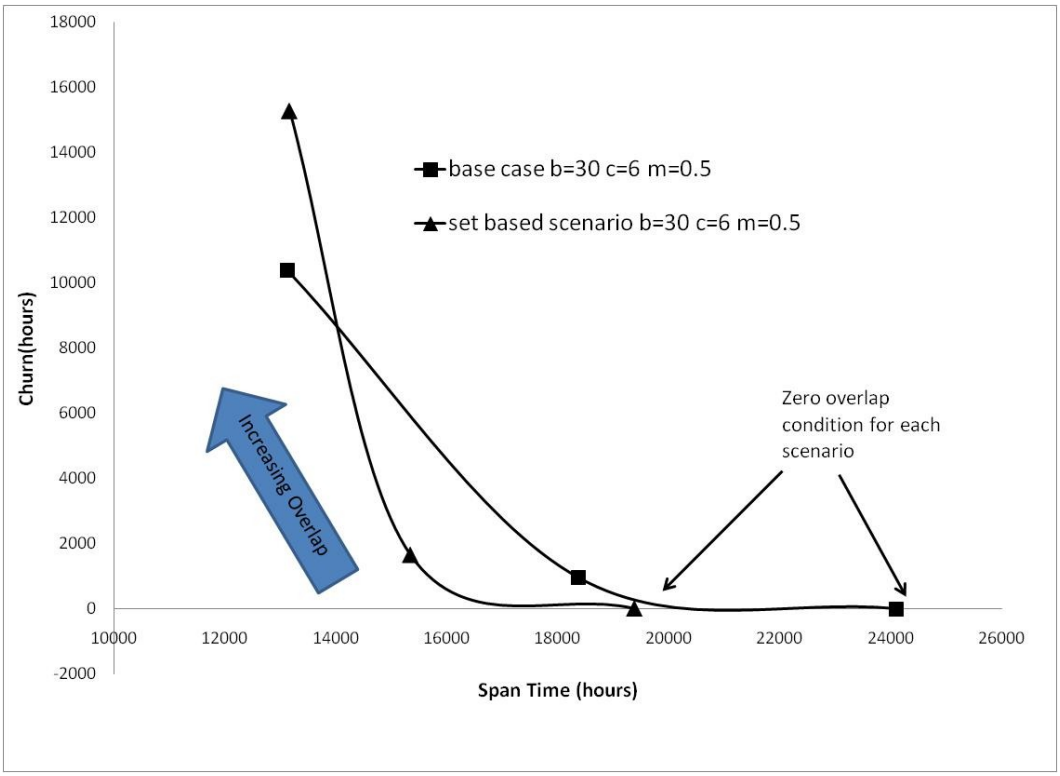


Figure 4-24 Churn for set based coordination with overlap in comparison to sequentially dependent scenario

The set-based coordination scheme can work effectively with overlapping to reduce span time in PD projects where uncertainty conditions are suitable. However, even with more rapid rates of uncertainty reduction there is more benefit to fully overlapping than to fully overlapping with set based coordination. This is evident in Figure 4-25 where the reduction in span time with full overlapping is the same in both the base case and the set-based coordination case, but there is greater effort required in the latter.

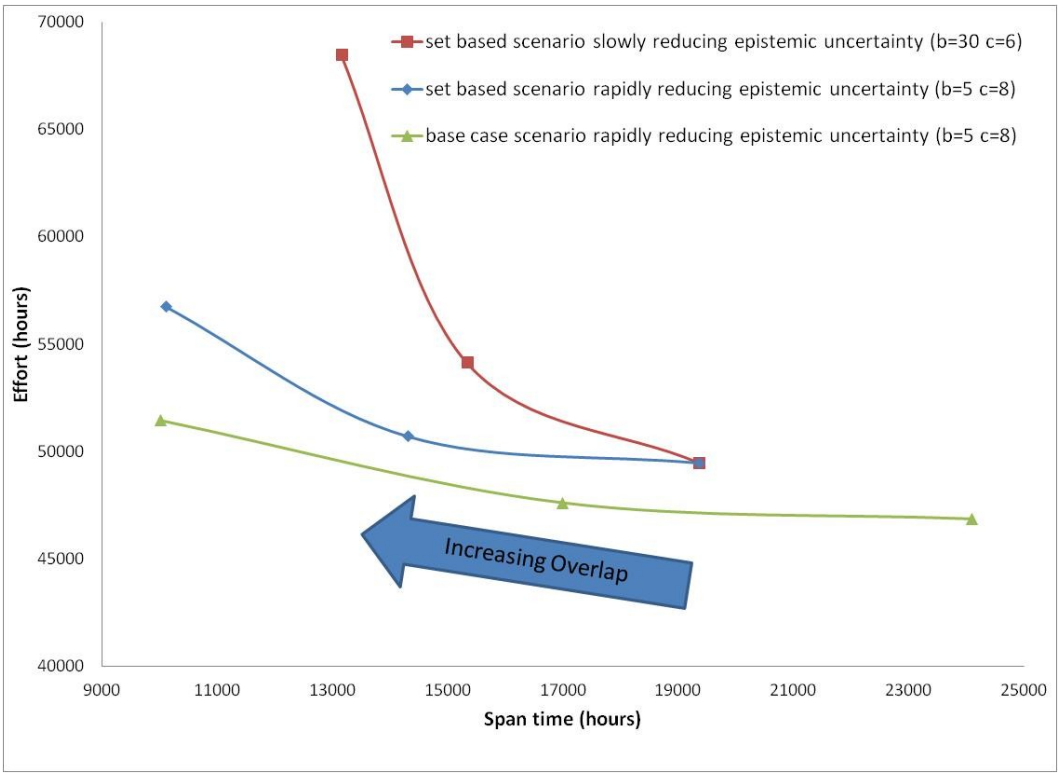


Figure 4-25 Comparison of set-based coordination and base case with different epistemic uncertainty reduction profiles and medium aleatory uncertainty (m=0.5)

Of course using the set-based coordination scheme is only possible when the sets of inputs can be defined for any eventuality of results of the upstream task whose dependency links are severed, and when the number of alternative solutions that must be generated is small. In Figure 4-26 and Figure 4-27 set-based coordination scenario 2 is shown where task 4 in the original base case would have had to be split into 3 alternate cases in order to eliminate the dependency with task 3. The results show that for both rapid and slow epistemic uncertainty reduction, it is not worthwhile to use this coordination scheme since it is possible to obtain an equal span time reduction with less effort simply by overlapping the original tasks by 50%.

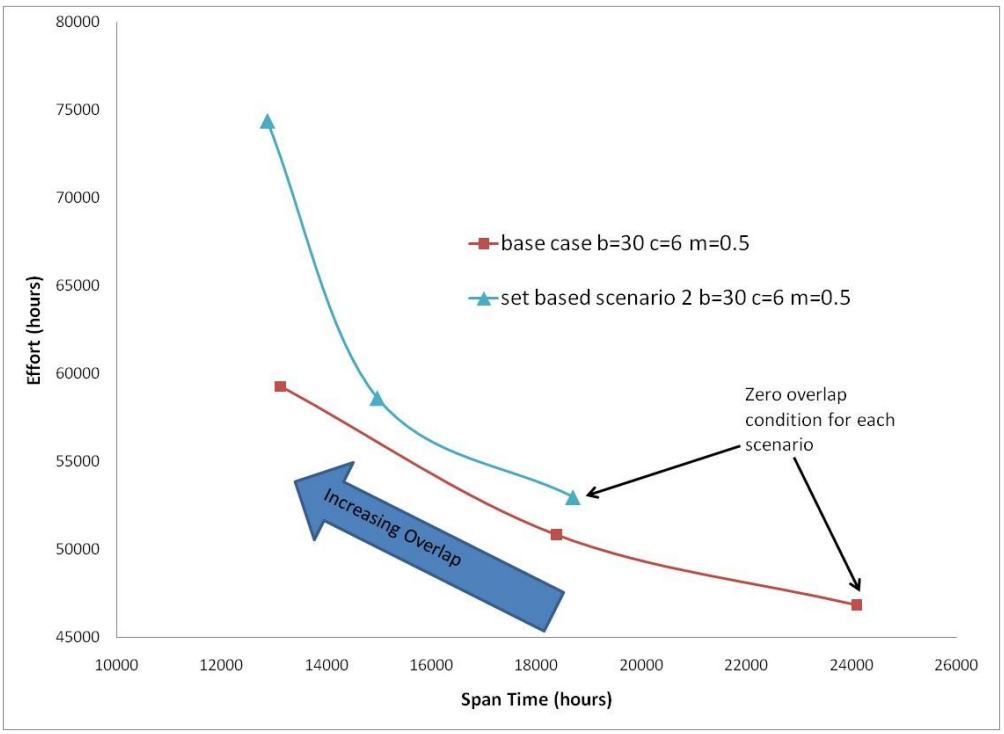


Figure 4-26 Set-based coordination scenario 2 with slow uncertainty reduction profiles

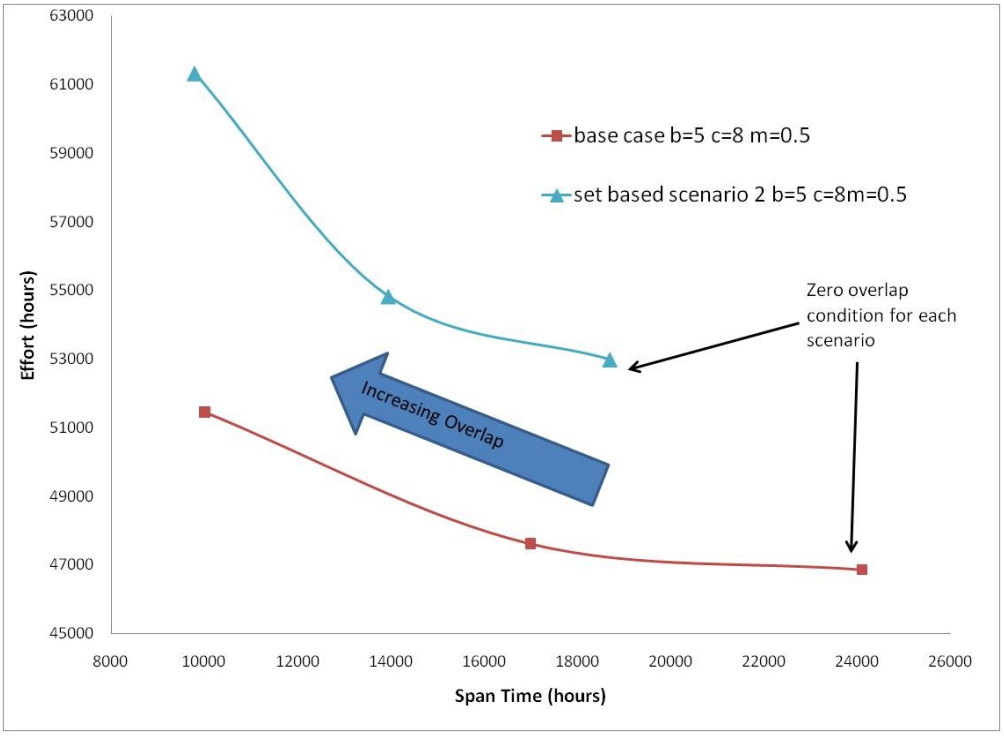


Figure 4-27 Set-based coordination scenario 2 with rapid uncertainty reduction profiles

4.6 Reciprocal dependency

Often in PD processes there are several groups or blocks of tasks that are reciprocally dependent (see section 2.8.1). Efforts to coordinate the work of these blocks of tasks most effectively have the most leverage in improving the overall performance of the PD project since it is from these blocks of tasks that much of the rework originates. In this section we look at various means of improving the coordination of groups of tasks with reciprocal dependencies.

In PD processes, reciprocal dependency often exists between tasks, although it is sometimes underestimated or ignored and results in unplanned rework cycles and longer span times. As an example consider the scenario where a product is designed and production documents are handed over to manufacturing. The process planning for manufacturing may uncover problems with the design where, for example, some parts are not able to be manufactured as designed due to limitations in manufacturing processes. These limitations might not be known to designers whose expertise is limited in manufacturing technologies or if the product or the manufacturing processes are new to their experience. Additionally, manufacturing process planning tasks may uncover substantial savings that can be gained by changes to the design that are of no significance to the customer requirements, but can have significant cost impact in manufacturing.

If these issues are important enough, manufacturing does not proceed until design changes are made. In effect, rework of design tasks is performed due to information received from downstream tasks in the PD process, and this happens because the dependency between manufacturing process planning tasks and design tasks was underestimated. This is an example of reciprocal dependency because the manufacturing process planning tasks require information from engineering design tasks, and engineering design tasks require information from manufacturing process planning.

This is also an example of the role of uncertainty in the dependency strength between tasks. If uncertainty, that is, the difference between the designers'

knowledge of the manufacturing processes and those required for this product, is high, there is a higher dependency between the tasks involved in manufacturing process planning and the tasks involved in product design.

In situations of reciprocal dependency, if uncertainty is low in some tasks, estimates of information required from them can be used to make progress in other tasks first. These tasks with greater uncertainty become the 'upstream' tasks in the process. They would be developed to the level of detail where their uncertainty becomes low enough to be comparable to that of the downstream tasks. This may be thought of in terms of the precision of the information that can be generated from them without further developing the downstream tasks.

An example of this sequencing of tasks according to the degree of dependency can be taken from aircraft design, where the information required from the design of interior systems is less uncertain at the outset than that of the overall aircraft configuration or external contours. In aircraft design, initial estimates are made of the weights and topology of interior structures and systems, such as air distribution, cockpit instrumentation, landing gear, electrical systems, etc., based on previous experience and trade data. Using these data, the exterior shape of the aircraft is designed according to the mission requirements, loads generated by the motion of the aircraft in the air, and the weights of the payload and equipment the aircraft is carrying. Once the exterior contours are largely fixed, design of the interior systems can proceed in greater detail within the external envelope and allowable weights. It is in the detailed design phase of these internal systems that the required level of precision increases and the level of uncertainty of information to this precision becomes significant for many of the components and systems. Once this stage of the design process is reached, there is more reciprocal dependency when, for example, precise dimensions of structures and systems require more precise details of adjacent structures and systems to complete the final design of parts and components.

One way to manage reciprocal dependency in order to minimize rework and thereby reduce span time in a PD project is to perform the tasks that are reciprocally dependent in parallel and exchange interim information developed in each task. In this way each task can proceed with some initial estimates of the input information that it requires, develop some output information required by other tasks, communicate it, receive input from other tasks that is more precise, use this to produce output of greater precision, and so on. Thus, each task can make progress in refining its results. This iterative process continues until the information being exchanged is sufficiently precise for the purposes of the PD project.

In this section we use the model to explore several aspects of reciprocal dependency in PD. We consider the base case defined by the input parameters in Table 4-11 along with the dependency matrix in Table 4-12 for reciprocally dependent tasks:

Table 4-11 Input parameters for reciprocal dependency scenarios

Input parameter	Value
α	1.0
BR	Each element 0
B	Each element 0
C	Each element 1
INTMAX	20
LF	0.2
MINT	15
MW	1200
NIQ	0.0002
NDT	Each element 0.01
NLAT	Each row identical (0.0005, 0.0001)
PRD	Each row identical (0.5, 0.1)
PR	Each row identical (1, 4, 9)
QMP	Each element 1
QMR	Each element 1
RD	Each row identical (1, 4, 8)
ATD, STD	0.5, 0.1
STSD	0.4
SCH	Each element 11

Input parameter	Value
TRHD	Each element 0.10
WK	Each row identical (WKmax/4,WKmax/2, WKmax)*

*Note that WKmax is derived from the input value of *TWK* using equation 4-4.

Table 4-12 The dependency matrix for 5 tasks with high reciprocal dependency

TASK	1	2	3	4	5
1	9	9	9	9	9
2	9	9	9	9	9
3	9	9	9	9	9
4	9	9	9	9	9
5	9	9	9	9	9

4.6.1 The effects of communication interval

With high reciprocal dependency, parallel execution with interim information exchange is an important method of enabling dependent tasks to effectively continue with their work. The timeliness with which interim information is exchanged allows each task to make progress towards a successful design review at the end of each phase. However, exchanging uncertain interim information too often can lead to unnecessary rework which impedes the progress of tasks.

In this section we examine the effect of the communication interval with which a task sends out interim information to other tasks. For these simulations we considered PD projects with 5 highly reciprocally dependent tasks and average total effort requirement *TWK*=16,000 hours. The resultant information exchange matrix **NC** had all elements of value 36. Other input parameters were as indicated in Table 4-11.

Figure 4-28 shows the variation of normalized span time with normalized communication interval for several cases of uncertainty. The results show that with slowly reducing epistemic uncertainty (b=30, c=6) span time reduces with smaller communication intervals until a minimum is reached at *NCI* approximately equal to 0.1 (the critical point). The reduction in span time with

more interim communication is nearly 75% to the critical point. The steepness and magnitude of the rise of span time as *NCI* reduces below the critical point is strongly affected by the value of aleatory uncertainty *m,* indicating how the magnitude of the variation of stochastic uncertainty affects the span time when information is communicated with high frequency. When *NCI* is above the critical value, a tenfold increase in the value of *m* has little effect on the slope of increase in span time or its magnitude .

With more rapid reduction in epistemic uncertainty (b=5, c=8), there is a 35% increase in the span time over the range of *NCI*. The minimum span time with more rapid reduction in uncertainty is 36% lower than that with more slowly reducing uncertainty.

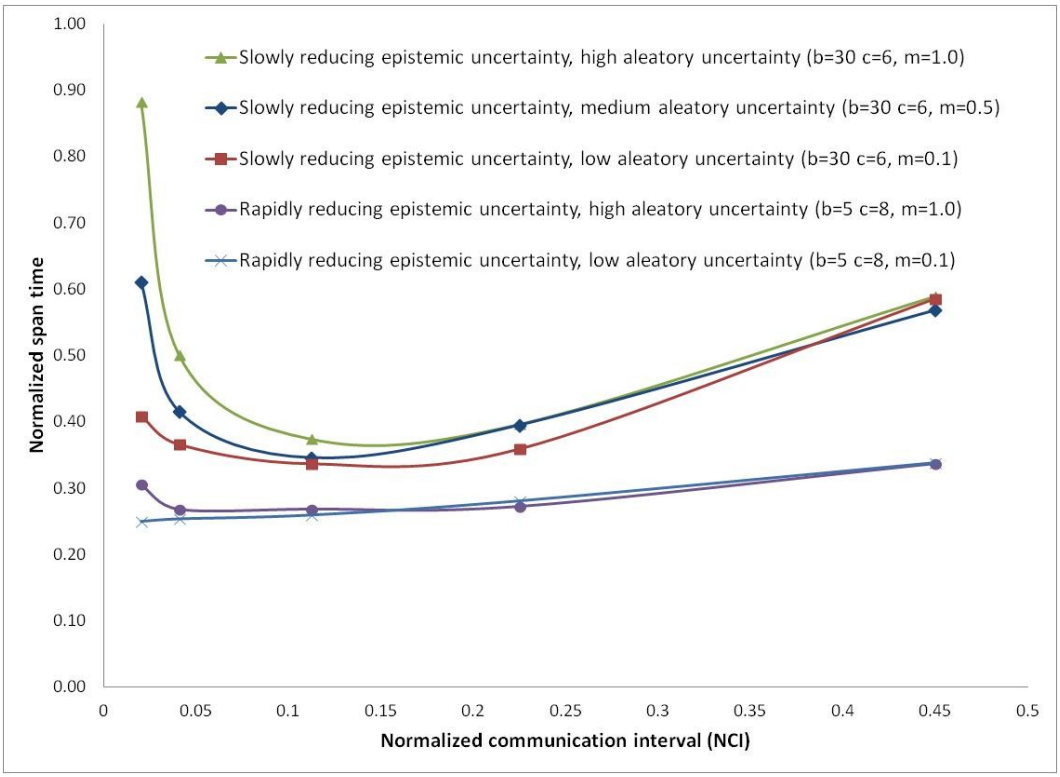


Figure 4-28 Normalized span time for PD with 5 reciprocally dependent tasks with varying communication interval for different cases of epistemic and aleatory uncertainty

The variation in effort shown in Figure 4-29 is similar to that in span time. Examination of the results shown for churn, starve time and design version rework in Figures 4-30 to 4-32 shows that the behaviour of the PD project is

largely affected by the increase in starve time when *NCI* is higher than the critical point. This results in an increase in design version rework.

The starve time increase is a result of tasks not getting enough information to allow them to progress in their *work* when the communication interval is too high. This causes an insufficient reduction in uncertainty when the design review takes place and triggers design version rework cycles.

This can be seen in Figure 4-31, where the number of design versions increase with increasing *NCI* past the critical point, but remain low below it. Increasing amounts of design version rework is the mechanism for increases in the effort and span time of the project when *NCI* is above the critical point.

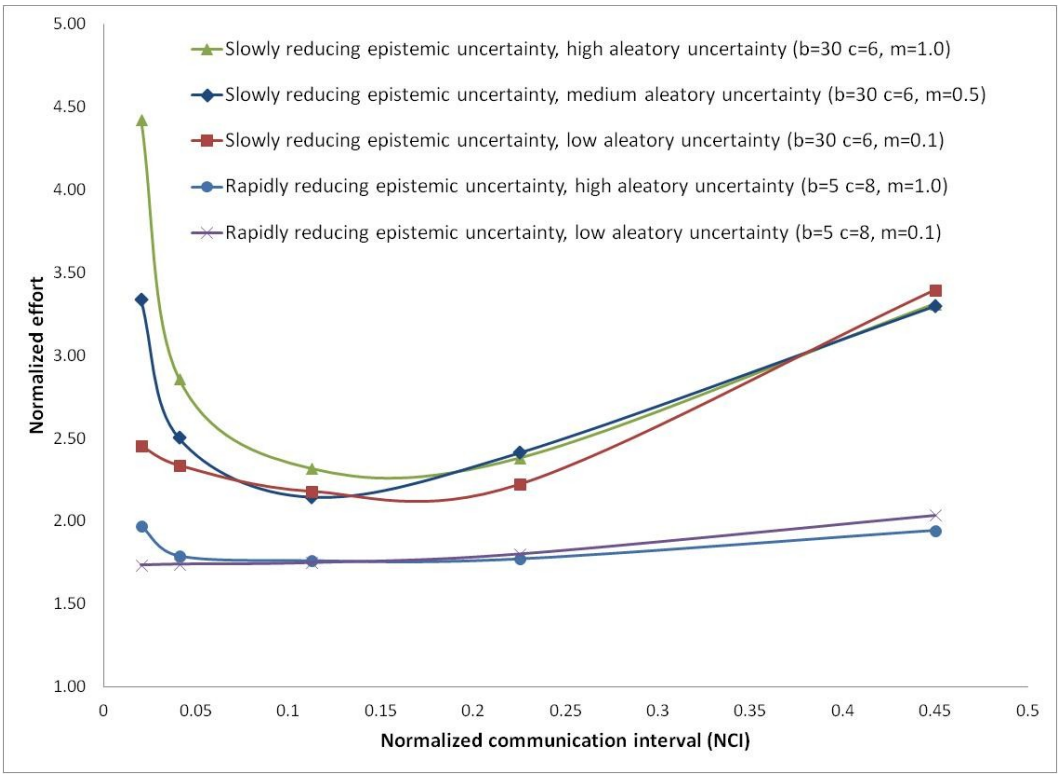


Figure 4-29 Normalized effort for PD with 5 reciprocally dependent tasks with varying communication interval for different cases of epistemic and aleatory uncertainty

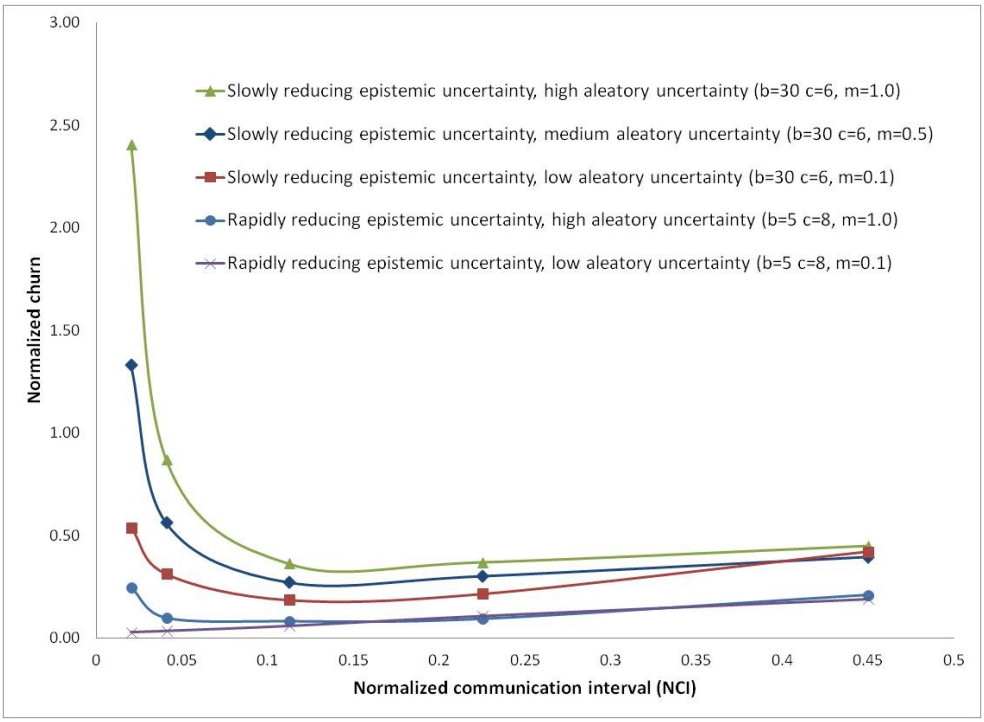


Figure 4-30 Cumulative normalized churn for PD with 5 reciprocally dependent tasks with varying communication interval for different cases of epistemic and aleatory uncertainty

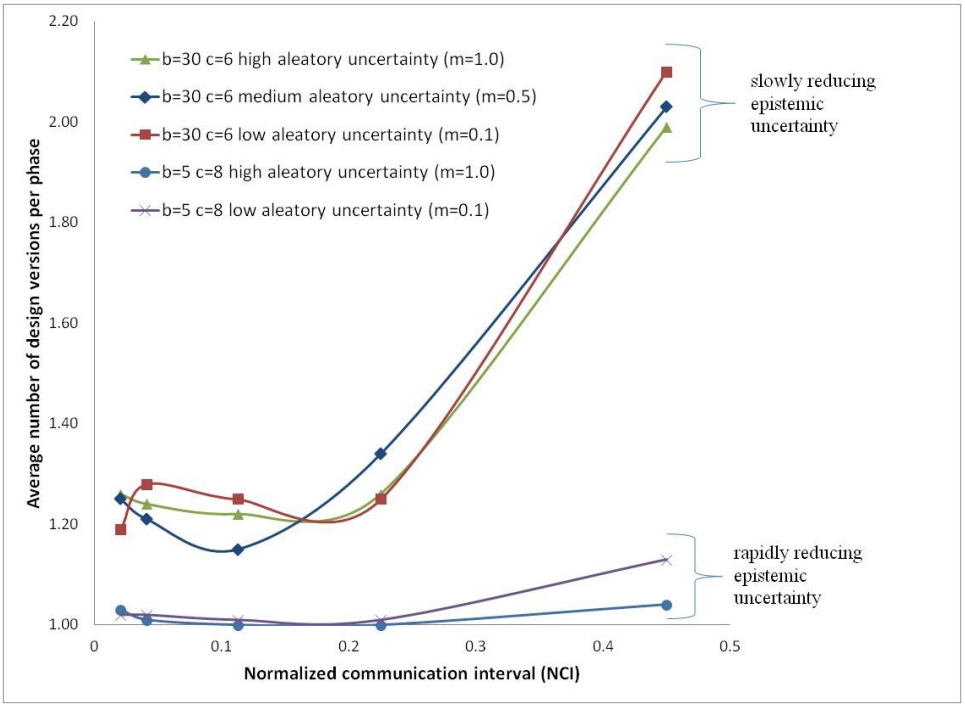


Figure 4-31 Average design versions per phase for PD with 5 reciprocally dependent tasks with varying communication interval for different cases of epistemic and aleatory uncertainty

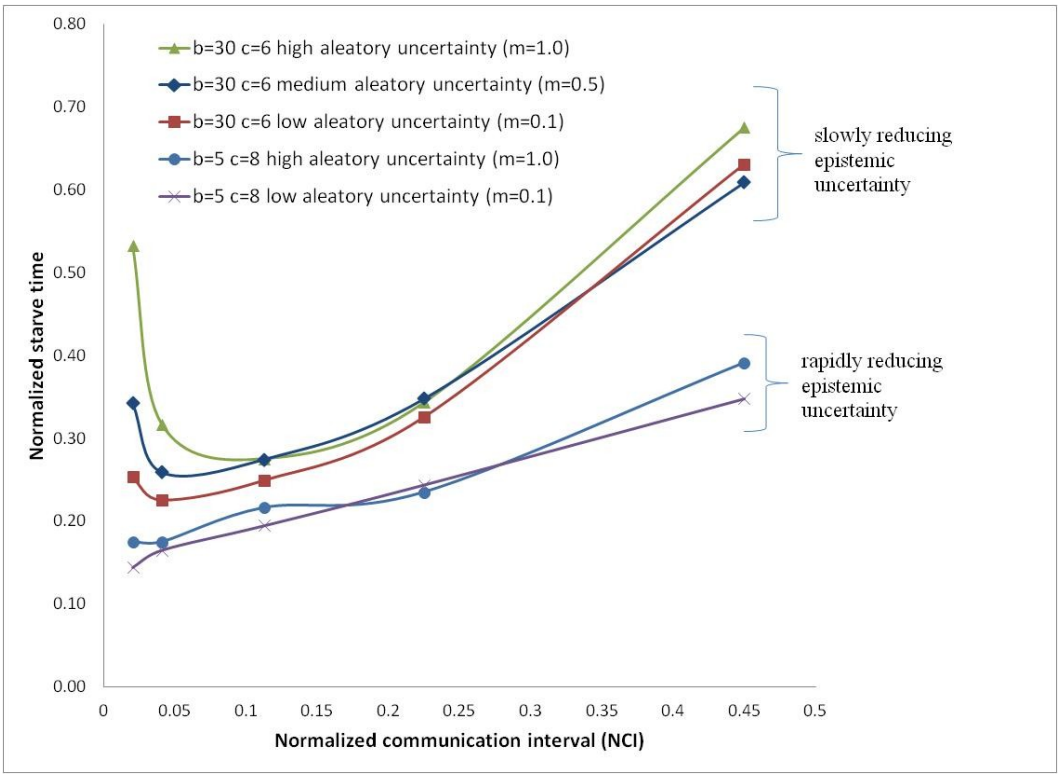


Figure 4-32 Cumulative normalized starve time for PD with 5 reciprocally dependent tasks with varying communication interval for different cases of epistemic and aleatory uncertainty

When *NCI* is below the critical point, it is the increase in churn that drives the span time and effort performance. As can be seen in Figure 4-30, there is a marked increase in churn below the critical point whereas churn remains flat with increasing *NCI* above it. This supports the explanation for increased span time caused by the rework from too frequent communication of interim information that is subject to stochastic uncertainty.

4.6.2 Effect of delays in information flow on project span time

In projects performed by many people in various locations, delays in simply getting information to the attention of those that can make use of it in their work can be a significant portion of the time required to do the work itself in each task. These delays, where information must travel through several layers of an organization, accumulate and can cause unnecessary rework with significant knock-on effects on many tasks. Moreover, since engineers and designers are often occupied with several projects at the same time, there is a delay before they turn their attention to following up on missing information required to make progress in one of their tasks. These delays to information flow are referred to as communication latency.

Various schemes to reduce communication latency are implemented in organizations ranging from project management personnel manually monitoring the status of deliverables of each task, the use of product data management systems, to the use of virtual models of the parts and assemblies of the new product being developed. These systems, if well designed, can reduce communication latency as well as facilitate the work in communication of interim information. However, a system to reduce latency, if not tailored to deliver interim information only to the people requiring it and if not able to indicate the precision of the interim information can be counterproductive. Too much information, too often, can result in wasted effort in communication and propagate imprecise and unstable data causing unnecessary rework.

The behaviour of the PD system as modeled here was used to study the impact of communication latency of various types on system performance. Simulations evaluate the impact of methods to reduce these delays for various product development system structures and levels of uncertainty. In order to examine the sensitivity of the span time of a project to these delays, permutations of the base case of Table 4-11 were simulated varying each input parameter affecting communication latency in turn.

The values for normalized span time in Table 4-13 are listed for each value of magnitude of aleatory uncertainty *m* shown in the top row, and for scenarios where the input parameter indicated in the left column was changed in turn from the base case of Table 4-11. These results are also illustrated in Figure 4-33. The combination scenario combined the indicated input values for each parameter together into one scenario.

Table 4-13 Values of normalized span time for each permutation of input parameter for slowly reducing epistemic uncertainty (b=30, c=6)

m	NSPAN @*m*=1.0	NSPAN @*m*=0.5	NSPAN @*m*=0.1	Δ @*m*=1.0	Δ @*m*=0.5	Δ @*m*=0.1
Base case (Table 4-11)	0.37	0.35	0.34	0	0	0
INTMAX=4	0.39	0.35	0.33	2%	1%	0%
NDT=0.1	0.39	0.40	0.37	2%	5%	4%
NLATM=0.1	0.44	0.41	0.37	6%	7%	3%
NIQ=0.1	0.42	0.37	0.35	4%	2%	2%
Combination scenario	0.70	0.69	0.65	32%	34%	32%

The span time generally increases with increasing magnitude of aleatory uncertainty in each scenario. The differences between the normalized span time for each scenario and those of the base case are shown in the three right columns of the table. What is apparent from this and from Figure 4-33 is that the increase in span time for the combination scenario is approximately 2-1/2 times larger than that of the simple addition of each of the effects alone. For example, when *m*=1.0, reducing the integrator resource capacity to *INTMAX*=4 alone results in a 2% increase in span time; increasing the average latency delay **NLATM** to 0.1 results in a 6% increase in span time; increasing the time period between which the development team deals with communication work **NDT** to 0.1 results in an increase in span time of 2%; and adding additional integrator resources only when *NIQ*=0.1 increases the span time by 4% (*NIQ,* defined in Table 4-1, is the normalized integrator queue time threshold where additional integrator resource capacity is added). The sum of these individual increases in span time is 14%; however, when all of these are combined, the span time increases by 32%. The

combination of all of these delays are such that tasks cannot complete their work in time, and design version rework ensues.

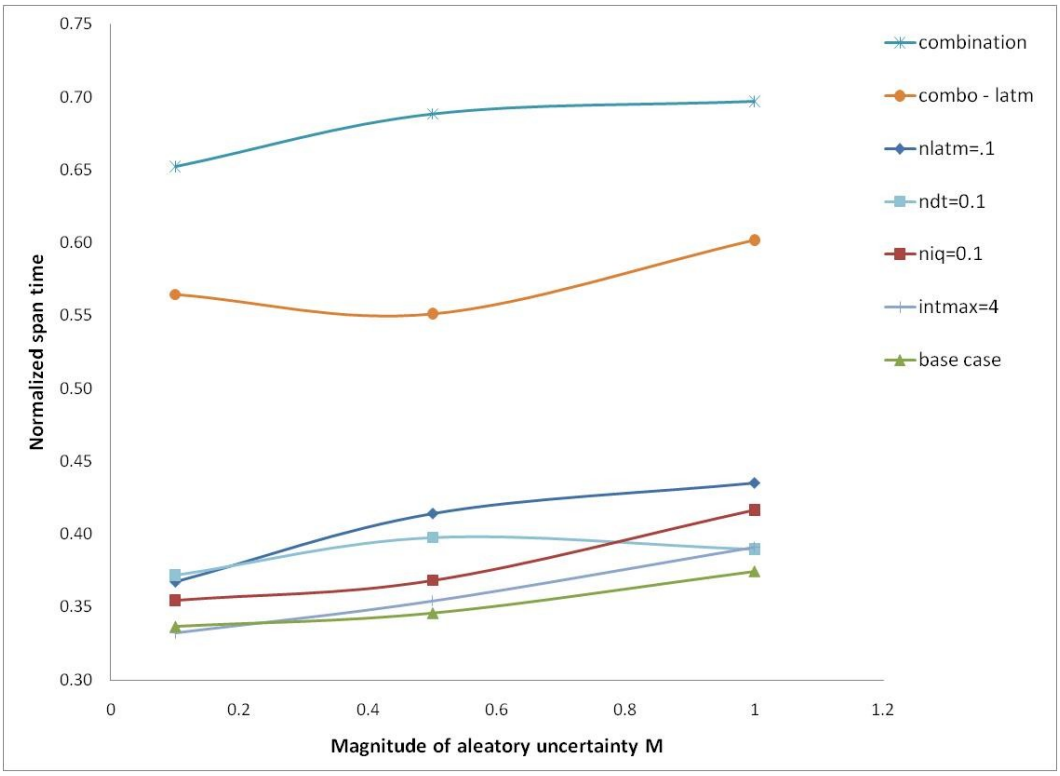


Figure 4-33 Normalized span time versus magnitude of aleatory uncertainty for cases in which each input variable indicated was changed in turn from the base case of Table 4-11 (all cases with slowly reducing epistemic uncertainty b=30, c=6).

Delays in getting information to dependent tasks increase the likelihood of rework due to design iteration. This is evident in Figure 4-34 where it can be seen that each source of delay to information flow increases churn. Here, the churn in the combination scenario is approximately equal to the sum of the churn in each individual case. Each type of delay increases the likelihood that information getting to a dependent task is not continuously reducing in uncertainty. This generates design iteration rework as tasks operate with less than up to date information. This phenomenon has been widely observed in several industries by other researchers (Mihm, Loch et al. 2003; Yassine, Joglekar et al. 2003), but has not been demonstrated to be made up of the combination from individual sources in this way.

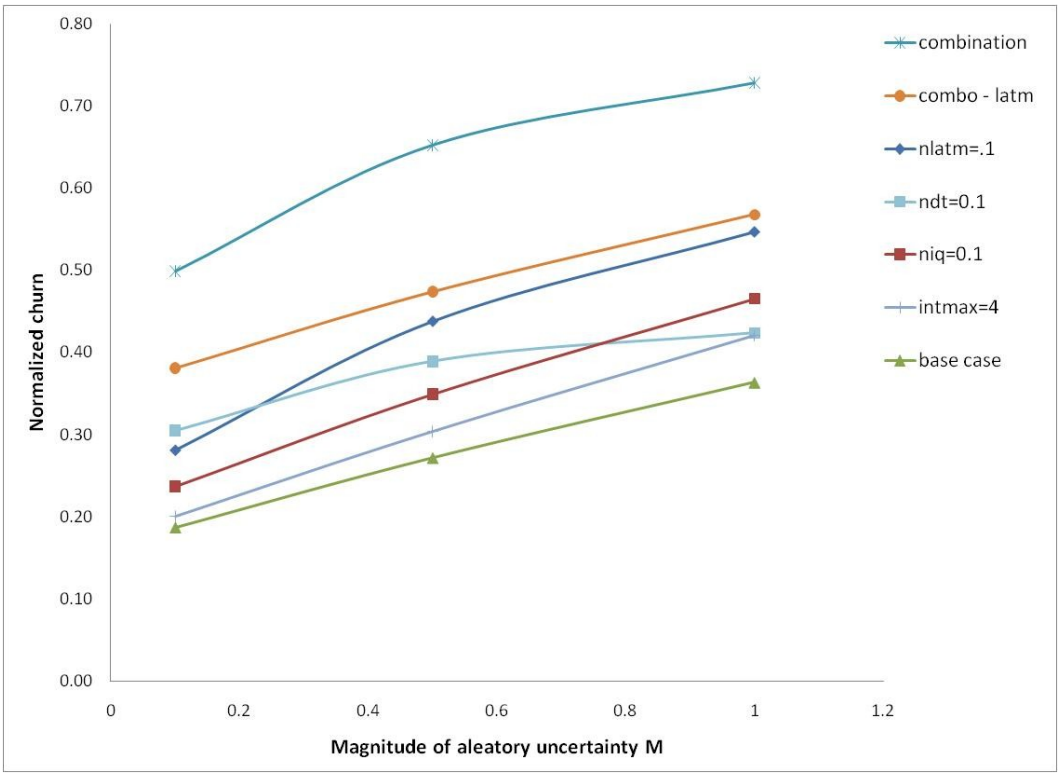


Figure 4-34 Normalized cumulative churn versus magnitude of aleatory uncertainty for cases in which each input variable indicated was changed in turn from the base case of Table 4-11 (all cases are with slowly reducing epistemic uncertainty b=30, c=6).

Thus, sources of delay in information flow combine in a non-linear way to increase span time significantly. This insight has important managerial implications in that reducing delays from various sources to information flow between interdependent tasks can have large leverage in reducing project span time. Making an effort to reduce one or more sources of delay to information flow would have a larger than proportional effect on project performance. For example, Figure 4-33 (curve labeled combo – latm) shows that the reduction in the delay due to latency, **NLATM,** from the combination case by itself reduces span time by 20%.

Similar results for cases with more rapidly reducing epistemic uncertainty are evident in Figure 4-35. Here, the span time increases are lower, but the combination scenario results in increased span time of 17%, whereas the effects from individual delays add up to 5-6% when increased individually.

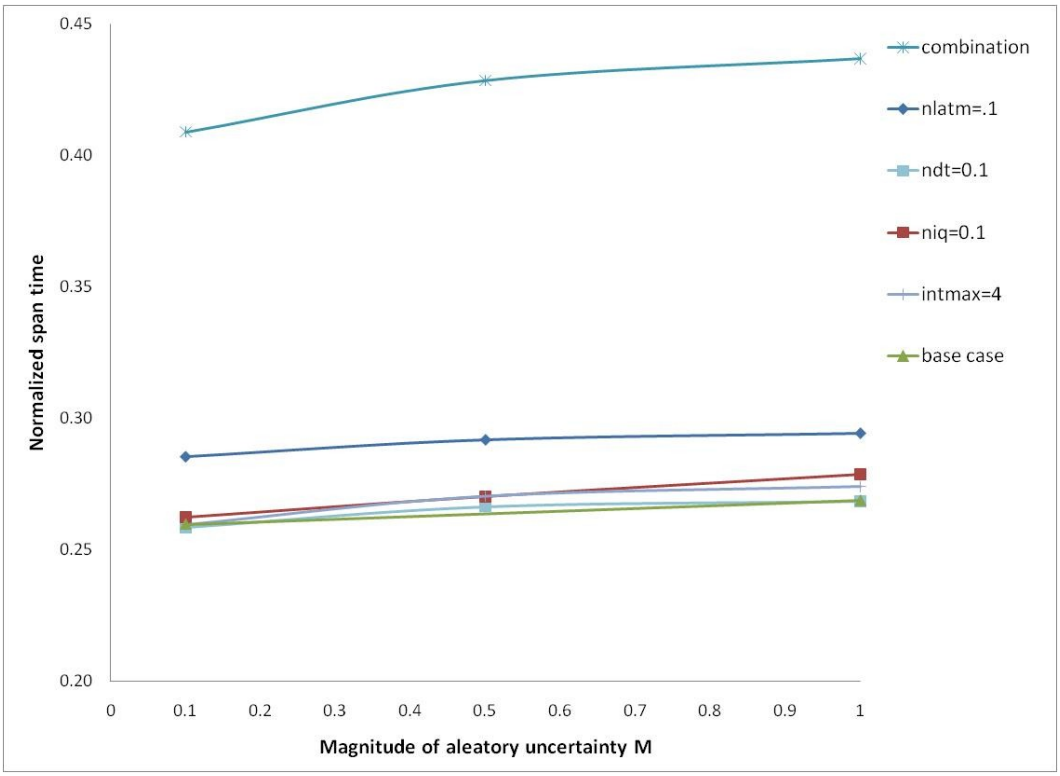


Figure 4-35 Normalized span time versus magnitude of aleatory uncertainty for cases in which each input variable indicated was changed in turn from the base case of Table 4-11 (all cases with rapidly reducing epistemic uncertainty b=5, c=8).

4.6.3 The effects of increasing the number of differentiated tasks in the project decomposition

Simulation results for the normalized span time of a process with 2 phases that has been split into *NPT* development tasks are shown in Figure 4-36. All the scenarios depict processes that require the same amount of average total work, i.e., *TWK* has the same value for each scenario. When splitting the work into *NPT* development tasks there are also *NPT* resources to execute these differentiated tasks. Ideally if there is perfect and effortless coordination, the span time would reduce in inverse proportion to *NPT* since there are more resources available to do the same amount of work. Recall from sections 3.3.3 and 3.3.4 that in our model, the total communication work is considered to be proportional to the total technical work, and so the difference between scenarios in this section is in the proportion of communication that is *between* tasks to that which is *internal* within one development team. Thus, this section examines the scenario where it is

possible to decompose a task into several differentiated, reciprocally dependent subtasks which are nominally equal in required effort, and assign a resource to each one.

The dashed line in Figure 4-36 shows the idealized case where the decrease in span time incurred by decomposing the work into more tasks is a function of the increased number of resources to do the work required, i.e., each normalized span time point is 1/*NPT*. The average total amount of work *TWK,* performed in each scenario in the figure, was 16,000 hours. The base case was run with the input parameters shown in Table 4-11.

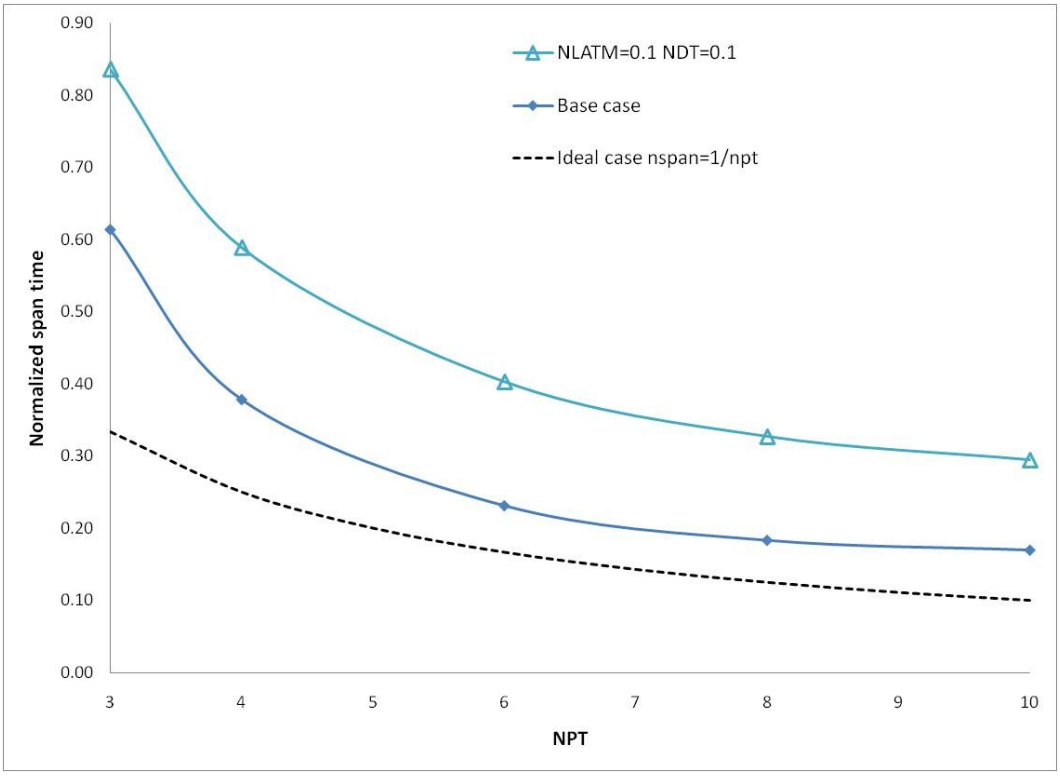


Figure 4-36 Normalized span time for increasing number of tasks in the work decomposition. All cases with slowly reducing epistemic uncertainty (b=30, c=6) and moderate magnitude of aleatory uncertainty (m=0.5).

The span time reduces with increasing number of tasks because of the increased number of resources available to execute the same nominal amount of work, but not as much as in the ideal case. As shown in the figure, there is a significant increase in span time when input parameters governing communication latency

and frequency of attention to communication work by development teams are increased from the base case values. The impact of these delays is not linear with the number of tasks in the decomposition.

In the cases shown, there are sufficient integrator resources at all times, although as shown in Figure 4-37, as the number of tasks increases, integrator effort increases. This is because, as there are more tasks in the process, an increasing amount of the information exchange is done with other tasks, and in our model all of this information must go through the system level integrator. The cumulative effort of the development teams, also shown in Figure 4-37, decreases initially as the number of tasks increases, but starts to grow gradually in the base case after the number of tasks is greater than 8. When there are more delays in information flow caused by **NLATM** and **NDT**=0.1, the effort of the development teams increases much more rapidly with more tasks in the decomposition, starting when *NPT* is 4. The cumulative effort of the development teams at *NPT* =10 is over 30% higher than in the base case and is over 30% higher than its minimum when *NPT*=4. Integrator effort also increases for higher *NPT* reaching 30% greater than the base case when *NPT*=10.

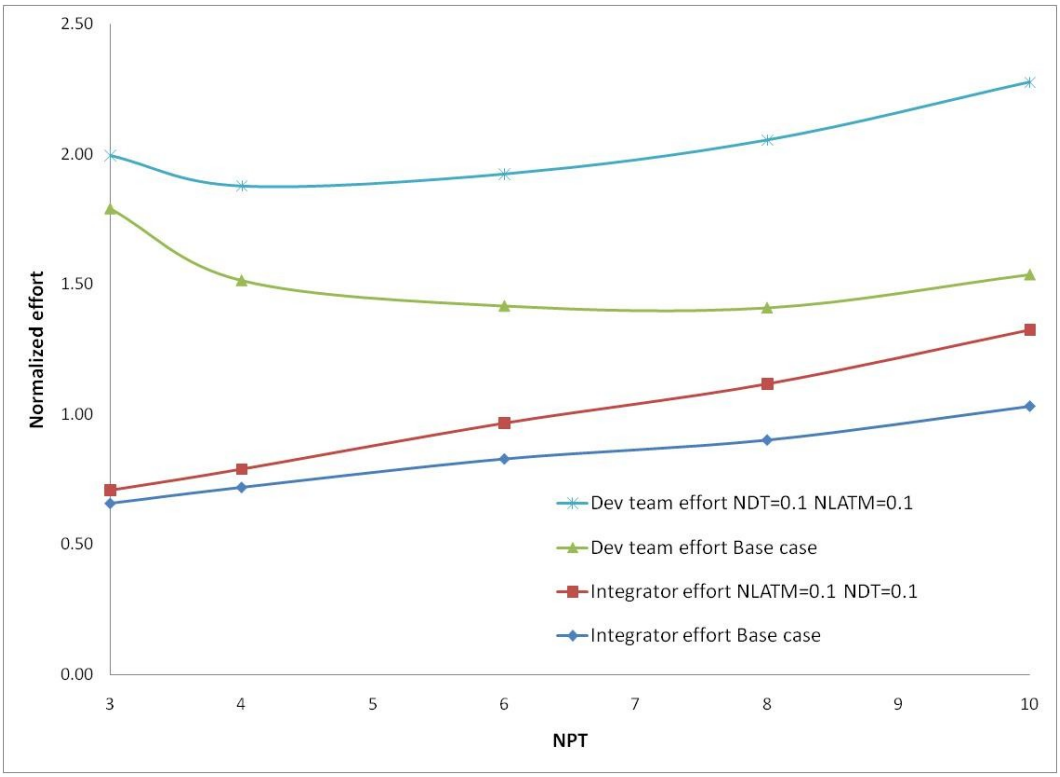


Figure 4-37 Normalized integrator and development team effort versus the number of tasks in the decomposition of the work for b=30, c=6, m=0.5. Base case refers to input parameters in Table 4-11.

Recall (sections 3.3.3 and 3.3.4) that in our model we hypothesized that the amount of information that must be processed in a PD project is a parameter related to the uncertainty in the work for the organization performing it. If there are fewer tasks in the decomposition, more information processing work is done within the development team, i.e., there is a larger proportion of internal communication. We modeled internal communication as being done without delays due to latency (controlled by the input parameter **NLATM**) and without the need for integrator scrutiny. Additionally in the model, the uncertainty of information exchanged internally within the development team does not contribute to the design iteration rework calculation. These model constructs were imposed because in practice the information exchange between members of the same development team is carried out in an informal, almost continuous manner, and with the high bandwidth typical of face to face communication. As such, the precise nature of the information being exchanged within a development team is

known to each member and would not lead to unnecessary rework as a result of misinterpretation of the precision of the information by the participants.

That PD projects with more tasks are prone to longer span times and excessive effort is largely borne out by empirical evidence (Mihm and Loch 2006). Mihm and Loch state that "large engineering projects, in particular, seem to be prone to failure." They also go on to state that according to random matrix theory, a complex PD project featuring many components or many interactions (or both) "inherently has a high probability of problem-solving oscillations, of instability, and of a long design conversion (span) time."

The results in this section point to the reason for longer span times when there are more interdependent tasks. The hierarchical manner in which the system level integration is carried out leads to delays that become more important when there are more tasks. This is not because there is a lack of resources to do this work (although this exacerbates the delays), but is due to the delay imposed by the system level task itself. This delay becomes increasingly onerous when the work is broken up into a greater number of tasks because a greater proportion of the information being exchanged must be processed by the system level task. Additionally, the delays in transmission of information and in getting it to other development teams participating in the project are more significant when there are more tasks in the project since a greater proportion of the required information exchange takes place between development teams performing different tasks.

A delay in getting updated information to each task in a project with high interdependency propagates through the project and effectively is amplified to add up to significant increases in span time. The amplification comes about because of the rework generated when information is delayed. In this sense, this is complementary to the result referenced earlier from random matrix theory, which states that perturbations in larger coupled systems take longer to converge and can actually diverge when there is a large number of elements and continual

perturbations. In the case of random matrix theory applied to PD, the perturbations are changes that take place in some tasks that affect other tasks.

In our model, we can see that the simulation of changes that occur in tasks due to aleatory uncertainty and slow reduction in epistemic uncertainty propagate with delays to other tasks, and this results in more rework cycles as the number of tasks increase. In effect, the delays caused by the integrator process result in dependent tasks being starved for information and being unable to reduce the uncertainty in their work sufficiently to successfully pass the design review. Thus, increasing the number of tasks tends to result in increasing amounts of design version rework. This is evident from the results shown in Figure 4-38 and Figure 4-39 which show that even in the base case of effectively no latency delays, there is a tendency for increased starve time and design version rework as the number of tasks climbs towards NPT=10.

The results with the model indicate that longer span times are not necessarily an *inherent* property of PD projects when there are more tasks in the decomposition. Rather, it is the latency in communication that becomes more important and the manner in which a PD project is managed to ensure that the work of individual development tasks is properly integrated. If these aspects of a PD project can be mitigated through managerial or technological systems to expedite information exchange between tasks and a more distributed way of managing the integration of the work of development teams is adopted, it is feasible that large PD projects can operate as efficiently as smaller ones.

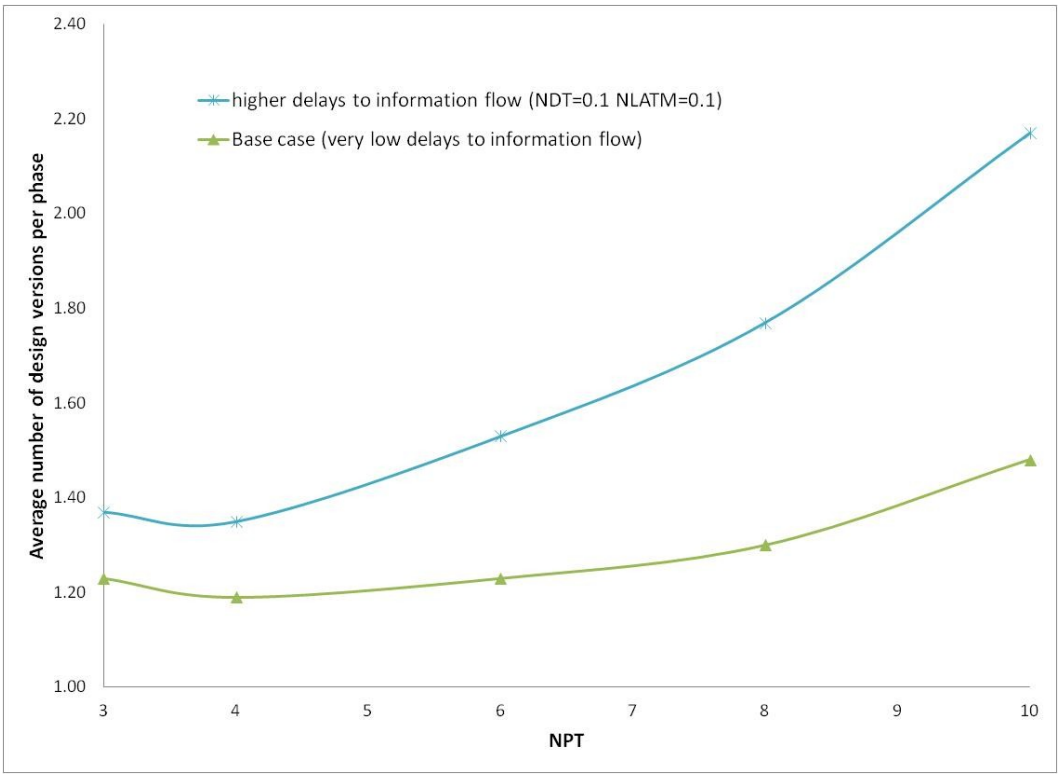


Figure 4-38 Average number of design versions per phase versus the number of tasks in the decomposition of the work for slowly reducing epistemic uncertainty and moderate aleatory uncertainty (b=30 c=6, m=0.5). Base case refers to input parameters in Table 4-11

This insight has implications in planning PD processes. In order to get a new product to market earlier, it would be advantageous to involve as many people as possible in the product development process. Yet, we find that increasing the number of differentiated subtasks in processes is counterproductive to reducing span time. Is there a way to ensure product integrity without using the vertical channels of the integration process as it is modeled here? In section 4.6.4 we examine a PD project structure that may be effective.

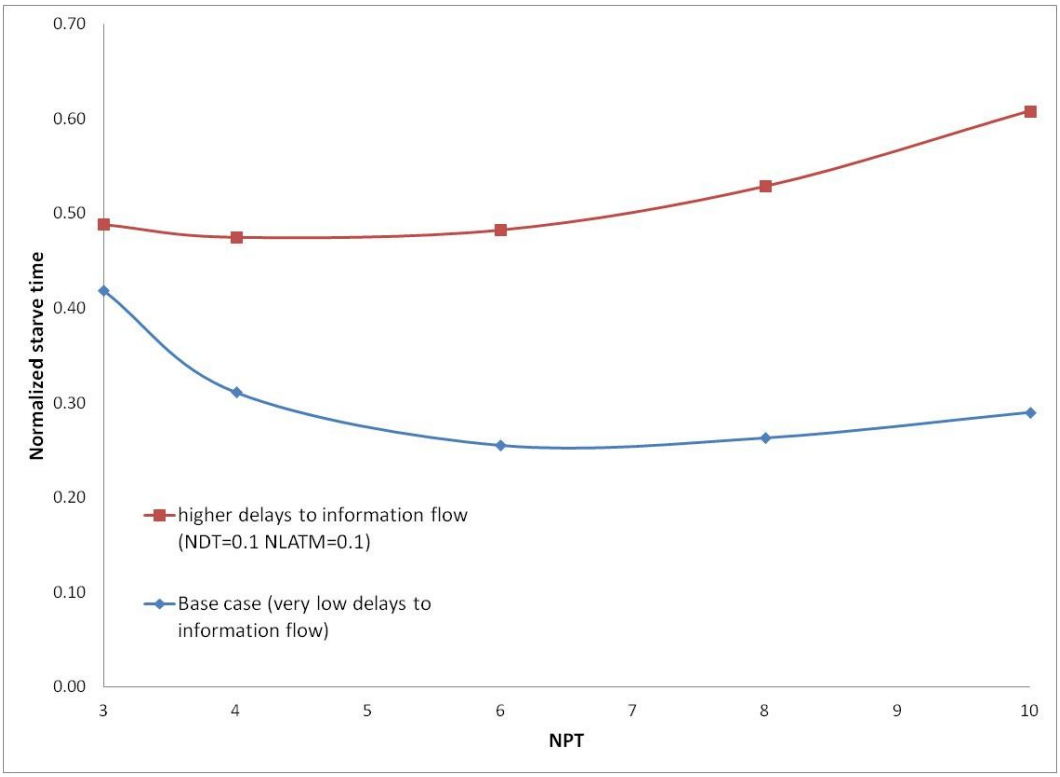


Figure 4-39 Normalized cumulative starve time versus the number of tasks in the decomposition of the work for b=30, c=6, m=0.5. Base case refers to input parameters in Table 4-11

The scenarios considered in the previous figures had ample integrator resources to perform the work required without impeding the project. In the scenarios shown in Figure 4-40 we use the model to predict the performance of these projects when integrator resources are more constrained. Here, when integrator resources are limited to 4 units, span time performance is identical to the less constrained case when the decomposition has less than 7 tasks, i.e., the span time reduces because there are more resources available to perform the same nominal amount of work and the impediments to information exchange are low. However, for projects with more tasks, a sharp increase in span time is predicted as the waiting time of information for the integrator process impedes the flow of information, effectively acting as a bottleneck to the process. In these two scenarios, additional resources are added to the project when the average waiting time in the queue for the integrator resource exceeds 0.2% of the nominal span time. In a third scenario, with sufficient resource capacity available, but with additional resources only

occurring when the average waiting time in the integrator queue reaches 20% of the nominal span time, results show an important influence on the span time even at lower *NPT*.

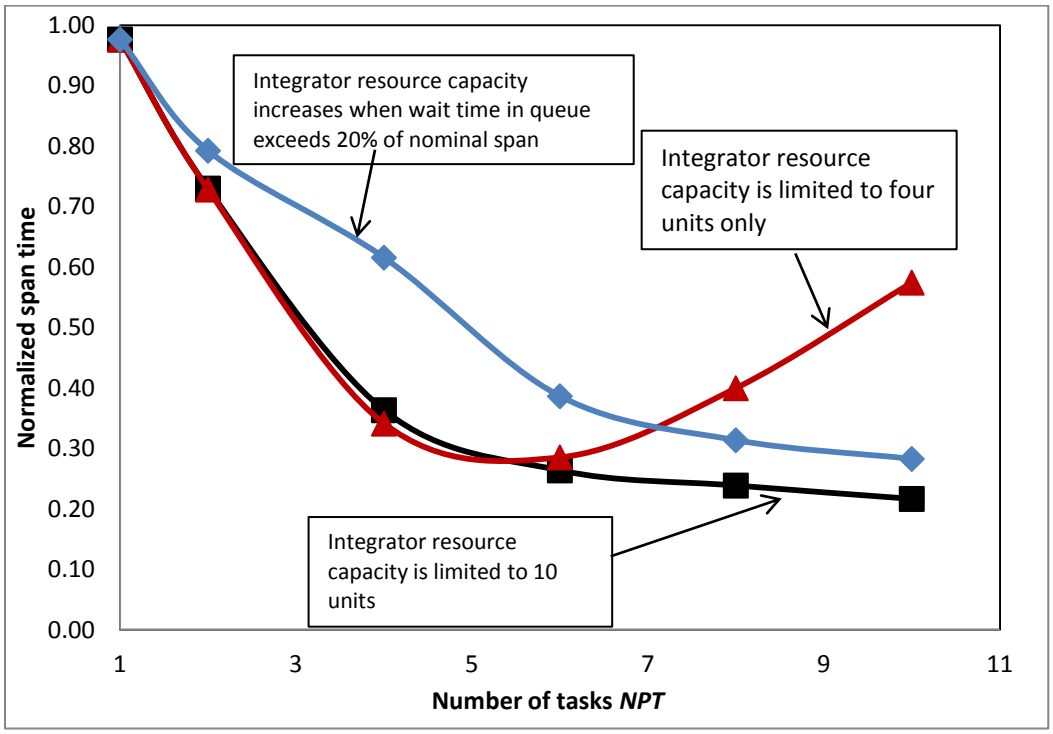


Figure 4-40 Normalized span time versus *NPT* with constrained integrator resources

Results for the effort required to complete the project are shown in Figure 4-41 below. Here, effort increases when the integrator resource capacity restriction causes a bottleneck in the process. Similarly, when the integrator resource capacity is insufficiently sensitive to the queue state, effort is higher even at low *NPT*. The mechanism that causes these aggregate performance effects in the simulation is an increase in the occurrence of design version rework. This is caused by the combination of two effects: one is the high uncertainty of information that still remains when the tasks are finished; and the second is due to incomplete scrutiny by the integrator of the information generated by the tasks. Recall from the description of the model in section 3.3.7 that information waiting longer than a determined by a normal PDF reneges the integrator queue and proceeds directly to the addressee task. This lack of system level scrutiny

increases the chance that a task will fail at the design review at the end of the phase.

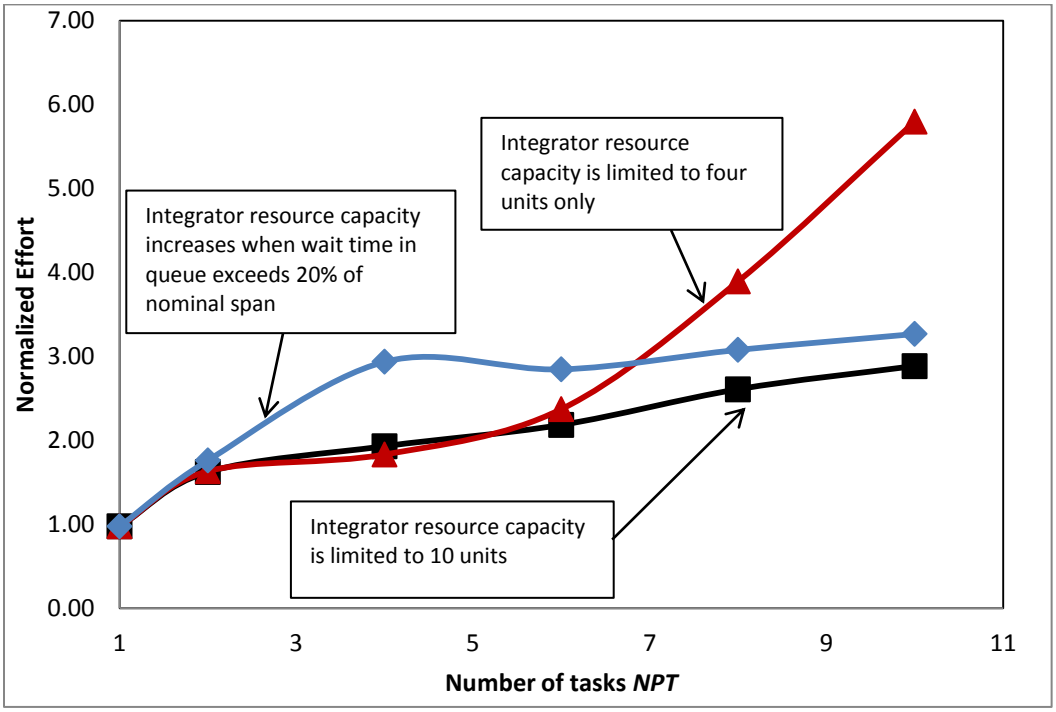


Figure 4-41 Normalized effort versus *NPT* with constrained integrator resources

4.6.4 Adapting Agile PD Methods to Non-Software PD

In this section we model a coordination scheme called 'scrum' that is part of Agile Product Development Methods (Cockburn 2006). In the scrum scheme, the development team is a cross-functional group that does the analysis, design, implementation, testing, etc., that is required to create an incremental deliverable for a product design in a short period of time (typically one month). During this period of time, called a 'sprint', the team is required to produce an entire, tested version of the software that completely answers a planned set of requirements. Each successive sprint goes on to add additional requirements so that at the end of the project the software meets the complete list of requirements for the final product. The sprints are characterized by intense communication within the self-managing development team (typically co-located), and ironclad commitment to achieving the agreed to requirements (which cannot be changed during a sprint)

within the allotted time frame. This product development method has been found to be effective in significantly reducing software development time.

Although this scheme is feasible in software development, it cannot be replicated literally in mechanical design where physical parts must be designed, materials procured, and then undergo a manufacturing process in order to produce a complete version of the product that can be tested. However, in an analogy to the scrum concept, we could consider the goal of a sprint in mechanical design as the solution to a series of well defined design problems, providing the information required to make a key decision in a PD project. Each successive sprint would then provide further information to an additional series of design problems until the final product design is achieved.

A key element here is that the tasks involved in each phase collectively and completely solve a specific set of design problems that can be evaluated during the design review. Since PD of a complex product is essentially a series of activities providing information that allow key design decisions to be made, the design review at the end of each of these short phases or 'sprints' formalizes the point at which these decisions are made. The tasks in each phase leading up to a design review are those that generate the information required to make these key decisions. As in the sprint method for software PD, the work in each phase must be done with intense interaction between all participants so that the design review is successful and rework of a phase with the same design requirements is not necessary. In practice, this is facilitated by smaller sized phases.

Thus, in our model, a simulated project is divided into a series of short phases, analogous to sprints with a design review at the end of each phase. To model the co-location and intense communication between the various tasks, we eliminated the system level integrator function from its role as the scrutinizer of each information item, but rather included the integrator as a participant in the co-located group of development teams. All delays due to latency of information exchange were removed. For comparison, we performed simulations of a project

with the same technical work content with two phases and five tasks with an integrator function acting as oversight of all information communicated between the tasks in the project, and communication latency to get information from one development team to the other. The input parameters are listed in Table 4-14.

Table 4-14 Input parameter values for the comparison of scrum PD with the standard method

Input parameter	Standard PD	Scrum method
TWK	20,000 hours	20,000 hours
TP	2	6
NPT	5	5
$NLATM$	0.05	0.0005
NIQ	0.05	0.0002
$INTMAX$	5	20
NCI	0.15	0.15
NDT	0.02	0.01
ATD	1.5 hours	0.1 hours
$MINT$	15 hours	0.01 hours
PRD	1.5 hours	0.1 hours
b	30	30
c	6	6
B	0	0
C	1	1
$\boldsymbol{\alpha}$	1	1
D_{ij} for all i, j	9	9

Figure 4-42 shows that with slow reduction in epistemic uncertainty span time is significantly lower with the scrum method. When the magnitude of stochastic uncertainty increases from 0 to 0.9 the span time remains approximately constant in the scrum scenarios, but increases by approximately 25% in the standard scenarios. This result, as illustrated in Figure 4-43, is primarily due to lower churn with the scrum method and the mitigation of the effects of increases in stochastic uncertainty on churn. When epistemic uncertainty reduces more rapidly, both the standard and scrum methods are insensitive to increases in magnitude of aleatory uncertainty as can also be seen in Figure 4-42.

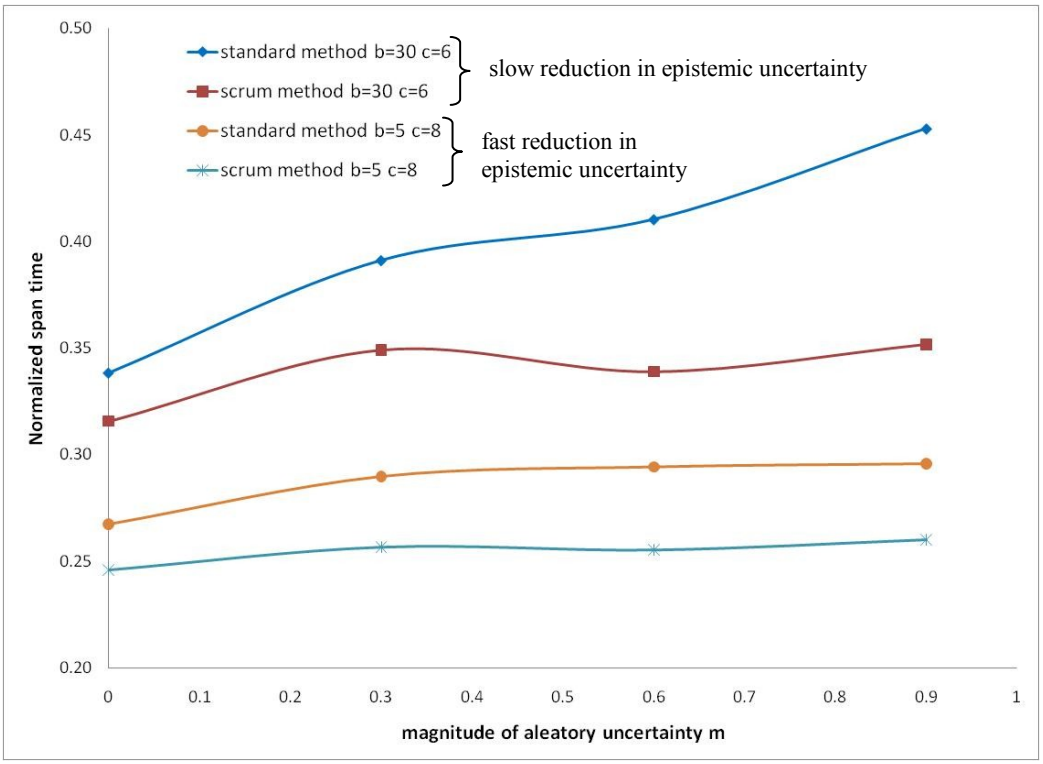


Figure 4-42 Comparison of span time in scrum and standard PD for several cases of uncertainty.

In scrum, short sprints with more design reviews provide opportunities to efficiently judge the completed portion of work accomplished in each sprint and to provide feedback, keeping the work from straying too far off track. In interviews with senior PD project managers[21] we found that frequent design reviews are sometimes used in PD in the aerospace industry at the discretion of the project director rather than as standard operating procedure. In the words of a veteran PD project director we interviewed, "with frequent interim design reviews we were able to keep the work on track … stay 'out of the ditch' and keep the work as close as possible to the 'white line' in the middle of the road." Thus, the scrum method works by allowing for intense coordination among the people doing the work who can most effectively do this coordination, while providing opportunity for managers to keep the work on track and to refine and clarify requirements as the project progresses.

[21] See Chapter 5.

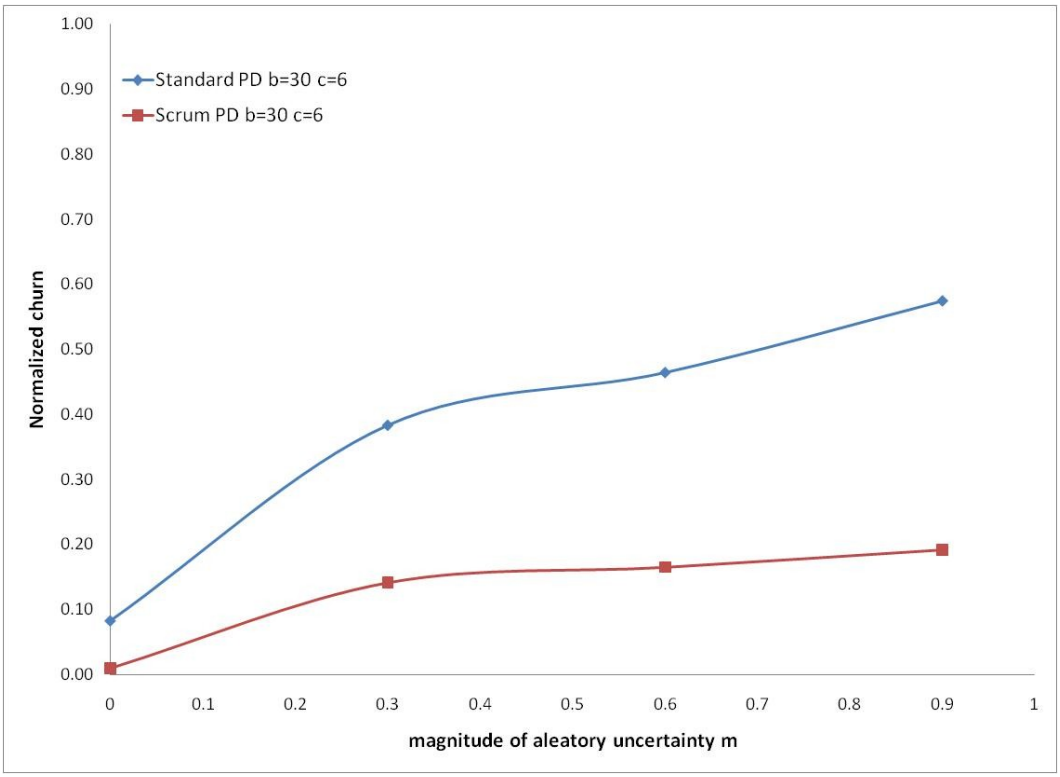


Figure 4-43 Comparison of cumulative churn in scrum and standard PD for slow reduction of epistemic uncertainty.

4.6.5 Effect of non-equal task sizes

Figure 4-44 shows how span time is affected if one of the tasks in the task decomposition of a project requires a greater proportion of effort than all the rest. The figure shows how span time for a project divided into 4 interdependent tasks and 2 phases changes with the effort multiple (*NW*) of the one unequal task. Thus, when *NW* = 2, one of the tasks requires twice the average total effort of each of the other 3 tasks. In each case the average total project effort is identical. This significant increase in span time can be understood when studying the simulation. The longer time taken by the larger task to generate information sufficient to keep the other tasks from reaching a starve condition makes the pace of the project as a whole follow that of the larger task. Even though the total work required is equal for all values of *NW*, the smaller tasks cannot make progress and are often idle waiting for input information. This result agrees with the recent work of

Beauregard, Bhuiyan et al. (2011) who studied the effect of job size and other parameters on engineering performance.

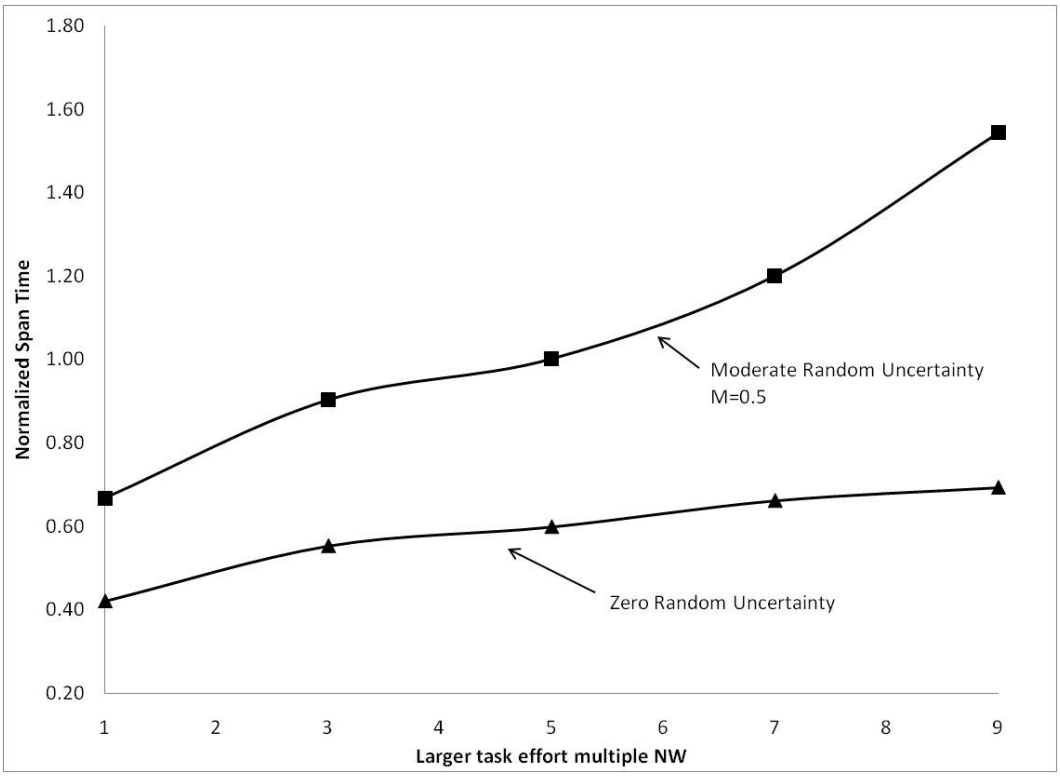


Figure 4-44 Effect on span time when one task is larger than the others

4.6.6 Influence of the relation between total communication work and total technical work in a phase: the parameter α

Recall that in the development of the model we reasoned that the number of communications between participants in product development projects is proportional to the dependency strength between them (refer to section 3.3.3.2). In order to calculate the proportionality constant C_L, we equated the average effort in communication work to **α** times the average effort in technical work for the PD project being studied (equation 3-8). For most of the simulations presented here, the value of **α** was taken to be unity based on discussions with participants in PD projects in industry that we interviewed. In this section we examine the implications on the results presented by varying the value of **α** from unity.

As illustrated in Figure 4-45, the effects of increasing and reducing **α** by 20% are shown in comparison to the base case where **α**=1. These results indicate that the behaviour of the process is identical in cases of slowly and rapidly reducing epistemic uncertainty versus communication interval except for a change in magnitude of the normalized value of span time.

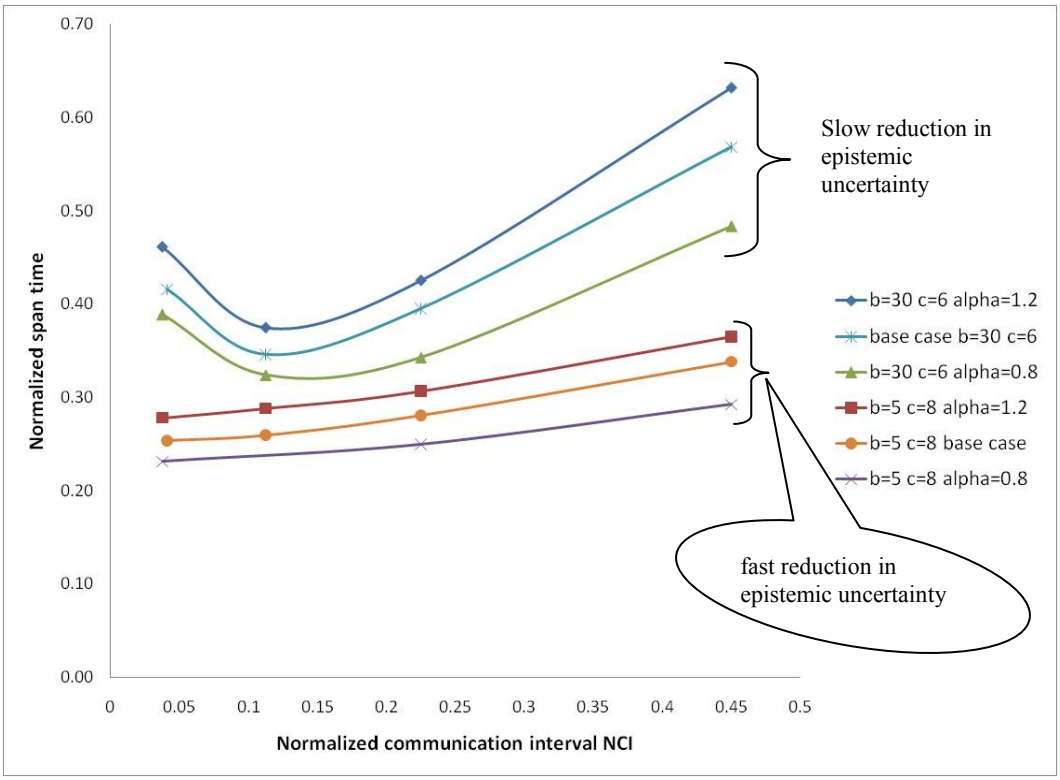


Figure 4-45 Effects of changing the value of parameter **α** on the behaviour of span time versus *NCI.*

Similarly in Figure 4-46, where the effects of varying **α** versus the magnitude of aleatory uncertainty for two profiles of epistemic uncertainty reduction are shown. These results illustrate that changing the value of **α**, while changing the absolute magnitude of the span time, does not change the character of the relationships between project performance and input parameters.

In using the model to study a particular situation, simulations must be calibrated in order to obtain meaningful magnitude predictions. The choice of the parameter **α** does not alter the relative performance characteristics of a particular scenario.

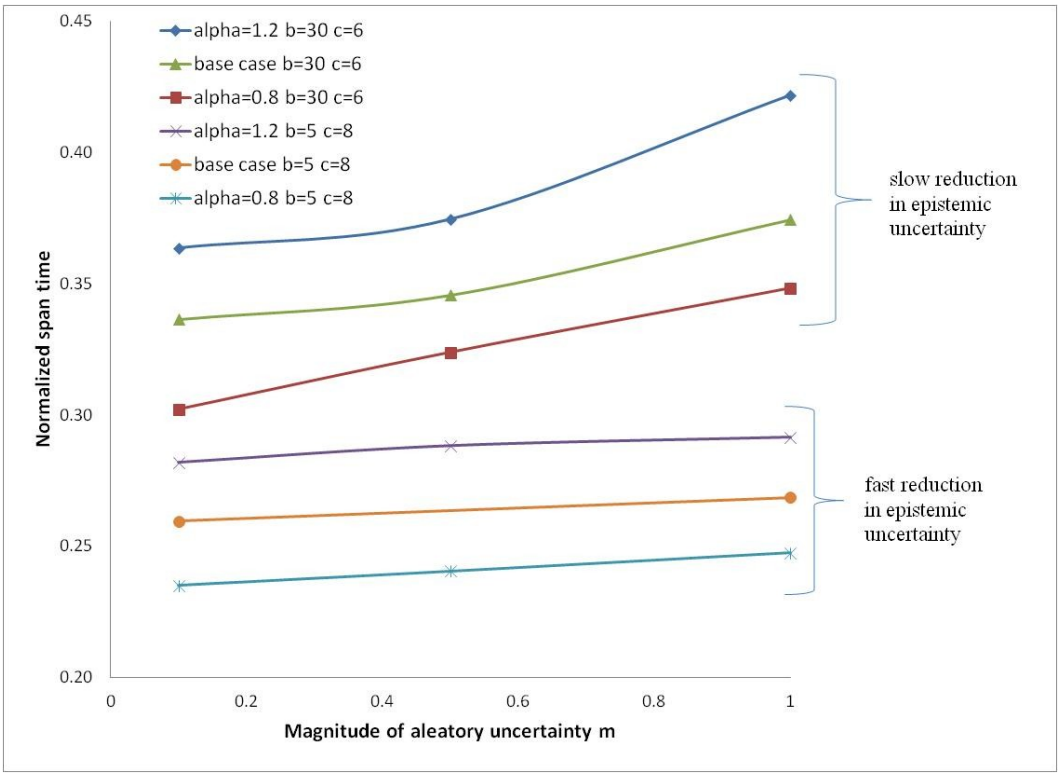


Figure 4-46 Effects of changing the value of parameter **α** on the behaviour of span time versus *m*.

4.6.7 Alternative management of the system level integrator resource

The system level integrator is effectively a shared resource among all development tasks in the process. Information communicated from a development team is evaluated in this process before being sent to the other development teams requiring it. If there are many tasks in the process, the resource capacity available to perform the system level integrator task can be a bottleneck. This was observed in simulations and led to the inclusion in the model of flexible resource capacity for this operation. The schemes to manage this resource, described in section 3.3.8, were modeled to simulate the managerial actions that would be used in practice.

Results are presented in Figure 4-47 for three cases chosen to highlight the effects of choosing one of these schemes. All three cases are with tasks with slowly reducing epistemic uncertainty and input parameters shown in Table 4-11, but with **SCH** = 1.5. The value of *NIQ* for the base case was 0.0002, which means

that effectively when the waiting time in the integrator queue is greater than zero, an additional resource is added. The top curve in the figure had a value of *NIQ* of 0.1, whereas the third case used the alternate scheme to manage the integrator resource, described in section 3.3.8. This alternate scheme, effectively performs an ongoing calculation of the expected time required to do the remaining integrator work and adds additional resources if this exceeds the remaining time available. The value of **SCH**=1.5 specifies that the allowed span time of each task cannot exceed 1.5 times the maximum effort specified to perform that task.

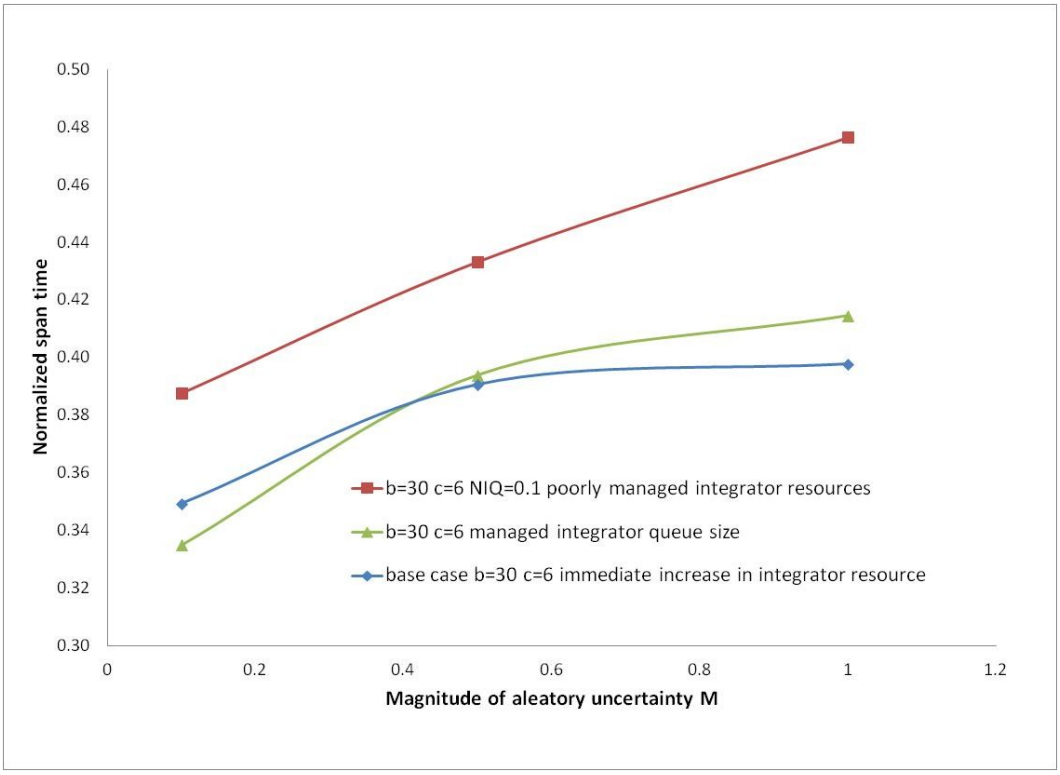


Figure 4-47 The effect of alternative schemes to manage the integrator resource capacity versus *m* for slowly reducing epistemic uncertainty

As seen in Figure 4-47, the base case yields a span time of 10% lower than the case with *NIQ*=0.1. When the alternate scheme is used, span time performance is much improved, being almost equal to that of the base case. Clearly, the timely availability of sufficient integrator capacity reduces the delays that cause rework as we have seen.

5 Current practices in a selection of aerospace companies

In order to construct the model described here, background information on complex PD was required. Several companies (Bombardier Aerospace, Pratt and Whitney Aircraft Canada, Esterline CMC Electronics, CAE, and Rolls Royce Canada) in the aerospace industry agreed to participate in research into process models of PD and provided the opportunity to study current practices and find insights into the key drivers of performance.

The investigation into current engineering design processes and process models in several participating companies was carried out by reviewing process model documentation and interviewing engineering personnel. The research indicated that, although formal prescriptive process models were in use to allow management of the complexity of designing aerospace systems and to satisfy quality, safety, and regulatory concerns, these process models were insufficiently detailed and incomplete for the purposes of process analysis and improvement. Information flow was insufficiently detailed to allow analysis of the dependencies between activities on the specific information output from one activity that is required by others.

Existing process definitions can refer to obsolete practices, can contain detail that is irrelevant to most jobs, can leave out key practices, or can capture activities that were once critically important, but have become irrelevant over time. Partly as a result, the details of these process definitions are followed only loosely. Quality work is nevertheless accomplished thanks to the professionalism of the engineers and managers involved and the conservative review and verification practices, but this situation has a high potential for process inefficiency. This inefficiency has been observed in aerospace companies industry-wide.

The product development process at the typical company is run as a series of phases in a stage-gate process that is formalized in the company's PD process documentation. The process is described in a hierarchical structure of tiers of

increasing granularity. In studying these processes we found that the descriptions of the tasks, while increasing in granularity in each tier, were not decomposed hierarchically into more detailed tasks in the lowest tier. There were different degrees of detail that were only loosely connected so that there was no straightforward way to integrate the existing process maps in the lowest tiers with those above it. As a result the usefulness of the process description is limited to a high level process description that allows tracking of the various formal deliverables. This, although being important in maintaining the integrity and quality of the product development process and for the monitoring of deliverables, can only serve as a useful starting point for process analysis and improvement.

5.1 Interviews

From interviews we carried out with functional managers, we found that many participants in product development projects belong to engineering departments that represent technical disciplines, such as flight sciences, aerodynamics, structures and loads, propulsion systems, etc. while also assigned to a development team in an aircraft program. The organization of development engineers is a matrix with engineers assigned to development tasks for a new aircraft program. This practice is optimal in terms of preserving and maintaining the discipline knowledge of personnel involved in development programs while promoting the collaboration and coordination needed for interdisciplinary product development teams. It has been found through empirical research that the departmental structure, although assuring that technical staff can keep in close contact with new developments within their specialties, is weak on combining and integrating knowledge from different specialties to develop a new product (Allen and Henn 2007). This requires coordination between specialists, each of whose work or approach can have a serious effect on the work of other specialists. Information must be exchanged among the specialists in order for them to understand what everyone is doing, often on a regular and quite frequent basis.

During interviews with participants in several areas of the product development process, we were able to gain a more detailed understanding of the process. We found out how the aircraft design begins with the requirements of the mission (payload, number of passengers, range, cruise speed and altitude, operating costs). These are translated into more specific requirements such as cabin layouts, fuselage length and diameter, wing area, engine specifications, and tail configuration by an advanced design group. Using this data, a 3D model of the exterior of the aircraft is created by a design team, often with only simple shapes representing the wing and other major elements. A parametric model of the aircraft is made such that as the wing sections and wing twist are refined after aerodynamic analysis, the overall area of the wing is still preserved, and fairings are regenerated to join the fuselage and wing. This process of wing design proceeds with a successive series of wing sections and twist distribution, computation fluid dynamics analysis of the aerodynamic loads, calculation of dynamics of the aircraft and its stability, design of the tail plane, and calculation of stress distributions and structural design, until the exterior of the aircraft is refined to an optimum configuration.

This process is carried out by a relatively small group of engineers from the engineering departments in consultation with customer representatives to ensure that the requirements for the characteristics of the aircraft are met both from the point of view of performance and accommodation of payload and passengers. At this point wind tunnel testing of the configuration may be undertaken to validate the performance of the aircraft design.

The geometric constraints, performance and mass of systems that must be incorporated into the aircraft are only estimated at this stage and are taken into consideration in the design of the aircraft exterior. These are based on the knowledge and experience of designers and information available from previous designs. Once sufficient confidence in the configuration of the new aircraft design is established, potential partners or external companies that are considered for the development of major aircraft systems are invited to participate in the further

development of the design. Further aspects of the design and costs of engines, landing gear, wing structure, and mechanical systems for flight controls such as actuators for flaps, slats, etc. are clarified during this collaboration. These pre-launch consultations serve to clarify features of these systems, allow for the consideration of newer technologies, and diminish uncertainty with regard to earlier estimates of performance, mass, geometrical and other constraints. Once these consultations are complete, the external shape of the aircraft is mainly frozen for the duration of the product development process with the intention that only minor changes localized to fairings would take place downstream in the process.

Several milestones are indicated in the process documentation representing reviews and decisions made as to the feasibility of the new aircraft to be developed from the point of view of technology, cost, schedule and markets. If these decisions are made in the positive, the aircraft development program is formally launched and the company makes commitments to its potential clients, the market, and its shareholders as to the performance characteristics, costs, and delivery date of the aircraft for service. It is at this point that the uncertainty of aircraft attributes must be sufficiently low that the decision can be made with confidence based on the work that has been performed.

Once the decision to launch has been made the company prepares itself to develop the preliminary definition of the aircraft by planning the balance of the development process in further detail. This starts with assigning key individuals to lead the project and is based on a breakdown of the project into work packages and detailed schedules, the calculation of manpower requirements and milestone dates of key deliverables and reviews that must be met so that the project can deliver on time. After a further review of these preparations the project team begins the preliminary definition of the aircraft starting from the exterior contour surfaces that have been established. It is during this phase that potential partners of key systems co-locate their personnel with members of the project team at the lead company's engineering facility.

The governance of the new aircraft development program is composed of a program manager that oversees the engineering design process, manufacturing, sourcing, and support functions required to bring the new aircraft to service. The governance is done by monitoring the delivery dates of deliverables of the work packages which for the work breakdown structure of the project and identifying risks along the way. Identifying risks is the means with which foresight can be used to mitigate potential risks before they become larger issues. However, to control the process only formal deliverables are reviewed, where deliverables are managed sequentially. This means that the completion of a scheduled deliverable is seen as the beginning of the subsequent activity which requires it as input.

Given the complexity of the design process, in this thesis, the focus was on the aspects of engineering design and the ways to coordinate the process better. Often deliverables are comprised of many pieces of information that are required by several different activities or sub-activities. Thus, there are many opportunities to sequence sub-activities to begin earlier by identifying the information content of deliverables in a more detailed manner. Moreover, different levels of precision of the information may be required by different sub-activities, allowing an earlier start for sub-activities requiring the same pieces of information. This level of analysis of the information content and flow highlights opportunities to synchronize the work of activities and provides guidance and feedback with intermediate design reviews.

At the moment, from our interviews with program managers and project directors, it seems that the process planning and governance structure at these companies does not allow for this kind of initiative in process management. The people performing the activities are doing their work according to the defined deliverable schedules. The people managing the project monitor and manage the project according to the granularity of the details in the schedule. In our interviews with product development project directors, it was conveyed to us that the initiative to go beyond the coordination practice set out in the project plan is only available to the highest level of project leadership. Here, at the discretion of the project

director, more frequent reviews of work progress or changes to the sequencing of individual activities can be carried out if required. The diagnosis and choices for action are initiated based on the experience and knowledge of the project director, but best practices and the criteria for making these decisions are not systematized in the company and are very much based on the intuition of key personnel.

In order to make further process improvements we believe that it is necessary to capture further details of the activities and information content of deliverables. However, the effort to do this is significant and does not need to be done at all parts of the process. Rather, effort should be focused on critical areas of the process, where there is higher uncertainty, more interdependence, and greatest complexity. This can begin with an analysis of the information content of the deliverables from the current documentation and continue with an analysis of the information requirements of activities and sub-activities in work packages. Where design activities are performed externally, the analysis can focus on the information content of the deliverables required to launch the external design work.

We were unable to obtain relevant data of a specific instance of a PD project since the data required was simply not collected in the necessary detail. However, from interviews we conducted with project directors, project managers, integrators, engineering department heads, and individual engineers, we were able to validate many of the assumptions made in the construction of the model, such as the following.

1. The overall structure of the process as described in Figures 3-3, 3-4, 3-5, and 3-6 is typical of PD processes in these companies.
2. Within the engineering departments the work associated with communication of information is roughly equal to the technical work in a PD project.
3. The technical work fraction and the input information fraction are qualitatively equal in importance in judging the status of a task's progress towards completion.
4. The precision of information developed by some tasks does indeed evolve more slowly towards final results whereas the precision of other tasks

evolves more quickly; this is very much determined by the nature of the task, and not necessarily by the level of initial uncertainty of the task.

5. Work on tasks does stop due to the lack of timely input information, although many engineering managers wait until receiving all of the needed information before starting a task. This is due to the perceived likelihood of having to stop a task and the wasted time that this creates. Often planned releases of interim information trigger a formal planned iteration that allows reciprocally dependent tasks to make progress towards a solution.
6. When new information contradicts earlier information, rework is done, whereas, if new information is within the range of previous inputs, progress can be made towards refinement.
7. Unexpected changes in requirements or other input information (performance or physical specifications) lead to rework if the changes are outside the range of the previous inputs.
8. Design reviews mandate rework if completed work has not sufficiently reduced uncertainty.

6 Discussion

The explicit modeling of information flow taking place in a PD process enables the model developed in this thesis (hereinafter referred to as the discrete event product development simulation model or DPDS for brevity) to capture the dynamic complexity of projects with interdependent tasks. This is accomplished through the explicit and detailed modeling of information exchange, the linkage of information exchange to the work accomplished in each task, the deployment of resources, and the techniques used to manage the PD process.

The model differentiates between unnecessary rework and the iterative refinement of tasks that occurs as information is communicated from task to dependent task. This iterative refinement is incorporated in the information exchange between each pair of tasks where each group of interdependent tasks must exchange information to carry out the refinement of their mutual work and information is increasingly updated as the process is carried out. It is the way in which information exchange takes place in combination with changing uncertainty that has an important effect on the overall progress of the process. In simulating information exchange as it encounters impediments from its preparation for communication to its utilization by development teams performing dependent tasks, the model is able to tie together aspects of the structure and management of the process to its overall span time and effort.

The behaviour predicted by DPDS about span time and effort is supported by the behaviour predicted in other models, but the work here goes further to determine the underlying mechanisms for the observed performance and expands the range of configurations and conditions that are studied. For example, Krishnan and Eppinger (1997) introduce the notion of evolution of upstream information to describe the continual improvement in the precision of information generated by an upstream task. The authors work through the implications of slow and fast evolution of information concerning the consequences of task overlap in

situations of sequentially dependent tasks with different sensitivities to change. Similar conclusions are supported by the quantitative results from using the DPDS, where the additional effects of coordination mechanisms and of aleatory uncertainty influencing process behaviour are simulated.

In a field study of concurrent engineering practices, Swink et al. (1996) found that companies tended not to overlap manufacturing process planning tasks with design tasks for the development of products when there was a high degree of innovation or new technology in the design. A high degree of innovation or new technology implies a high magnitude of aleatory uncertainty, and therefore, this observation of what works in industry practice supports the findings here that overlapping downstream tasks is counterproductive when there is high aleatory uncertainty (Figure 4-13 Figure 4-15).

In a model by Bhuiyan, Gerwin et al. (2004), the authors studied the effects of overlap and functional interaction on the span time and effort of PD with sequential dependency. Using input probability distribution functions based on empirical observations, the authors related churn and design version rework to functional interaction and the rate of upstream evolution. Their results, for the configurations studied, indicated that it is not recommended to overlap tasks when there is slow reduction in upstream uncertainty. However, their study did not have the flexibility of examining the effects of better coordination. Recall from Chapter 3, that in DPDS relationships between aggregate process characteristics are not assumed, but rather are a *result* of the simulations. DPDS predicts that even for slow reduction in epistemic uncertainty, a reduction in span time **can** be achieved with overlapping **if** there is sufficiently frequent communication of interim information and if this information flow is not weighed down by delays due to latency or insufficient resource capacity. Furthermore, as indicated in section 4.5, with sufficiently low magnitudes of aleatory uncertainty, reductions in span time with some overlapping are predicted to be obtainable with relatively little increase in effort. The degree to which rework occurs in DPDS, for given profiles of reduction in epistemic uncertainty, depends upon the coordination mechanisms

that are in use and their effectiveness in facilitating information flow. Here, we see that the amount of overlapping can be usefully increased to further reduce span time without large increases in effort if better coordination reduces latency and other impediments to information flow. Furthermore, the effects of aleatory uncertainty can be studied with DPDS, showing that for a given profile of epistemic uncertainty reduction, larger magnitudes of aleatory uncertainty reduce the effective extent of task overlap.

Using a dynamic model of work transformation in PD based on a process structure of development teams and a system level integration task, Yassine, Joglekar et al. (2003) analyzed the effects of delays in communication of information. Their analysis treated the PD system as a linear periodic system, and they were able to determine under what conditions this system exhibited instabilities. The instabilities for this model represented cases in PD where the number of open design problems to be solved did not diminish, but actually increased due to the generation of rework. The authors state that "delays in information flows… have a destabilizing effect on system behaviour," and that the main sources of churn are interdependency, mismanaged concurrency where flows of information are too frequent and generate more rework than the development teams can handle, and delays in information flow. These findings, developed with the use of a dynamics of systems approach, support those generated with DPDS. Once again, however, DPDS can examine in more detail each of these factors and the way in which they affect the magnitude of churn.

In a model developed by Mihm, Loch et al. (2003) where updates of design information are made to occur at exponentially distributed time intervals, the authors state that "design oscillations are …dampened if component engineers communicate not only final solutions, but also intermediate results…, at least for the key dependencies." This result tends to support the work in this thesis, but in the mathematical model by Mihm, Loch et al., the authors did not consider the aspects of uncertainty that would increase rework when communication of interim information is **too** frequent. In a later paper, Loch and Terwiesch (2005) consider

a model of preliminary information and state that participants in PD should not "treat preliminary information as final, but communicate it at an appropriate level of precision and stability." Their descriptive model of the exchange of preliminary information in that paper points to the duality of preliminary information, that it could be considered stable but imprecise, or precise but unstable. Imprecise preliminary information may be useful for dependent design tasks to carry on with their work and thus allow for more efficient overlapping of tasks. This aspect of uncertainty is incorporated in the model developed here to enable the calculation of unnecessary design rework.

In analyzing PD projects with DPDS, it was found that an appropriate interval of communication of interim information optimizes task progress and minimizes project span time and effort. This interval must be sufficiently small so that dependent tasks are not starved for information and remain idle for too long, but not so small as to cause excessive rework during iteration. Using simulation it was found that the optimal or critical point of communication interval is quite defined when the uncertainty of tasks reduces slowly and there is significant magnitude of stochastic uncertainty. If uncertainty reduces quickly or there is little stochastic uncertainty, the more frequent the communication the better. The mechanisms that cause an increase in span time and effort at either side of the critical point are design iteration rework (churn) due to the too frequent communication of uncertain information, and design version rework where there is insufficient frequency of communication to avoid a deadlocked condition in interdependent tasks.

Analysis of the process structure in a given PD project to determine the relative magnitudes of the dependencies between tasks, as described in sections 2.8.1 and 3.3.3, can point out to PD project planners where information exchange is important. Tasks which are highly dependent benefit from the receipt of interim information, and tasks upon which other tasks are highly dependent should provide interim information frequently. The interval between communications from each of these tasks can be determined by estimating the rate of reduction of

epistemic uncertainty. DPDS results show that those tasks with slow reduction of epistemic uncertainty should not communicate interim information more often than each time an additional ten percent of work progress is achieved.

In examining scenarios where information flow is restricted between interdependent tasks, it was found that the effects due to delays and resource bottlenecks combine in a non-linear way and lead to tipping points. Each delay has a small effect in itself, but also serves to exacerbate the effects caused by other delays in the process leading to increasing levels of churn and design version rework. Conversely, the reduction of some delays and bottlenecks with the implementation of managerial and technological solutions can have a highly leveraged effect on reducing span time. These solutions to reducing delays and bottlenecks can take the form of sufficient scrutiny by managers to address the source of the delay or bottleneck by adding additional resources or expediting information transfers. For example, if it is expected that system level integrators have a backlog of work in evaluating information received from development teams at a particular milestone, managers should ensure that there are additional people available and prepared to participate in addressing this volume of work, to ensure the release of information to dependent development teams on a timely basis. An example of a technological solution to reducing delays in information flow would be the implementation of an 'intelligent' product data management system that notifies only those people whose work is affected that design changes have been made to parts of a product in addition to highlighting them in the system. An 'intelligent' product data management system was not simulated as a separate scenario since its ability to reduce span time and rework is based on reducing delays in information flow, which was simulated extensively.

It was also found that, when employing a greater number of differentiated subtasks, project span times are not reduced in proportion to the number of additional resources employed. This is due to the non-linear combination of delays imposed on information that must travel between the different tasks and the need, in complex projects, for a system level integration task that scrutinizes

the work of development teams and ensures that their work meets the requirements of the product. When information flow is restricted between interdependent tasks, the consequences for projects with a larger number of tasks are more severe because a larger proportion of the required information exchange must take place between different development teams. The degree to which an ideal span time is approached can be improved if interim information is exchanged at an optimal frequency as described above, and if there are minimal impediments to delay information flow. Recently, more and more companies are expanding the number of partners, especially global partners, to reduce cost risk in product development projects. This increases the number of tasks, restricts information flow, and increases project oversight, thus increasing the risk of greater span times. Changing coordination mechanisms to take the change of information flow into account is very important.

When there is a high degree of reciprocal dependency between tasks in a PD process, we found that span time can be reduced significantly by adopting the 'scrum' method as described in section 4.6.4. Here, the work that needs to be performed must be structured to allow for 'sprints' with intensive coordination between self-managing teams executing tasks that together solve well-defined design problems. After each sprint, an interim design review is performed to evaluate the intermediate outcomes, provide feedback, and set the detailed requirements for the next sprint. In this way, the impediments to information exchange are minimized and the need for system level oversight is effectively met. It is these interim design reviews that ensure that the work remains on track. In tasks with higher magnitudes of stochastic uncertainty, this methodology has the greatest impact in reducing span times. The sprint technique has good potential for productivity improvement in design disciplines other than software.

Therefore, if a PD project can be divided into a greater number of differentiated tasks such that additional resources can be brought to bear, and the scrum method can be used to improve coordination between tasks with highly reciprocal dependencies, PD projects can be efficiently completed in significantly shorter

span times. Conversely, reducing span times of PD projects when there are strong interdependencies between tasks cannot be accomplished by simply adding more resources unless coordination schemes are implemented to reduce the delays in information exchange between dependent tasks. Failure to do this leads to greater amounts of rework and waste of effort which are counterproductive to span time reductions.

Simulations of PD processes with DPDS allowed detailed examination of the effects of process structure, critical resource management, communication policies, and uncertainty on rework, project span time and effort. Results showed that significant potential for reducing project span time and effort can be achieved by applying the following methods with a knowledge of the nature of task interdependencies:

a. Overlap sequentially dependent tasks when there is sufficiently frequent, interim information exchange to the extent warranted by the rates of reduction of epistemic and aleatory uncertainty in tasks. Useful work can be accomplished in downstream tasks even without receiving final information from upstream tasks if the range of precision of interim information received is well understood.

b. Use set-based coordination when there are a small number of possible solutions to upstream tasks. This eliminates the dependency of downstream tasks using the assumed information and allows these tasks to be performed concurrently without the need for further information exchanges with upstream tasks.

c. Structure projects such that the size of interdependent tasks is similar. All tasks wait for information generated by tasks that take a longer time, and thus, are held to the rate of progress of the slowest task.

d. Provide sufficient resource capacity early enough in critical support tasks by anticipating workload requirements. Insufficient resource capacity in these tasks impedes information flow and causes other tasks to be starved

or to receive unsynchronized information and thus perform unnecessary rework.

e. Implement 'scrum' methods where high uncertainty occurs to enable intense coordination and keep PD projects on track. This reduces latency delays to information exchange most effectively, forces synchronization of information more frequently, and allows project leaders to review the work done in each sprint and provide feedback.

f. Adopt policies and implement systems to reduce delays in information flow between dependent tasks. Simulation shows that delays in information flow combine in non-linear ways to reach tipping points of greatly increased rework cycles, leading to greatly increased effort and span times.

g. Adopt policies and implement systems to reduce effort in communication of information between dependent tasks. Reducing effort in communication reduces the burden on tasks that must exchange information to refine their solutions. The reduction in effort adds up over a large number of communications and dependent tasks. Care must be taken to only target the tasks that are in need of the information; otherwise, information overload ensues.

Overall, the simulation can be used to help in planning and management of actual PD projects by providing guidelines for improving information flow. Importantly, the analysis of PD projects using DPDS can provide a deeper understanding of the mechanisms that drive project performance. Analysis clarifies the functioning of complex process mechanisms and overall project performance.

A major contribution of the thesis is the explicit modeling of information exchange in PD and the linkage of the information exchange to tasks, resources and management techniques which allows the model to mimic the dynamic complexity of projects with interdependent tasks. It is the combination of many influences with differing profiles of uncertainty reduction, resource capacity, and

task effort that determines the overall performance of a project. No one influence dominates, but rather the mitigation of complex interactions determines the best mechanisms to improve a process.

7 Conclusion

The typology of coordination introduced in Chapter 2 classified coordination mechanisms according to the uncertainty in tasks being performed. These ranged from hierarchical management and standard operating procedures for low uncertainty to the use of lateral relations, liaisons, and cross-functional teams for high levels of uncertainty. In this thesis we have been concerned with PD processes where uncertainty is high. Essentially, the uncertainty or newness of the PD work makes it unlikely that coordination can be done only via a hierarchy of managers making decisions to deal with the frequent number of exceptions to standard operating procedures. For these types of projects, it is clear that more decisions involving exceptions must be made by suitable personnel that are closer to the creation of information allowing them to make more rapid decisions. Yet, the problem of ensuring that these decisions are being made in accordance with overall goals remains.

In PD of complex products we find that lateral relations and liaison roles in the form of integrators and managers are used to deal with coordination issues between tasks developing different subsystems and between the various disciplines, and to represent the requirements of the end user. This integrator role is especially important when more tasks are being performed by participants from other organizations to whom some of the development work has been outsourced. However, we found that when there are many tasks that are reciprocally dependent within one block or cluster of tasks, the span time is longer due to the latency of communication and the integrator process itself.

Rapidly resolving coordination problems reduces the amount of design iteration and design version rework. Engaging in ‘sprints’ of tightly scheduled, intense coordination within self-managing teams executing a series of interdependent tasks that provide well-defined intermediate outcomes can achieve this rapid resolution of coordination issues. Convening design reviews at the end of each

‘sprint’ to evaluate task results, to take decisions, to clarify requirements for the next sprint, and to give feedback, provides the degree of guidance required to ensure that the work in each sprint remains on track with respect to the overall project goals. This coordination scheme, analogous to agile methodologies that have been applied to software PD, is beneficial for all design work and is a key finding of the work in this dissertation.

The objectives of the thesis stated in chapter 1

1. to identify or define a typology of coordination mechanisms that can be applied to PD;
2. to develop a model of PD that can be used to capture the impact of various coordination mechanisms for reduction of span time under all potential scenarios;
3. to use the model to determine the general principles governing the reduction of span time and the level of effort for various types of coordination mechanisms under various types of PD programs;

were achieved and described in chapters 2, 3 and 4 respectively.

In order to meet the objectives, the details of the information flow in a PD project that are affected by various coordination mechanisms were modeled. The model considered the fundamental dependencies between tasks, the iterative nature of engineering design, and the variation of uncertainty of information, and captured the impact of the coordination mechanisms on PD process cycle time. These coordination mechanisms, outlined in Table 2-2, deal with shared resources, various consumer/producer dependencies (prerequisite, transfer, usability), simultaneity, and task/subtask. Efforts to coordinate the PD process can be judged according to how a coordination mechanism facilitates information flow between tasks. To this end we modeled the details of the information flow between individual tasks in a novel way, and captured how coordination mechanisms affect this flow in order to meet the modeling objective of this thesis.

With respect to the classification of dependencies and coordination mechanisms shown in Table 2-2, we incorporated the resource dependency in the system level integrator task with two management schemes, one with additional capacity when the length of the queue exceeded the value of input parameter *IQ* and a second scheme when the waiting time was such that the expected time to complete the remaining work exceeded the remaining scheduled time to perform it. Resource dependency of development teams where the subtasks of read, prepare and work had to be performed in turn by one resource, was managed by resource scheduling to perform communication work at appropriate intervals. Various scenarios were tested with different input values to determine the effects of the various schemes on overall project span time and effort.

Other dependencies indicated in Table 2-2 are the producer/consumer type. In PD this dependency refers to tasks that produce information that other tasks require. We considered this for cases of sequential or one way and reciprocal dependency between tasks. The coordination mechanisms to manage sequential dependency are the sequencing of tasks with varying degrees of overlap and the influence that the communication of interim information has on project performance. We also examined the elimination of the dependency between tasks with the use of set-based coordination and the conditions under which this would be a preferred mechanism. For cases of reciprocal dependency, we examined the relationship of frequency of communication (transfer mechanism) and of the usability of information to the span time of projects with varying rates of uncertainty reduction.

We studied the task/subtask dependency and the manner of task decomposition both with regard to the relative size of tasks and the number of subtasks.

This thesis studied how to improve present coordination mechanisms:

- task decomposition and sequencing;
- systems to support information processing and decision making;
- the number and timing of approval cycles employed at various stages;

- the type, frequency, and efficiency of information exchange;
- the manner in which shared resources are allocated.

7.1 Review of research contributions

This research contributes to knowledge at three levels:

1. The development of a method to quantify the effects of coordination mechanisms under different conditions of uncertainty and task decomposition (division of labour). The following are the novel features of the method.

    a. A stochastic, discrete event model of the PD process was developed.

    b. In the model, information in the PD process was discretized and each unit of information was treated as an entity that followed its own processing thread in the simulation.

    c. The process path that each information entity takes was influenced in the model by its originating task's work progress and information received from other tasks. In this way the model could capture the effects of impediments to information flow in the process and the extent to which these impediments could be mitigated by improved coordination.

    d. Unlike previous models of engineering design, the amount and type of iteration was not an input to the model, but rather came about as a consequence of the dependency relationships between tasks and the dynamics of the information flow that took place as the simulated process unfolded.

    e. The model related the generation of churn to increases in uncertainty of input information in dependent tasks.

    f. A stochastic starvation condition was incorporated to capture the effects of lack of information flow on task progress. Managerial

intervention was simulated if a deadlock condition occurred when reciprocally dependent tasks were starved for information from each other.

g. Rework due to unsuccessful achievement of intermediate outcomes (design version rework) was generated in the simulation by insufficient uncertainty reduction in tasks which was in turn a result of the characteristics of the information exchange and work accomplished during simulation. Feedback was simulated with the reduction of uncertainty profiles based on achievements in earlier iterations.

2. The study of the effects of various coordination mechanisms on product development where the work presented here has been able to look further into the mechanisms driving performance.

   a. The degree of overlapping tasks with sequential dependency under different conditions of epistemic and aleatory uncertainty.

   b. The suitability of set-based coordination to reduce iterations for various scenarios.

   c. The optimal interval of interim information exchange between dependent tasks under different conditions of epistemic and aleatory uncertainty.

   d. The importance of reduction of delays due to communication latency and resource constraints.

   e. The effectiveness of various schemes to manage critical resource constraints.

   f. The value of changing the number and relative size of differentiated tasks in the work breakdown structure.

   g. The suitability of increasing the number of interim design reviews where intermediate outcomes are evaluated.

h. The value in reducing the effort required in communication with the use of a virtual product model.

3. The development and validation of the following mechanisms that can greatly improve coordination among partners doing PD and reduce development times were made possible through insights gained when modeling engineering design.

   a. Overlap sequentially dependent tasks when there is sufficiently frequent, interim information exchange to the extent warranted by the rates of reduction of epistemic uncertainty in tasks.

   b. Use set-based coordination when there are a small number of possible solutions to upstream tasks.

   c. Structure projects such that interdependent task sizes are similar.

   d. Provide sufficient resource capacity early enough in critical support tasks by anticipating workload requirements.

   e. Implement 'scrum' methods where high uncertainty occurs to enable intense coordination and keep PD projects on track.

   f. Adopt policies and implement systems to reduce delays in information flow between dependent tasks.

   g. Adopt policies and implement systems to reduce effort in communication of information between dependent tasks.

7.2 Benefits of this research

7.2.1 For industry

The model of the product development process as a system of tasks dependent on each other for developing information should lead to greater understanding of the drivers of PD project performance. The key dimensions defining the product development system: the dependency strength between tasks; the profile of reduction of epistemic uncertainty in each task; and the magnitude of aleatory

uncertainty in each task; provide the criteria to tailor the appropriate coordination mechanisms to employ in each part and stage of a PD process. Insights provided by the simulations illustrate how interactions of the various elements of the product development system can combine to impact the overall performance characteristics of a project.

The understanding of the drivers of span time and effort performance of PD processes can be employed to better plan PD projects and change industry behaviour in the implementation of coordination mechanisms. The structure of PD projects can be improved to incorporate more self-management of development teams when working on highly interdependent groups of tasks, allowing them to more rapidly respond to developments that occur during the process. The timing and frequency of interim design reviews should be reconsidered in light of the role these can play in maintaining the direction of the design work on the correct path towards the organization's goals. The sequencing of tasks should be reconsidered in view of the dependencies between tasks, the reduction of uncertainty of information of upstream tasks, and the ability of tasks to make progress with interim information from upstream. The non-linear effects on span time of the multiplicity of impediments to information flow should highlight the need to find ways to reduce each of these to eliminate bottlenecks in the PD process. Planning for resources should consider the time varying nature of information flow on the workloads of project participants to better man the tasks when required.

All of these insights point to the need for greater process knowledge by project planners, managers, and participants. Knowing who makes use of the information developed in each task empowers project participants to make proper decisions about releasing interim information, expediting it to those waiting to receive it, and the importance of conveying its precision. Process knowledge more readily allows participants to actively solicit the information they require from those that are generating it, and makes for a more proactive and responsive project.

Insights from the model can also be of benefit in the management of complex PD processes in industry through the study of specific types of managerial intervention to various situations that arise during the course of a process. Using the methods developed here, these interventions can be simulated and their impact on the process can be foreseen leading to a greater understanding of the potential knock on effects. Of important benefit as well is the ability to study a particular PD process using the model to gain insights into the important mechanisms driving span time performance.

7.2.2 For academia

This research is of benefit to academia in demonstrating the value and feasibility of a discrete event model as described here to better understand complex product development processes. This thesis shows that aggregate characteristics of PD processes can be derived from the analysis of the mechanisms pertaining to the flow of information. The focus on information exchange enables the investigation of the effects of various coordination mechanisms and managerial actions on the characteristics of the process that affect span time and effort.

Iteration in engineering design is an important driver of PD project characteristics. In this thesis, iteration as the refinement of intermediate outcomes was incorporated into the requirement for the total amount of information exchange between dependent tasks. This requirement formed an important part of the determination of the state of progress in a task during the process, and enabled the calculation of interim uncertainty of information and ultimately the generation of rework. This novel method of modeling iteration permitted a more detailed examination of its mechanisms and the way in which the management of the process influences overall span time and effort.

Uncertainty is also intrinsic in PD, but a portion of uncertainty in processes is caused by delays and complexities of the process itself and is not inherent in the tasks. Here, the uncertainty inherent in tasks was modeled with their interdependency relationships and with profiles defining the functional relation of

epistemic uncertainty and the state of task progress. This approach enabled the derivation of uncertainty of information generated during the process, and the assignment of attributes of epistemic and aleatory uncertainty to discrete units of information in the model. These units of information, by virtue of the individual paths they followed during the process, thus manifested the effects of the delays and complexities of the process on the uncertainty of information and its effects on the process itself. The approach developed here allows further study of the way in which uncertainty can influence iteration and process progress in PD.

Key elements that were incorporated into the model were:

- the relation of uncertainty of information generated by a task at any time to the proportion of work done and input information received by the task at that time;
- the relation between the amount of technical work done in a task and the proportion of information generated;
- the relationship between the amount of work that can be done in a task before it is starved for information;
- the criteria for success or failure in a design review when only proportions of uncertainty, information exchange, and work required are measurable;
- the criteria for design iteration rework when the content of the information are unknown, but only the changes in its uncertainty are available for comparison;
- the model for feedback addressing only the relative reduction in uncertainty of information and not its actual content.

Although it is not possible, in the general case, to state the precise relationships for each of these items, we were able to infer relationships about their probability distributions, and with the use of a stochastic model, were able to obtain results for the average behaviour of PD processes.

The approach presented in this thesis can be of benefit in further research in engineering design, and points to the possibility of incorporating more complex

algorithms to simulate the effects of specific actions by managers. Managerial intervention that is based on the response to changing project status can be modeled with the use of the values of state and system variables that are updated as the simulation unfolds.

7.3 Opportunities for further research

Design of controlled experiments. Data from product development processes are needed in order to calibrate models and provide further insights about drivers of behaviour such as iteration, communication practices, communication frequency versus task dependency, and managerial intervention. There are difficulties in obtaining data because there are few physical manifestations of PD process state, and therefore, obtaining data is often limited to cost data of process participants (effort) or through interviews. Information about the process structure itself is often incomplete and of insufficient detail. Therefore, research of short (less than six months) cycle development processes should be carried out in an PD organization. The PD organization must participate in the capture of detailed information about the process structure. Characteristics of individual tasks must be identified, information requirements of each task must be captured, and estimates of uncertainty, sensitivity, uncertainty reduction profiles, and aleatory uncertainty must be gathered for each task. Effort spent on individual tasks, the span time of each phase, and the entire process cycle would be required for each PD project studied. Data would be required from several PD projects.

Modeling scenarios with multiple concurrent PD projects. The model developed here should be expanded to simulate several concurrent PD projects simultaneously. This would allow the study of scheduling problems that often occur in PD departments. Often resources with requisite expertise are constrained and the organization must balance throughput and effort. A model of this type would be helpful to industry by examining the trade-offs among resource requirements, throughput and maintaining schedules and priorities.

8 References and Bibliography

Adler, P. S. and K. B. Clark (1991). "Behind the Learning Curve: A Sketch of the Learning Process." Management Science **37**(3): 267-281.

Allen, T. J. (2007). "Architecture and Communication among Product Development Engineers." California Management Review **49**(2): 23-41.

Allen, T. J. and G. W. Henn (2007). The Organization and Architecture of Innovation, Elsevier, Butterworth-Hienemann.

Antonsson, E. K. and K. N. Otto (1995). "Imprecision in engineering design." Transactions of the American Society of Mechanical Engineers, Journal of Mechanical Design **117**: 25-25.

Aughenbaugh, J. Matthew, Paredis and C. J. J. (2006). "The Value of Using Imprecise Probabilities in Engineering Design." Journal of Mechanical Design **128**(4): 969-979.

Bavelas, A., Ed. (1960). Communication and Organization. Management Organization and the Computer. Chicago, Chicago Free Press.

Beauregard, Y., N. Bhuiyan and V. Thomson (2011). "Post-Certification Engineering Taxonomy and Task Value Optimization in the Aerospace Industry." Engineering Management Journal **23**(1): 86-100.

Bhuiyan, F. (2001). Dynamic Models of Concurrent Engineering Processes and Performance. PhD thesis. McGill University. Montreal, Canada.

Bhuiyan, N., D. Gerwin and V. Thomson (2004). "Simulation of the New Product Development Process for Performance Improvement." Management Science **50**(12): 1690-1703.

Bhuiyan, N., V. Thomson and D. Gerwin (2006). "A Systematic Framework for Implementing Concurrent Engineering." Research and Technology Management **49**(1): 38-43.

Black, L. J. and N. P. Repenning (2001). "Why firefighting is never enough: preserving high-quality product development." System Dynamics Review **17**(1): 33-62.

Box, G. E. P. (1979). Robustness in the strategy of scientific model building, Wisconsin University- Madison Mathematics Research Center.

Brooks, F. P. J. (1975). The Mythical Man-Month, Addison-Wesley Publishing Company.

Browning, T. R. (1998). Modeling and Analyzing Cost, Schedule, and Performance in Complex System Product Development. PhD Thesis. Technology, Management, and Policy Program. Cambridge MA, Massachusetts Institute of Technology. 299.

Browning, T. R. (1999). "Sources of Schedule Risk in Complex System Development." Systems Engineering **3**: 129-142.

Browning, T. R., E. Fricke and H. Negele (2006). "Key Concepts in Modeling Product Development Processes." Systems Engineering **9**(2): 104-128.

Browning, T. R. and R. V. Ramesh (2007). "A survey of activity network-based process models for managing product development projects." Production and Operations Management **16**(2): 217-240.

Carley, K. M. and Z. Lin (1997). "A theoretical study of organizational performance under information distortion." Management Science **43**: 976-997.

Carrascosa, M., S. D. Eppinger and D. E. Whitney (1998). Using the design structure matrix to estimate product development time. Proceedings of the ASME Design Engineering Technical Conferences (Design Automation Conference). Atlanta, Georgia, USA.

Carroll, G. R. and J. R. Harrison (1994). "On the historical efficiency of competition between organizational populations." American Journal of Sociology **100**: 720-749.

Chalupnik, M. J., D. C. Wynn and P. J. Clarkson (2009). Approaches to mitigate the impact of uncertainty in development processes. International Conference on Engineering Design P. n. ICED'09/464. Stanford, CA.

Clark, K., B. Chew and T. Fujimoto (1987). "Product Development in the World Auto Industry." Brookings Papers on Economic Activity **3**: 729-771.

Cockburn, A. (2006). Agile Software Development: The Cooperative Game, 2006, Addison Wesley.

Collins, S. T., A. A. Yassine and S. P. Borgatti (2009). "Evaluating Product Development systems using network analysis." Systems Engineering **12**(1): 55-68.

Cota, B. A. and R. G. Sargent (1992). "A modification of the process interaction world view." ACM Trans. Model. Comput. Simul. **2**(2): 109-129.

Crowston, K. (2003). "The Evolution of High-Reliability Coordination Mechanisms For Collision Avoidance." Journal of Information Technology Theory and Application **5**(3): 1-29.

Danilovic, M. and T. R. Browning (2007). "Managing complex product development projects with design structure matrices and domain mapping matrices." International Journal of Project Management **25**(3): 300-314.

Devroye, L. (1986). Non-Uniform Random Variate Generation. New York, Springer-Verlag.

Eckert, C., J. Clarkson and M. Stacey (2001). Information flow in engineering companies: Problems and their causes. International Conference on Engineering Design Glasgow, U.K.**:** 43-50.

Eppinger, S. D., M. V. Nukala and D. E. Whitney (1997). "Generalised Models of Design Iteration Using Signal Flow Graphs." Research in Engineering Design **9**: 112-123.

Eppinger, S. D., D. E. Whitney, R. P. Smith and D. A. Gebala (1994). "A model-based method for organizing tasks in product development." Research in Engineering Design **6**(1): 1-13.

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

Ford, D. N. and J. D. Sterman (1998). "Expert knowledge elicitation to improve formal and mental models." System Dynamics Review **14**(4): 309-340.

Ford, D. N. and J. D. Sterman (2003). "The Liar's Club: Concealing Rework in Concurrent Development." Concurrent Engineering **11**(3): 211-219.

Fredriksson, B. (1994). "Systems Engineering–A Holistic Approach to Product Development." Griffin **94**: 95-105.

Galbraith, J. R. (1977). Organization Design. Reading, Mass., Addison-Wesley Pub. Co.

Grebici, K., D. C. Wynn and P. J. Clarkson (2008). Modelling the relationship between uncertainty levels in design descriptions and design process duration. International Conference on Integrated Design and Manufacturing in Mechanical Engineering. Beijing, China.

Ha, A. Y. and E. L. Porteus (1995). "Optimal Timing of Reviews in Concurrent Design for Manufacturability." Management Science **41**(9): 1431-1447.

Hage, J., M. Aiken and C. B. Marrett (1971). "Organization structure and communication." American Sociological Review **Oct.**: 860-871.

Hales, C. and S. Gooch (2004). Managing engineering design, Springer Verlag.

Hammer, M. (2001). "Seven insights about processes." Proc Strategic Power Process Ensuring Survival Creating Competitive Advantage, Boston.

Harrison, J. R., Z. Lin, G. R. Carroll and K. M. Carley (2007). "Simulation Modeling in Organizational and Management Research." The Academy of Management Review (AMR) **32**(4): 1229-1245.

Harter, D. E., M. S. Krishnan and S. A. Slaughter (2000). "Effects of Process Maturity on Quality, Cycle Time, and Effort in Software Product Development." Management Science **46**(4): 451-466.

Hounshell, D. (1984). From the American System to Mass Production, 1800-1932. Baltimore, Johns Hopkins University Press.

Hykin, D. H. W. and L. C. Laming (1975). "Design case histories: Report of a field study of design in the United Kingdom engineering industry." Proceedings of the Institution of Mechanical Engineers 1847-1982 **189**(1975): 203-211.

Jing, N. N. and C. Yang (2009). The interrelationship among quality planning, knowledge process and new product development performance.

International Conference on Industrial Engineering and Engineering Management. Beijing, China.**:** 1051-1055.

Karniel, A. and Y.Reich (2009). "From DSM-based planning to design process simulation: A review of process scheme logic verification issues." IEEE Transactions on Engineering Management **56**(4): 636-649.

Kleijen, J. P. C. (1999). Validation of models: statistical techniques and data availability. 1999 Winter Simulation Conference Proceedings, ACM New York, NY, USA.

Kline, S. J. (1985). "Innovation is not a linear process." Research Management **28**(4): 36-45.

Krishnan, V. (1996). "Managing the Simultaneous Execution of Coupled Phases in Concurrent Product Development." IEEE Transactions on Engineering Management **43**(2): 210-217.

Krishnan, V., S. D. Eppinger and D. E. Whitney (1997). "A model-based framework to overlap product development activities." Management Science **43**(4): 437-451.

Lant, T. K. and S. J. Mezias (1992). "An organizational learning model of convergence and reorientation." Organization Science **3**: 47-71.

Law, A. M. and W. D. Kelton (2000). Simulation Modeling and Analysis, McGraw Hill.

Levitt, R. E., J. Thomsen, T. R. Christiansen, J. C. Kunz, Y. Jin and C. Nass (1999). "Simulating project work processes and organizations: Toward a micro-contingency Theory of Organizational Design." Management Science **45**(11): 1479-1495.

Loch, C. H. and C. Terwiesch (2005). "Rush and be wrong or wait and be late? A model of information in collaborative processes." Production and Operations Management **14**(3): 331–343.

Maier, A. M., M. Kreimeyer, C. Hepperle, C. M. Eckert, U. Lindemann and P. J. Clarkson (2008). "Exploration of correlations between factors influencing

communication in complex product development." Concurrent Engineering-Research and Applications **16**(1): 37-59.

Malone, T. W. and K. Crowston (1994). "The interdisciplinary study of coordination." ACM Computing Surveys (CSUR) **26**(1): 87-119.

Malone, T. W., K. Crowston, J. Lee and B. Pentland (1999). "Tools for inventing organizations: Toward a handbook or organizational processes." Management Science **45**(3): 425-443.

Mark, N. (2002). "Cultural transmission, disporoportionate prior exposure, and the evolution of cooperation." American Sociological Review **67**: 323-344.

Martinez, J. I. and J. C. Jarillo (1989). "The Evolution of Research on Coordination Mechanisms in Multinational Corporations." Journal of International Business Studies **20**(3): 489-513.

Mihm, J. and C. H. Loch (2006). Spiraling out of control:Problem-solving dynamics in complex distributed engineering projects. Complex Engineering Systems. D. Braha, A. Minai and Y. Bar-Yam. New York, Perseus Books.

Mihm, J., C. H. Loch and A. Huchzermeier (2003). "Problem-Solving Oscillations in Complex Engineering Projects." Management Science **46**(6): 733-750.

Mintzberg, H. (1979). The Structuring of Organizations. Englewood Cliffs, N.J., Prentice-Hall.

Nandakumar, P., S. M. Datar and R. Akella (1993). "Models for measuring and accounting for cost of conformance quality." Management Science **39**(1): 1-16.

O'Donovan, B. D., P. J. Clarkson and C. M. Eckert (2003). Signposting: Modelling uncertainty in design processes. International Conference on Engineering Design. Stockholm**:** 1-10.

Oberkampf, W. L., J. C. Helton, C. A. Joslyn, S. F. Wojtkiewicz and S. Ferson (2004). "Challenge problems: uncertainty in system response given

uncertain parameters." Reliability Engineering & System Safety **85**(1-3): 11-19.

Oppenheim, B. W. (2004). "Lean Product Development Flow." Systems Engineering **7**(4): 352-376.

Pahl, G. and W. Beitz (1996). Engineering design: A Systematic Approach. London, New York, Springer.

Pajerek, L. (2000). "Processes and organizations as systems: when the processors are people, not pentiums." Systems Engineering **3**(2): 103-111.

Peters, T. (1986). "The Mythology of Innovation, A Skunkworks Tale." Strategic Planning: Selected Readings. Ed. J. William Pfeiffer. San Diego, CA: University Associates, Inc: 485-500.

PMI (2001). A Guide to the Project Management Body of Knowledge, Project Management Institute Publications.

Port, O. (1989). Pssst! Want a Secret for Making Superproducts? Business Week. **October 2, 1989:** 106-107.

Porter, L., E. Lawler and R. Hackman (1975). Behaviour in organizations. New York, McGraw-Hill.

Pugh, S. (1995). Total design: integrated methods for successful product engineering, Addison-Wesley Wokingham, UK.

Rechtin, E. (1991). Systems Architecting: creating and building complex systems, Prentice Hall.

Reinertsen, D. (1999). "Lean thinking isn't so simple." Electronic Design **47**(10): 48H.

Ritter, F. E. and L. J. Schooler (2002). The Learning Curve. International encyclopedia of the social and behavioural sciences. Amsterdam, Pergamon**:** 8602-8605.

Safoutin, M. J. (2003). A methodology for empirical measurement of iteration in engineering design processes. PhD thesis. University of Washington. Seattle, Washington.

Safoutin, M. J. and R. P. Smith (1996). The iterative component of design. International Conference on Engineering and Technology Management Vancouver, Canada**:** 564-569.

Schmidt, J. W. and R. E. Taylor (1970). Simulation and Analysis of Industrial Systems. Homewood Illinois, Richard D. Irwin.

Simon, H. A. (1969). The Science of the Artificial. Boston, MA., MIT Press.

Smith, R. P. and S. D. Eppinger (1997). "A Predictive Model of Sequential Iteration in Engineering Design." Management Science **43**(8): 1104-1120.

Sobek, D. K. I. (1996). "A set-based model of design." Mechanical Engineering **118**(7): 4-7.

Steward, D. V. (1991). Planning and managing the design of systems. Conference on Technology Management: the New International Language. Portland, Oregon**:** 189-193.

Susman, G. I. (1992). Integrating design and manufacturing for competitive advantage, Oxford University Press, USA.

Suss, S., K. Grebici and V. Thomson (2010). The Effect of Uncertainty on Span Time and Effort within a Complex Design Process. Modelling and Management of Engineering Processes. P. Heisig, J. Clarkson and S. Vajna. London Springer**:** 77-88.

Sweat, J. (2001). "Communication Aids Design." Information Week, from http://www.informationweek.com/news/6508331.

Swink, M. L., C. J. Sandvig and V. A. Mabert (1996). "Customizing Concurrent Engineering Processes: Five Case Studies." Journal of Product Innovation Management **13**(3): 229-244.

Terwiesch, C., C. H. Loch and A. D. Meyer (2002). "Exchanging Preliminary Information in Concurrent Engineering: Alternative Coordination Strategies." Organization Science **13**(4): 402-419.

Thompson, J. D. (1967). Organizations in action. New York, McGraw Hill.

Thunnissen, D. P. (2004). Propagating and Mitigating Uncertainty in the Design of Complex Multidisciplinary Systems. PhD thesis., California Institute of Technology, Pasadena, California

Twigg, D. (1998). "Managing product development within a design chain." International Journal of Operations & Production Management **18**(5): 508-524.

VanDeVen, A. H., A. L. Delbecq, Richard and J. Koenig (1976). "Determinants of Coordination Modes within Organizations." American Sociological Review **41**(2): 322-338.

Vangheluwe, H. (2008). Discrete Event Modelling and Simulation. Modelling and Simulation Course Notes Computer Science, McGill University, Montreal Canada

Von-Hippel, E. and M. J. Tyre (1995). "How learning by doing is done: problem identification in novel process equipment." Research Policy **24**(1): 1-12.

Wheelwright, S. and K. Clark (1992). Revolutionizing Product Development. New York, The Free Press.

Whitney, D. E. (1990). "Designing the design process." Research in Engineering Design **2**(1): 3-13.

Wood, K. L., E. K. Antonsson and J. L. Beck (1990). "Representing imprecision in engineering design: comparing fuzzy and probability calculus." Research in Engineering Design **1**(3): 187-203.

Wynn, D. C. (2007). Model-based approaches to support process improvement in complex product development. PhD thesis. Cambridge University. Cambridge, U.K.

Yassine, A. A. and D. Braha (2003). "Complex Concurrent Engineering and the Design Structure Matrix Method." Concurrent Engineering: Research and Applications **11**(3): 165-176.

Yassine, A. A., D. Falkenburg and K. Chelst (1999). "Engineering design management: an information structure approach." International Journal of Production Research **37**(13): 2957-2975.

Yassine, A. A., N. Joglekar, D. Braha, S. Eppinger and D. Whitney (2003). "Information hiding in product development: the design churn effect." Research in Engineering Design **14**(3): 145-161.