Resource Dependence in

Product Development Improvement Efforts

Nelson P. Repenning

Department of Operations Management/System Dynamics Group
Sloan School of Management
Massachusetts Institute of Technology
E53-339, 30 Wadsworth St.
Cambridge, MA USA 02142

Phone 617-258-6889; Fax: 617-258-7579; E-Mail: <nelsonr@mit.edu>

December 1997

version 0.5

Support has been provided by the National Science Foundation, grant SBR-9422228, The Center for Innovative Product Development at MIT, the Harley-Davidson Motor Company and the Ford Motor Company. Thanks to Drew Jones for providing useful comment on early versions of the model. Special thanks to Don Kieffer of Harley-Davidson for providing the catalyst to this study.

For more information on the research program that generated this paper, visit our World Wide Web site at http://web.mit.edu/jsterman/www/.

Abstract

Managers and scholars have increasingly come to recognize the central role that design and engineering play in the overall process of delivering products to the final customer. Give their critical role, it is not surprising that design and engineering processes have increasingly become the focus of improvement and redesign efforts. In this paper one hypothesis for the difficulty of making the transition between existing and new development processes is developed in the form of a simulation model. The starting point for the analysis is the observation that in an organization in which multiple products are being developed, scarce resources must be allocated between competing projects in *different* phases of the development process. The scarcity of resource creates interdependence; the performance of a given project is not independent of the outcomes of other projects. Resource dependence among projects leads to the two main ideas presented in the paper: First, the existence of resource dependence between projects, coupled with locally rational decision making leads to an undesirable allocation of resources between competing activities, and second, this error is self-reinforcing. Strategies for mitigating these undesirable dynamics are also considered.

1. Introduction

Managers and scholars have increasingly come to recognize the central role that design and engineering play in the overall process of delivering products to the final customer. Researchers analyzing a wide range of business problems have arrived at a similar conclusion; the high leverage point for improving many manufacturing organizations lies in product design engineering. Wheelwright and Clark (1992) and Clark and Fujimoto (1991) both argue that a firm's product development capability is a significant determinant of its performance in a wide range of areas including productivity, quality, cost, and flexibility. Zangwill (1993) suggests the ability to quickly deliver innovative product to market is the single biggest factor separating top firms from the rest. Wheelwright and Clark (1992:*xi*) write "Time after time that search [for the determinants of firm performance] led us to product design, the engineering organization, and the speed and effectiveness of the development process as the explanation for superior performance." Similarly, Dertouszos, Lester and Solow (1989) find that weaknesses in the area of product design and engineering are partially responsible for the low growth rate of productivity in American industry.

Give their critical role, it is not surprising that design and engineering processes have increasingly become the focus of improvement and redesign efforts. A variety of authors suggest ways to develop new products faster, cheaper, and with higher quality. Wheelwright and Clark (1992) focus on the role of general managers and Wheelwright and Clark (1995) re-frame their earlier work for senior leaders. Ulrich and Eppinger (1995) provide a detailed, operational perspective, and Zangwill (1993) list seven key

factors that characterize the processes used by the top performers. These are not purely theoretical efforts. All of these books are based on the detailed study of many firms and their attempts to improve their development processes. Many companies have, at one time or another, undertaken a serious effort to redesign and improve their product development processes (Repenning 1996, Krahmer and Oliva 1996, and Jones 1997 are a few examples).

Yet, while much has been written about the structure of the ideal development process, less has been said about how to implement these ideas. Simon (1962) and Weick (1979) draw a useful distinction between recipes and blueprints. Blueprints describe the ideal structure of a given system, process or organization, but give no information about how it might be constructed. In contrast, recipes, Weick writes, "...provide the means to generate structures that have the characteristics you want." In the product development context, much of the research has focused on better blueprints; new processes that, if implemented, provide higher capability. Less has been written about constructing recipes that create the structures outlined in those blueprints. For example, Wheelwright and Clark (1992), Wheelwright and Clark (1995) and Zangwill (1993) each dedicate only one chapter to the challenges of implementing the blueprint outlined in the rest of their books.

Given the lack of knowledge about recipes for improving PD it is not surprising that many improvement efforts fail to produce substantial results. Repenning and Sterman (1997), Repenning (1996) and Jones (1997) document cases in which firms invested substantial time, money and energy in developing a new product development process yet saw little benefit from their efforts. While many within both companies studied agreed that the new process constituted a better way to develop products, the usage of the new

process was sporadic at best. In the introduction of their second book Wheelwright and Clark (1995) discuss the difficulty that many organizations experienced trying to implement the suggestions from their first volume (Wheelwright and Clark 1992). Unfortunately, in many firms, the new development process, which may have cost millions of dollars to develop, is little more than a three ring binder book sitting on a shelf gathering dust. Designing a new development process is only half the battle. Once the new process has been designed, managers must find a way to ensure that the new process is actually used. The question currently facing many senior manager is, in the words of Weick (1993), "...given a blueprint, what recipe will produce it?"

The process of implementing large scale organizational change has been studied extensively by behavioral scientists and a massive literature has developed on the subject (for overviews see e.g. Van de Ven and Poole 1995; Huber and Glick 1993; Kanter, Jick and Stein 1992). Unfortunately, whereas product development scholars have focused almost exclusively on blueprints, organizational scholars have focused on generic change processes. Less attention has been paid to how change efforts in product development processes might differ from those in other functions such as manufacturing. Dean and Bowen (1994:408) write "...management theorists may have gone too far in emphasizing socio-behavioral over process and technical factors in explaining variation in performance....researchers rarely extended their theories to the social and technical aspects of organizational and process design." To take full advantage of the substantial progress made in process *design*, a better of theory of process *improvement* is needed, one that integrates both the technical and behavioral aspects of product design and engineering.

Resource Dependence and the Tilting Hypothesis

In this paper one hypothesis for the difficulty of making the transition between existing and new development processes is developed. The starting point for the analysis is the observation that in an organization in which multiple products are being developed, scarce resources must be allocated between competing projects in *different* phases of the development process. The scarcity of resource creates interdependence; the performance of a given project is not independent of the outcomes of other projects. Instead, a decision to change the level of resources devoted to a given project affects every other project currently underway and, potentially, every project to come.

Resource dependence among projects leads to the two main ideas presented in the paper: First, the existence of resource dependence between projects, coupled with locally rational decision making leads to an undesirable allocation of resources between competing activities, and second, this error is self-reinforcing. The structure of the product development system, rather than correcting the initial error in resource allocation, amplifies it, creating a vicious cycle that drives the system to a low performance level The phenomenon is termed the 'tilting' hypothesis to capture the idea that the distribution of resources among projects is likely to be skewed towards those closer to introduction into the market place and that this bias is self-reinforcing. Thus, once the distribution of resources begins to 'tilt' towards the projects closer to its launch into the market, the imbalance is amplified to the point where much of the organization's resources are focused on 'fire-fighting' and delivering current projects, little time is invested in up-front activities for future projects.

The tilting hypothesis has important implications for the implementation of new tools and processes. Up-front activities such as documenting customer requirements and establishing technical feasibility play a critical role in many suggested blueprints for product development (Wheelwright and Clark 1992, Ulrich and Eppinger 1995, Zangwill 1993). Organizations stuck in the 'fire fighting' mode allocate most of their resources to fixing problems in products late in their development cycle. Introducing tools such as Quality Function Deployment (Hauser and Clausing 1988) have little value if they do not get used. Prior to taking full advantage of many of these innovations, particularly those that focus on the early phases of the development process, the allocation of resources between current and advanced projects must be redressed. In fact, as will be discussed below, the naive introduction of new tools and processes to an existing organization may actually *worsen* the resource imbalance rather than improve it.

There is, however, a simple, easily implementable, policy that greatly increases the organization's robustness to resource tilting. The policy is one example of the class of limiting strategies proposed in Repenning (*in progress*). The essence of limiting strategies is the progressive elimination of the system's ability to compensate for its root cause problems which, in turn, spurs efforts to make fundamental improvements. In this case, a limiting strategy can be implemented in the form of a resource allocation rule. Such a policy does entail important trade-offs between robustness and ideal state performance, but is quite desirable when the possibility of creating improvement via new tools or processes is introduced.

The rest of the paper is organized as follows: in section two the basic model is developed and analyzed. In section three the possibility of improving the process through the use of

new tools is introduced. Section four contains discussion, and section five presents concluding thoughts and future research directions.

2. A Model of Resource Dependence

Model Structure

Overview

The model represents a product development system in which a new product is introduced to the market every twelve months. Twenty-four months are required to develop a given product, so the organization always works on two products at once. New development projects are introduced into the system at twelve month intervals coinciding with the product launch date. The product in the second year of the development cycle (meaning its launch is less than twelve months away) is called the *current* product, and the product in the first year of the development cycle is called the *advanced* product.

A development project is composed of tasks. Although in real product development systems there is a variety of different activities, in the model all of these are aggregated into one generic activity. Similarly, there is only one type of resource needed to accomplish these tasks, engineering hours. These assumptions represent a substantial simplification of any real world process. As will be shown below, however, the model is still capable of generating quite complicated behavior.

There are four key assumptions embedded in the structure of the model:

- The product launch date is fixed. Every twelve months the current product is introduced to the market regardless of its state. There may, for example, be known defects in the design, however the product is still introduced.
- Any time a development task is completed, there is some probability that it is done incorrectly. If so, then it can be corrected through re-work. There is also some chance that re-work is done incorrectly, thus generating more re-work.

- The probability of doing a task incorrectly is *higher* for current work.
- Engineering headcount is fixed. Engineers can adjust for changes in the work load by reducing the number of hours they spend per task.

While there is not sufficient space to give an equation by equation description of the model, the key structures and formulations are discussed below. Readers interested in more detail can contact the author.[1]

Work Accumulations and Flows

The basic stock and flow structure of the model is shown in figure 1. Rectangles represent an accumulation of development tasks in the system. There are four accumulations in each of the advanced and current phases. The solid arrows represent the entry of new projects into the advanced phase and the exit of current products to the market and are only active at the annual advanced-current transition. The solid arrow at the top left of the figure represents the introduction of new tasks into the development system which accumulate in the stock of advanced tasks not completed.

[1] . The model is written using the VENSIM software produced by Ventanna Systems Inc. A run-only version of the model can be downloaded from <http://web.mit.edu/*****>

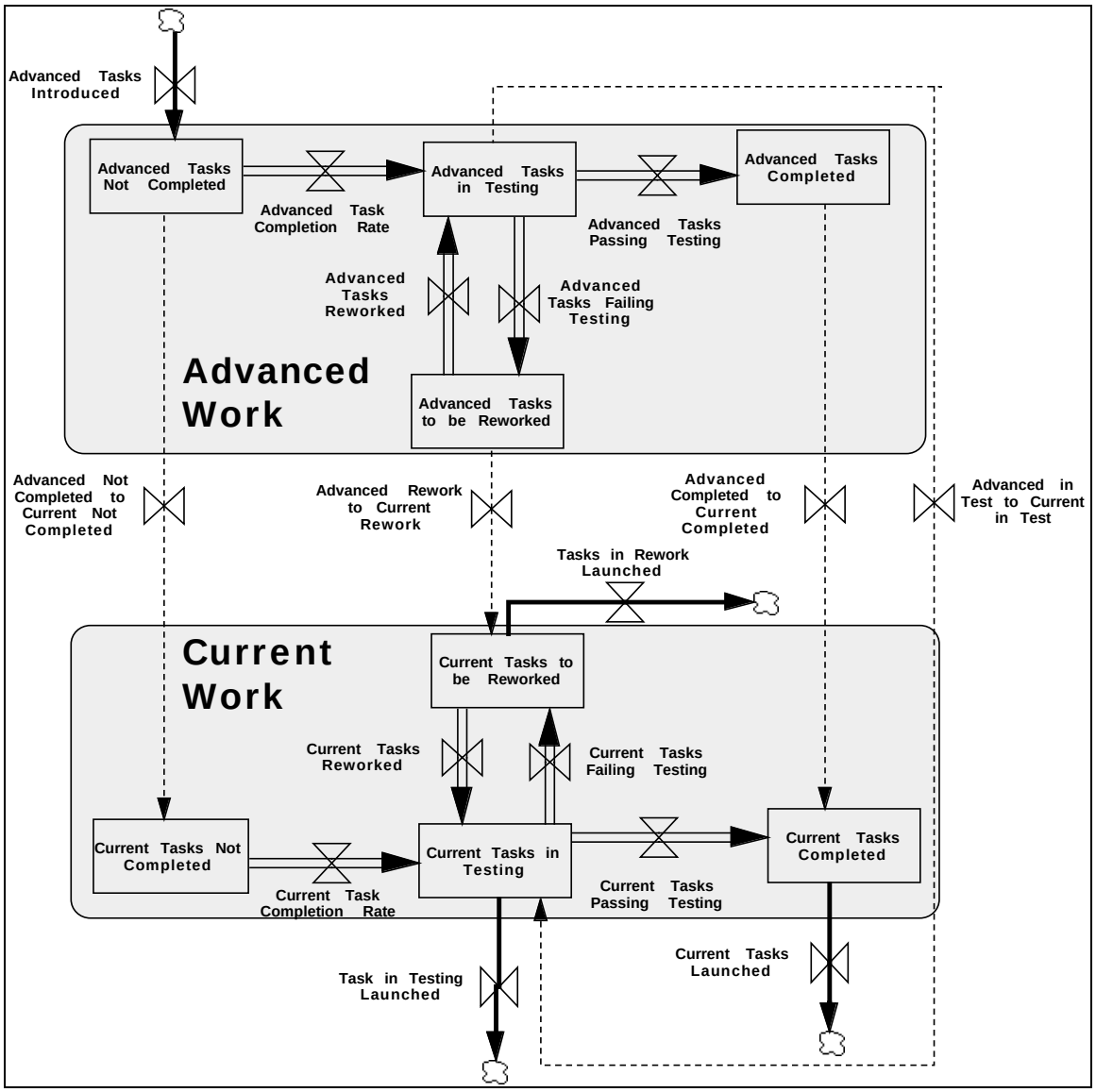

Figure 1: Arrows with valve symbols represent the flow of tasks in the system. Solid arrows represent the entry and exit of tasks to and from the product development process. Flows with double lines represent the continuous flow of work within a given model year. Flows represented by dotted lines represent the annual transition of advanced work to current work.

The double line arrows represent the continuous flow of work between the stocks within the advanced and current phases. The stock of advanced tasks not completed is drained by the advanced task completion rate. Once completed, tasks accumulate in the stock of advanced tasks in testing. If a task passes the test, meaning it was done correctly and is not defective, it flows to the stock of advanced tasks completed. Tasks failing testing

flow to the stock of advanced tasks requiring rework. Once reworked, tasks flow back to the stock awaiting testing.

The dotted arrows represent the annual transition of advanced work to current work. Each stock of advanced tasks has such an outflow. At the transition between model years, the entire contents of each advanced work stock is transferred to the corresponding stock of current work. The task flow in the current work phase is identical to that in the advanced portion. Completion of new tasks drains the stock of current tasks not completed and increases the stock of current tasks awaiting testing. Tasks either pass the test, in which case they flow to the stock of current tasks completed, or fail and move to the stock of current tasks awaiting rework. Once reworked, tasks return to the stock of those awaiting testing. At the end of the model year all current tasks completed are introduced to the market, regardless of their state. These outflows are represented by the solid black arrows.

Defects

Figure 2 shows the structure used to track the number of defective tasks in the stock of tasks awaiting testing. Each time a task is completed or re-worked there is some probability that the task will be done incorrectly and be defective. The new task and rework completion flows are represented by the light gray flow icons. The number of defects introduced through new work or re-work is determined by multiplying the flow of tasks by the probability that a given task is done incorrectly. Once a task is completed or re-worked, its associated defect flows into the stock of defects in advanced testing. Dividing this stock by the total number of tasks currently in testing determines the fraction of defective tasks in testing, which, in turn, determines the rate of tasks passing

and failing. Tasks that fail testing remove one defect from the stock of defects in testing while tasks that pass testing remove no defects from the stock. In real systems there are both type I and type II errors, however, in the model testing is assumed to be perfectly accurate. The formulation is based on the coincident flow structure commonly used in System Dynamics simulation models (Richardson and Pugh 1981). If the launch date is reached and some tasks remain untested, they are still introduced to the market. The number of defective products introduced to the market will be used as the key measure of the system's performance.

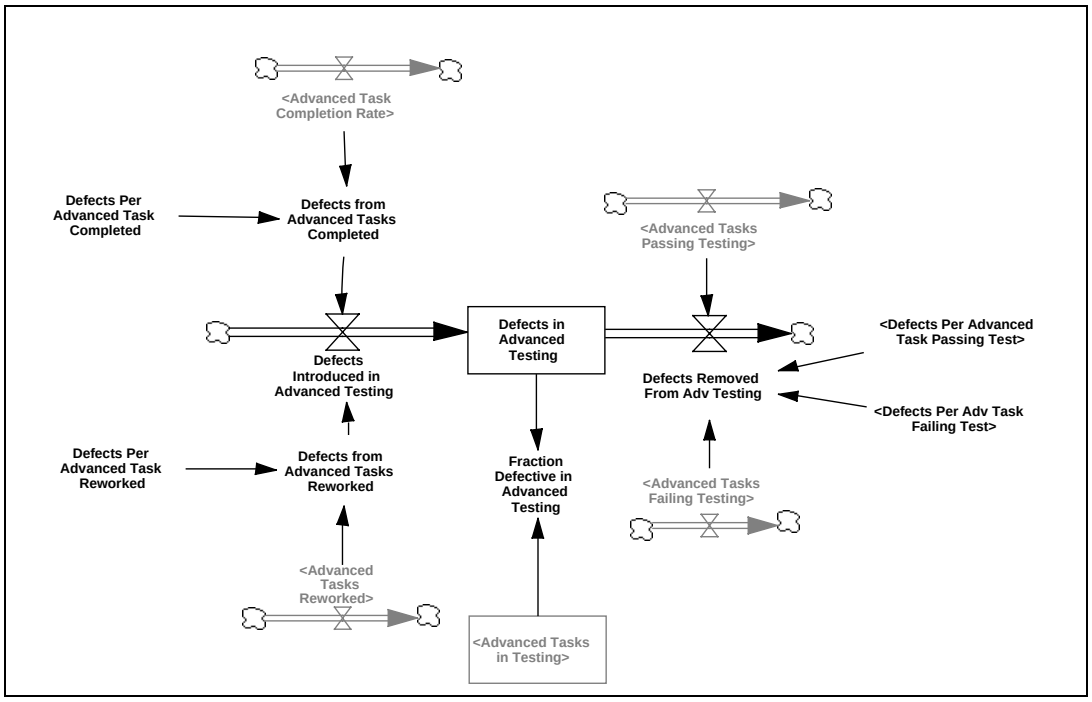

Figure 2

Allocation of Engineering Capacity

The critical decision function in the model is the allocation of engineering capacity between the four types of development work 1) new current tasks (CW), 2) current re-work (CR), 3) new advanced tasks (AW), and 4) advanced re-work (AR). The allocation

of effort between these categories is determined via a two step process. First, an indicated completion rate (ICR_i) is determined for each type of work. The indicated rate of work completion is calculated by dividing the stock of work to be completed by the number of months remaining until product launch. Thus

ICR_i=Work Remaining/# of months to product launch

The formulation represents a decision process in which engineers look at the work to be completed and allocate it evenly over the time remaining. All four of the indicated rates are formulated identically except for the rate of current work completion where the denominator, instead of being the number of months until launch, is the number of months until a prototype deadline (assumed to be six months prior to the product launch).

Second, once the indicated completion rates are determined, the actual rates are calculated. Two rules are used, 1) new work takes priority over re-work, and 2) current work takes priority over advanced work. The actual rate of current work completion (ACWR) is the minimum of the maximum development capacity (MDC) and the indicated completion rate for current work (ICR_{CW}).

$$ACWR=Min(MDC,ICR_{CW})$$

MDC is the maximum rate at which development work can be completed, and is equal to the total number of available engineer hours, AEH, divided by the minimum number of hours required per task (MHT):

$$MDC=AEH/MHT$$

MDC does not measure the rate at which development work can be done properly, but the rate at which work can be completed when engineers are working as quickly as is possible and skipping all steps that do not directly contribute to finishing the design. Such skipped steps might included documenting work and testing new designs.

If, after completing the current new work, there is capacity remaining, then that capacity is allocated to current rework. The actual rate of current re-work (ACRR) is equal to minimum of capacity remaining, MDC-ACWR, and the indicated rate of current re-work (ICR_{CR}):

$$ACRR=Min(MDC\text{-}ACWR,ICR_{CR})$$

Capacity allocated to advance work and advanced re-work is determined in a similar fashion. First, the capacity that *can* be allocated to all advanced tasks (CAT) is determined by subtracting the capacity allocated to current tasks from the normal development capacity (NDC). NDC is the rate at which engineers can complete tasks when they are following the prescribed development process. Thus,

$$CAT=Max(NDC-(ACWR+ACCR),0)$$

The maximum function is important because the rate of work completion can exceed the normal development capacity if the organization has more work to do than is possible under normal operations and completes that work by cutting corners and skipping development steps-- MDC>NDC. If CAT is greater than zero, then the remaining development capacity is allocated to advanced work and then advanced re-work. Normal development capacity is equal to the available engineer hours (AEH) divided by the normal hours required per engineering tasks (NHT):

$$NDC=AEH/NHT$$

This formulation embodies an important assumption: Advanced work does not get completed unless the current work rate is less than the normal development capacity. If the normal development capacity is not sufficient to complete all the outstanding current

tasks, the engineers will reduce their time per tasks to reach the required work rate. However, this does not happen for advanced work. If the organization is working beyond normal capacity, advanced work is simply not done.

Steady State Model Behavior

This section describes the model's steady state behavior. In all runs the base task introduction rate is 1,500 per year.[2] All tasks are introduced in one time step when there are twelve months to product launch.

Base Case

Figure 3 shows base case behavior for advanced and current tasks yet to be completed.

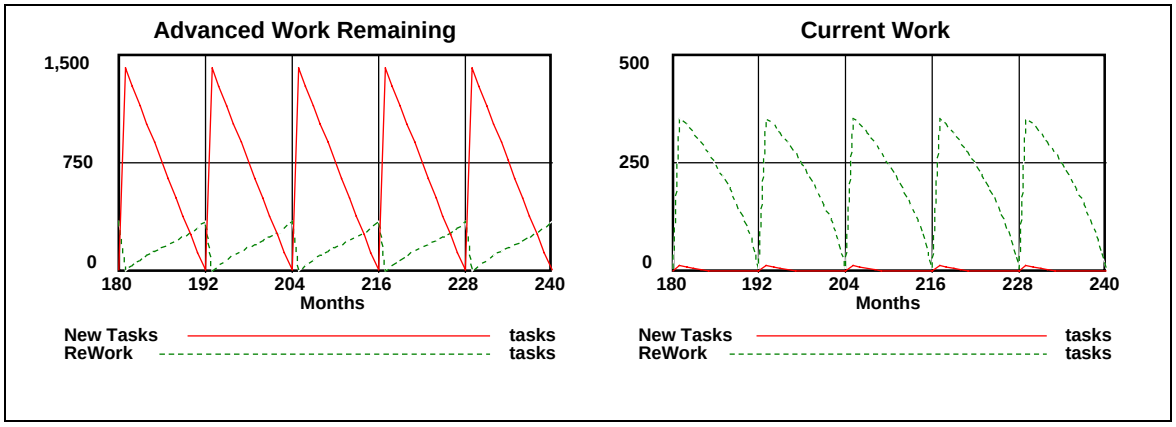

Figure 3

At each twelve month interval 1500 new tasks, constituting a new development project, are introduced. Advanced work is completed at a constant rate throughout the year so the level of advanced tasks to be completed declines linearly. Most new tasks are completed in the advanced phase and only a small number (less than 50) are sent to the current phase uncompleted. The level of re-work to be completed grows during the advanced phase as

tasks done incorrectly are discovered through testing. The small number of uncompleted tasks received in the current phase are quickly completed. Approximately 300 tasks requiring rework are also received from the advanced phase. The rate of re-work completion rises through the year as the product launch date approaches and the organization increasingly focuses on current work in the hope of re-working all the defective tasks before product launch.

Figure 4 shows the average defect level in the advanced and current re-work phases and the allocation of resources between the two phases. The defect fraction in advanced work starts at 40% and declines for the first 9 months of the year. It then increases slightly as the advanced re-work completion rate drops. The fraction of defective tasks in the current phase drops dramatically throughout the year, from 30% to less than 5% improving the quality of the finished product substantially.

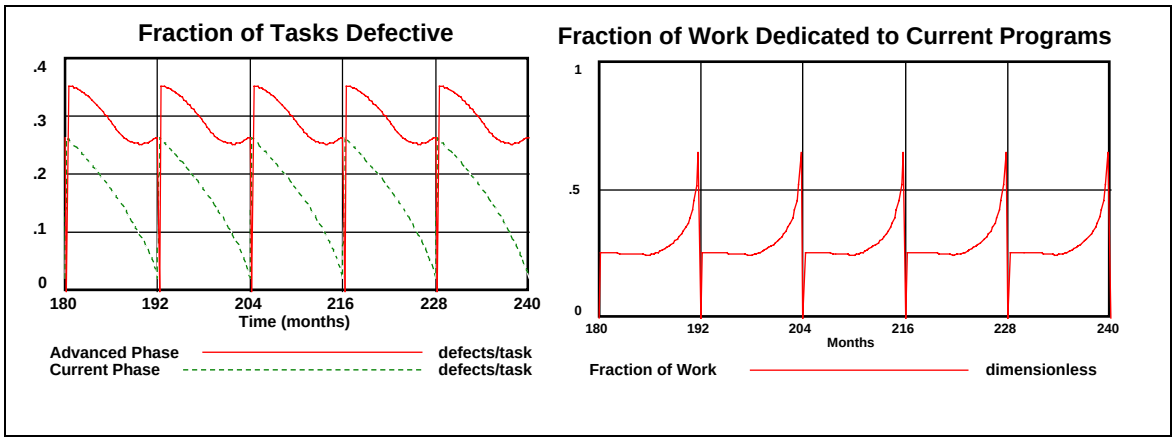

Figure 4

At the beginning of the year approximately 25% of the total capacity is allocated to current programs. The allocation remains constant for the first half of the year and then increases as product launch date approaches. The shift in the allocation of resources leads

---

[2] . The model is simulated using the Euler integration method and runs in months with a time step of .25. In

to an increasing rate of defect reduction in the current phase. The base case depicts a development organization that does the vast majority of its initial development work while the product is in the advanced stage. However, due to low process capability, a significant fraction of that work is done incorrectly. During the current phase many of those defects are corrected through re-work.

Sensitivity Analysis

The model's behavior has been analyzed extensively and a wide range of sensitivity tests have been conducted. Only a few of these tests will be shown here. Readers interested in more detail can consult the accompanying technical report, Repenning (*in progress*). Figures 5 and 6 show the steady state behavior of the model as the task introduction rate is systematically varied. Figure 5 shows how the allocation of resources between the current and advanced phases changes as the task introduction rate is increased. The behavior is highly non-linear. At task introduction rates of 1500 tasks or less the vast majority of resources are allocated to the advanced phase. As the task introduction rates increases beyond 1500 the allocation changes dramatically. The response surface has a steep face, and for small increase beyond 1500 the behavior changes dramatically. At a task introduction rate of 1700 the behavior suddenly changes and the vast majority of resources are allocated to the current phase.

---

each case, it is run for one hundred and eighty months to eliminate transients.

Figure 5

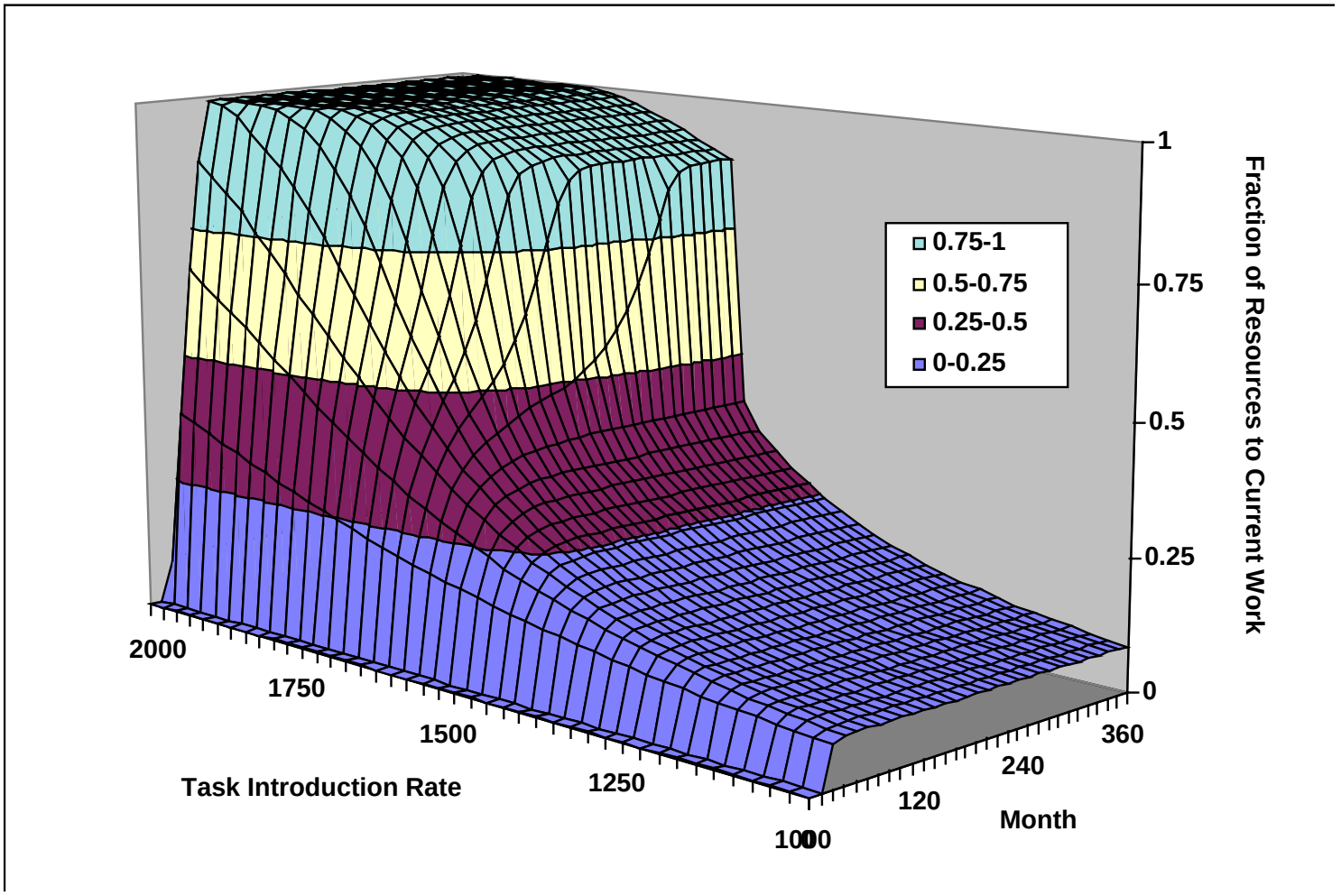
0.75-1
0.5-0.75
0.25-0.5
0-0.25
1
0.75
0.5
0.25
0
Fraction of Resources to Current Work
2000
1750
1500
1250
1000
Task Introduction Rate
0
120
240
360
Month

Figure 6

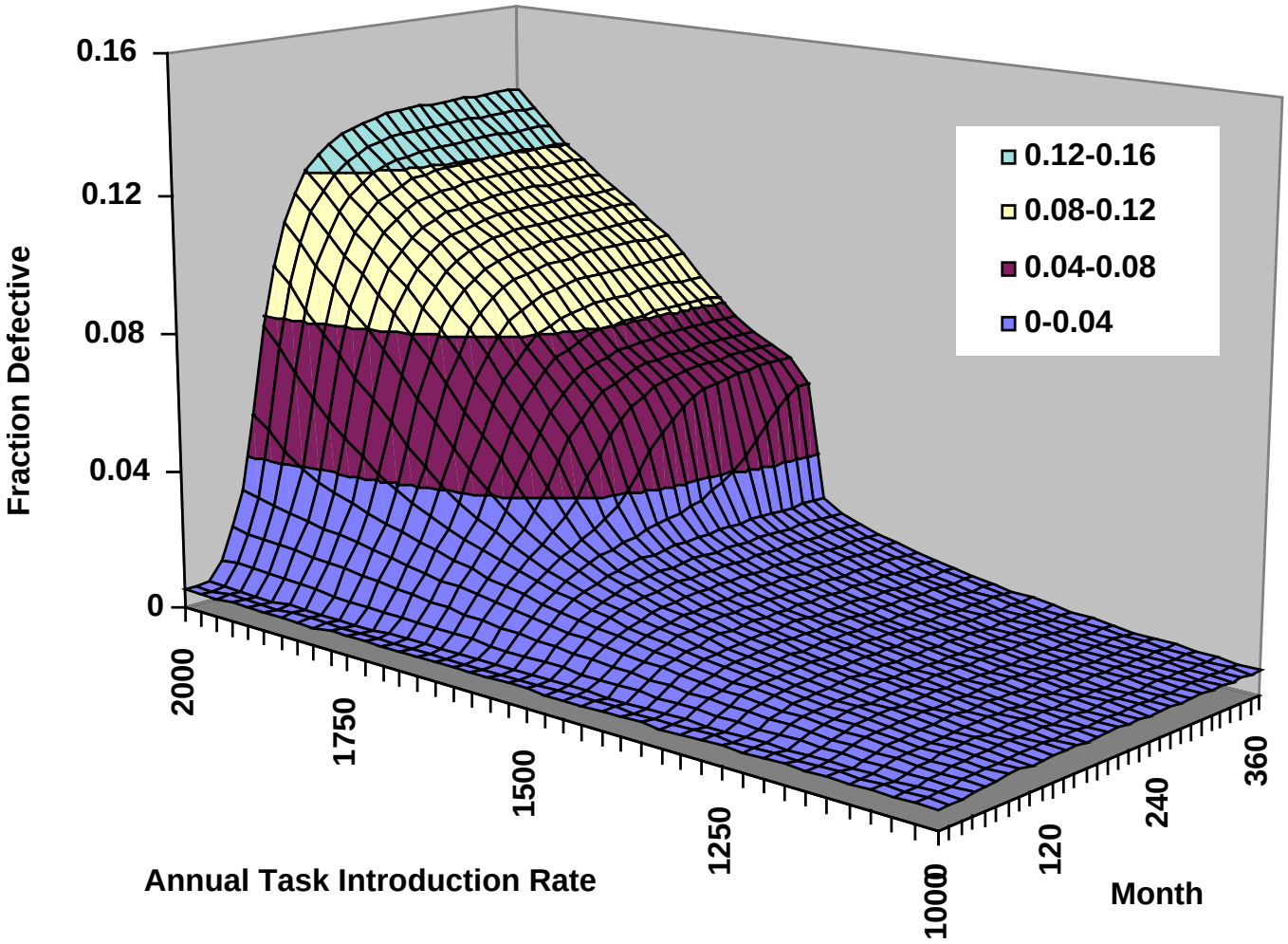

Figure 6 shows the impact of resource loading on the fraction of defective products introduced to the market. The response surface is even more uneven and shows three distinct regimes. Below introduction rates of 1500, the impact of additional tasks is small, the surface has a relatively flat slope. At approximately 1600, the defect introduction rate increase dramatically, and the surface is almost vertical. The change occurs due to the re-allocation of resources caused by the increased work rate. At high

task introduction rates, resources are increasingly allocated to the current phase where the chance of doing a task incorrectly is higher. Beyond the face, the defect introduction rate increases with the task introduction rate, now at a much steeper slope. The increased slope stems from the workload being beyond the system's capacity and additional loading reduces the ability of the organization to complete iterations of the re-work cycle.

The system's response to increases in loading is highly non-linear. Resources are assumed to be finite, so it is not surprising the performance changes with an increased task load. More interesting, however, is how the performance degrades. Under conditions of extreme scarcity, the best performance is achieved by doing *all* the work in the advanced phase, since the chances of doing a task incorrectly are lower. However, the physical structure combined with the myopic decision rule that favors current projects leads to just the opposite outcome, all the work is done in the current phase. Additional work causes the resource distribution to 'tilt' is the wrong direction, reducing the system's capability to create defect free products.

Wheelwright and Clark (1992) discuss the problems associated with overloading a development system. They call the phenomenon the 'canary cage' problem: too many birds in a cage reduces the available resources, food, water and space, and none of the animals thrive. Similarly, in product development, they argue, too many development projects leads to a poor allocation of resources and mediocre products. The model both operationalizes and expands upon the canary cage story. The resource allocation rule implicit in their model allocates resources evenly across all projects. The formulation presented here is an alternative, resources are allocated to projects closer to their launch date. It also expands the 'canary cage' logic by introducing tilting. Increasing the

number of projects in the system has two impacts. First it reduces the resources allocated to any one project and second, it changes the distribution of resources, favoring those closer to launch. Thus, not only are there too few resources, but overloading causes them also to be allocated in an undesirable way.

Transient Behavior

Pulse Test

A particularly instructive test of the model's transient behavior is to introduce a one time increase in the amount of development work introduced each year. In each of the runs presented in this section, the pulse increase takes place at month 192. Figure 7 shows the effect of different pulse sizes on the average fraction of work that is allocated to current programs and Figure 8 shows the fraction of defective products introduced to the market.

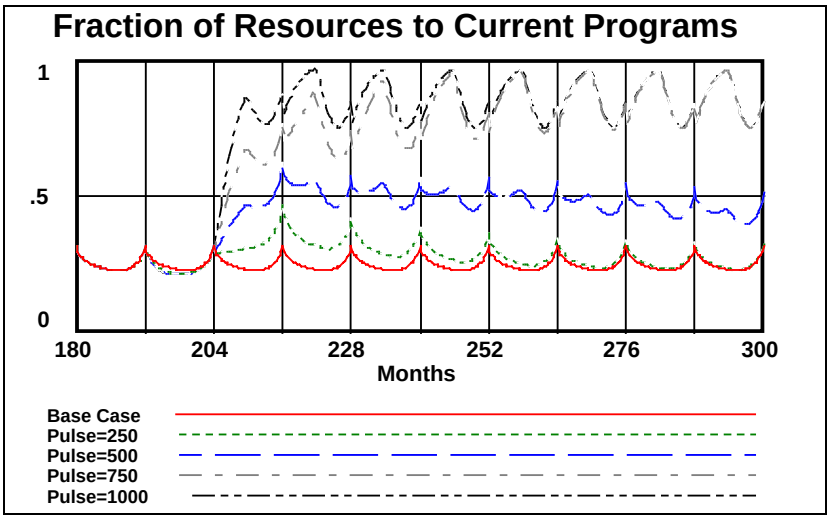

Figure 5

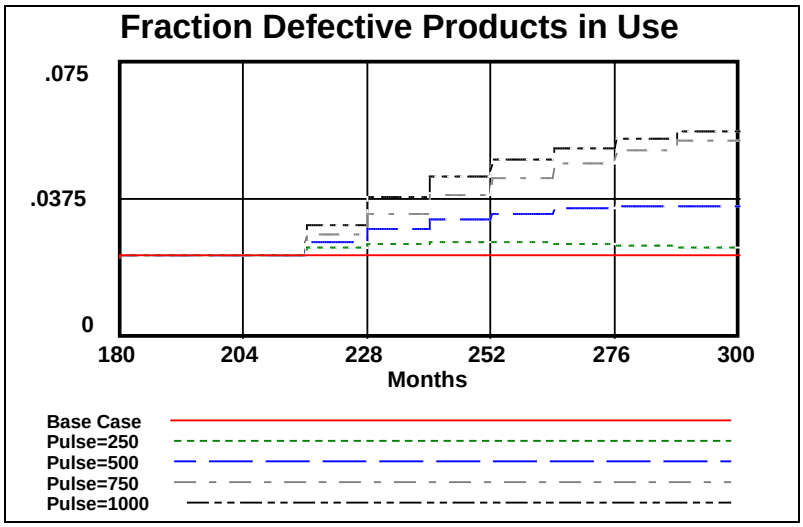

Figure 6

In the base case, approximately 25% of the total resources are allocated to the current phase. A one time increase in the work rate of 250 changes causes the fraction of current work to immediately grow and then slowly return to the base case behavior. The results are quite different, however, for larger pulses. When the pulse size is 1000 units the fraction of capacity dedicated to current work immediately jumps to near 100% and does not return to the pre-pulse behavior. Similarly, if the pulse size is 750 units, the fraction of current work migrates towards 100%, albeit more slowly than with the larger pulse. For large changes, the system shows a significant amount of irreversibility: once it reaches the point where a large fraction of work is being done in the current phase, it never returns to its pre-pulse behavior. Once resources are 'tilted' towards current work, the initial change is self-reinforcing and the system does not recover to its pre-pulse behavior.

Figures 9 and 10 help explain why this is the case.

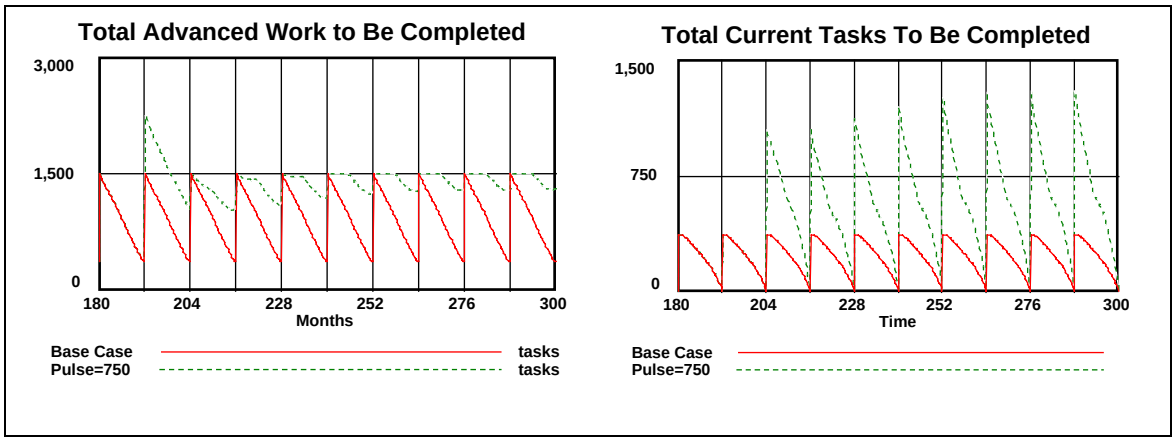

Figure 7

After the pulse there are now 2000 changes to be completed rather than 1500 in the advanced phase. In the first year following the pulse, approximately the same number of new advanced tasks are completed. However, at the model year transition, 500 additional tasks move to the current stage uncompleted. In the second year, even fewer tasks are completed and more are moved to the current phase unfinished. Over the succeeding years progressively less advanced work is done, and by month 300 the vast majority of advanced work remains uncompleted for the entire year.

In the current phase there is no change after the first year. In the second year, however, 500 more tasks are received uncompleted (the 500 that did not get done in advanced work in the previous year). In subsequent years, as the advanced work completion rate declines, more and more work enters the current stage uncompleted, amplifying , rather than attenuating, the effect of the initial pulse.

Figure 10 shows the source of the amplification. As the fraction of work done during the current phase grows, the amount of new work done in the current phase increases. More work done in the current phase leads to more defects introduced and creates more current

re-work. In the first year after the pulse, the fraction of current tasks defective increases, and more resources are needed for the current phase due to increased re-work. The stock of current rework grows substantially due to these extra changes. In the following year it is smaller, but grows over time as more defects are introduced due to the higher fraction of new work done in the current phase.

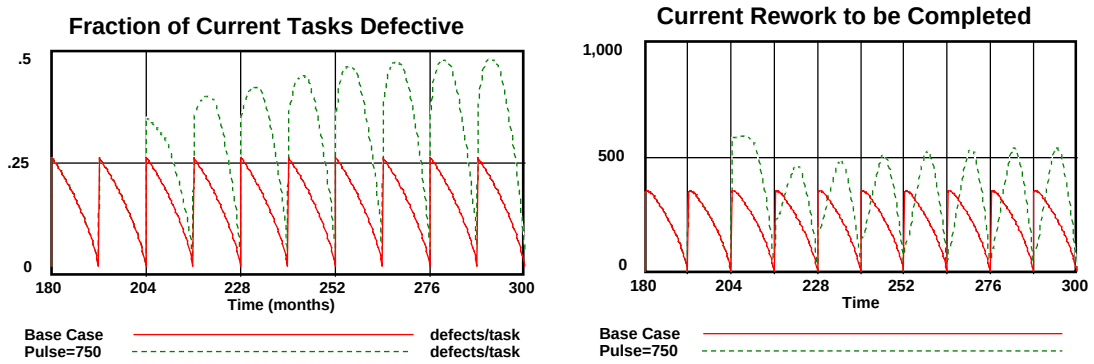

Figure 8

The growth in current rework increases the resources dedicated to the current phase further starving advanced programs and creating more re-work in subsequent years. Figure 10 shows the fraction of all current tasks that are defective. In the base case, this fraction declines monotonically as the re-work loop works to improve quality. In the pulse case, this fraction grows in the beginning of the year as new defects are introduced through the completion of new current tasks. Once the current work is completed then the re-work loop reduces the defect level dramatically. However, although it is not totally apparent from the figure, the defective fraction does not reach the same value that it does in the base case. The end result, as shown in figure 8, is that a greater number of defects are introduced into the market.

The pulse test shows an important property of the system. Although the pulse occurs only once, if it is large enough, it can push the system into a new steady state behavior in which the ability to produce products without defects is permanently reduced. This key feature is due to resource dependence and tilting. Without resource dependence between the advanced and current phases, the impact of the pulse would only be felt until all changes were completed and the product introduced, then the system would be re-set and return to its steady state behavior. However, because additional work done in the current phase reduces the resources allocated to the advanced phase, the system does not re-set itself. Instead the pulse has a permanent effect on system behavior. To use an analogy from stochastic processes, the high re-work, high stress, crisis mode of development represented in the model by a high fraction of work dedicated to the current project is an absorbing state. Once the system enters this regime, it does not leave unless some additional intervention is made.

3. Introducing Process Improvement

The model provides one explanation for why organizations often operate in a high stress, high rework mode. A common response to such a mode of operation is to introduce some modifications to the engineering and design process. Such changes include new tools to reduce the probability of introducing a defect and speed the re-work cycle, and specification of a modified sequence of development steps including appropriate checkpoints and reviews. In this section, additional structure is added to the model to capture the introduction of new tools and processes. Introducing a new tool or process is assumed to have two impacts on the system. First, the new technology reduces the probability of introducing a defect in all types of work. However, the full benefit of the

tool is not realized until the organization has developed experience using it; there is a delay between beginning to use a new tool and receiving its full benefit. Second, using the new technology makes completing each development task take *longer*. The increase in task duration is designed to capture the fact that new tools and processes generally require time to be learned and have additional documentation and reporting requirements. The model is parameterized in such a manner that the reduction in the probability of introducing a defect more than compensates for the increase in task duration. Thus, if the firm can gain the necessary experience, adopting the new methodology increases the total capability of the development system.

Model Structure

The introduction of a process improvement initiative is captured in a simple way. First, there is assumed to be a threshold allocation of resources to the advanced phase required to achieve the full benefit of the new tools. For the purpose of this model it is assumed to be 75% (the average value in the base case). The organization's current adherence to the methodology (CAM) is then calculated by dividing the current fraction of resources (FRA) dedicated to the advanced phase by the threshold value (FRA*). Thus:

$$CAM=MIN(1,FRA/FRA^*)$$

The rate of defect introduction is determined by the organization's average adherence to the methodology (AAM). AAM is a weighted average of the current adherence. The average number of changes completed over the three previous model years is used as the weighting factor. Thus, AAM equals:

$$AAM(t) = \frac{\int_{t_0}^{t} (CIR \cdot CAM - CRR \cdot AAM) ds}{\int_{t_0}^{t} (CIR - CRR) ds}$$

where CIR is the change introduction rate and CRR is the change retirement rate. CRR is simply assumed to be a first order exponential delay of CIR with a time constant, τ, of three years. Thus CRR equals:

$$CRR(t) = \frac{\int_{t_0}^{t} (CIR - CRR) ds}{\tau}$$

The impact of the new methodology on capacity is captured by changing the equation for normal development capacity (NDC). Once the new technology is introduced, NDC equals:

NDC=AEH/DHT

where DHT represents the time required to complete a task while following the methodology.

Model Behavior

The new development methodology is introduced in the model in month 192. Figure 10 shows the consequence of introducing a new methodology under three different assumptions about the necessary change in normal development capacity. In the first, the methodology requires no more time than the current operating procedure and thus normal capacity is not reduced -- NHT=DHT. In the second the methodology increases the required time per task from 300 to 350 hours, and in the third, the time required per task is increased to 400 hours.

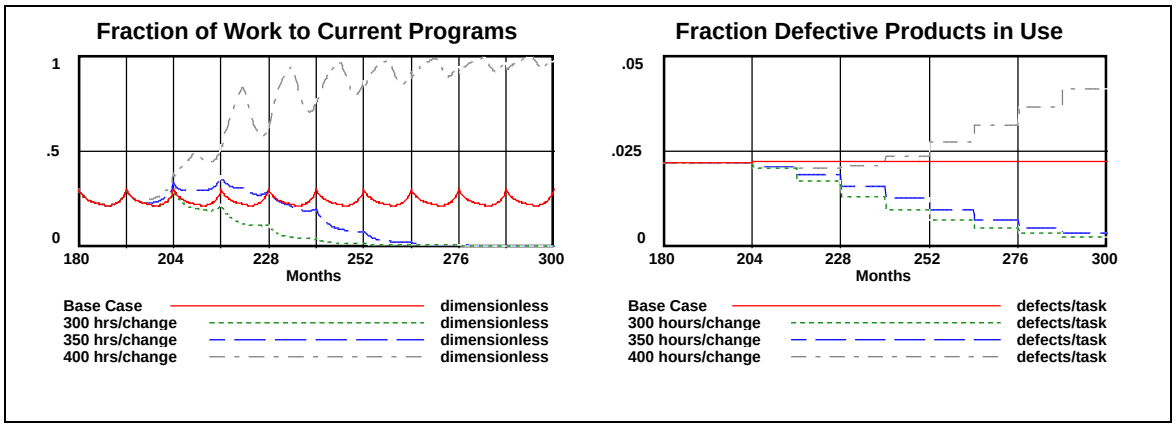

Figure 9

If the methodology entails no sacrifice in capacity (case #1), then the fraction of work devoted to current programs decreases immediately after the methodology is introduced. As more work is shifted to the advanced phase, fewer defects are introduced and less re-work is required enabling an even larger fraction of work to be completed in the advanced phase. If the cycle works in this, virtuous, direction, the fraction of products introduced with defects decline significantly. In case #2, the decrease in development capacity caused by the new methodology initially causes the fraction of current work to increase. The change is, however, only temporary. After three model years, the fraction begins to decline and the new methodology again leads to a substantial reduction in defects introduced.

Case #3 shows different behavior. Whereas in cases #1 and #2 the introduction of the new methodology leads to an improvement in system performance, in case #3 it leads to a substantial reduction, even though the model is parameterized so that the new methodology, if fully adopted, leads to an improvement in capability. Figure 11 shows that in case #3, the system evolves to the point where the vast majority of the work is done in the current phase.

Figure 12 helps explain the different outcomes.

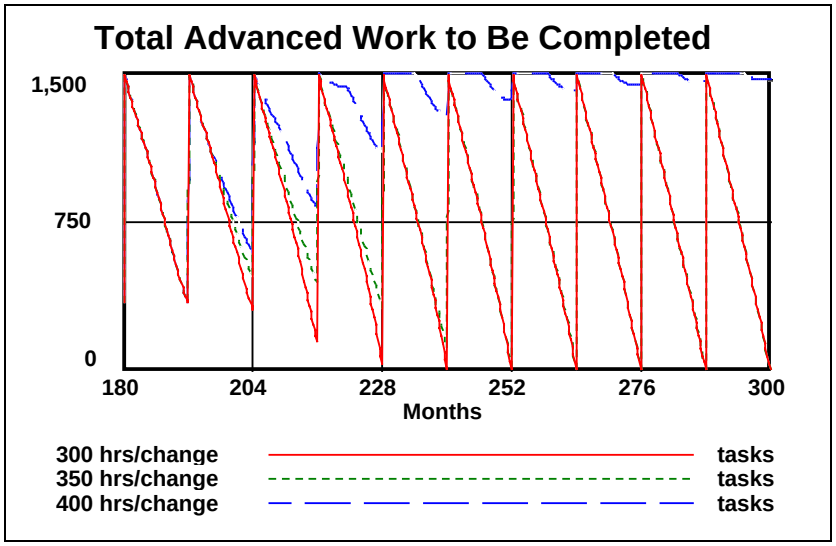

Figure 10

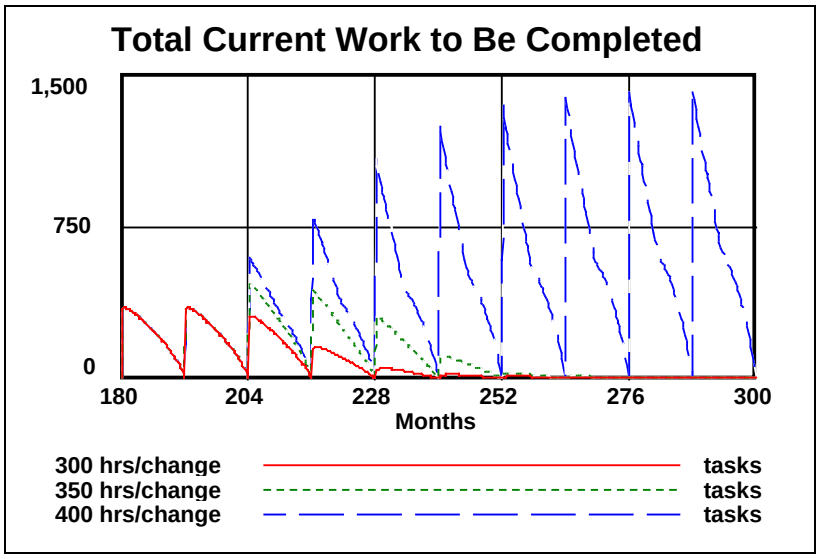

Figure 11

In both cases #2 and #3, fewer advanced tasks are completed in the years following the introduction of the new methodology. As a consequence, more work is moved to the current phase uncompleted. As in the earlier cases, more work in the current phase leads to a higher defect introduction rate (figure 9). In case two, this change is temporary. As

the positive benefits of the new methodology kick in, the defect introduction rate falls, fewer resources are needed for the current phase and the advanced work completion rate begins to rise. As the advanced work rate rises, adherence to the methodology increases and system performance improves as the positive feedback loop works in the desired direction.

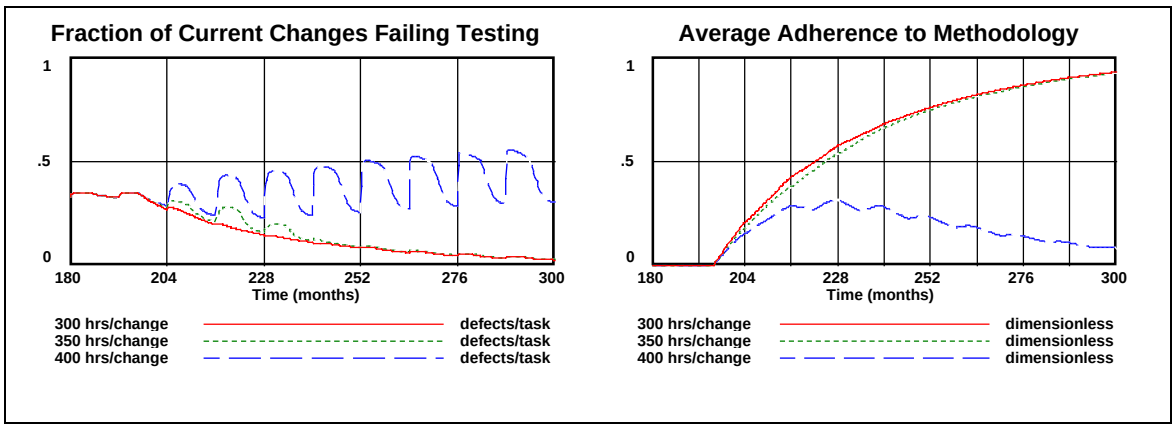

Figure 12

In contrast, in case #3 the advanced work completion rate does not recover. Instead, as more work is shifted to the current phase, the defect introduction fraction rises creating even more current work. In this case, the initial transient is large enough that the system never recovers. The decline in performance is reinforced as an increasing fraction of resources are allocated to current programs and adherence to the methodology declines.

Unlike cases #1 and #2, the system does not make the transition to the desired state. Instead the behavior is very similar to that of the pulse test described above. In case #3, the introduction of the new methodology actually makes performance *worse* rather than better. The introduction of a new methodology, when it entails a large decrease in initial capacity, creates behavior very similar to the pulse test discussed above. The transient behavior leads to a new steady-state in which the system performance is clearly degraded.

Rarely do those who promote new methodologies worry that their introduction may make the system worse rather than better. In an environment with resource dependence, however, this is a possibility.

Limiting Strategies

The simulation experiments show how introducing a beneficial new tool or process in an environment with resource dependence can worsen the situation trying to be improved. In this section a policy for mitigating these effects is proposed. A simplified feedback representation of the feedback structure of the model is show in figure 15. There are two main loops that drive the dynamics. The balancing loop, labeled B, represents the organization's desire to launch a product with few defects. There is a goal for the defect level which is compared to the current state. If the comparison generates a positive gap (meaning more defects than desired), additional effort is allocated to the current product. As the resource level dedicated to the current product is increased, the defect level is reduced and the gap is closed.

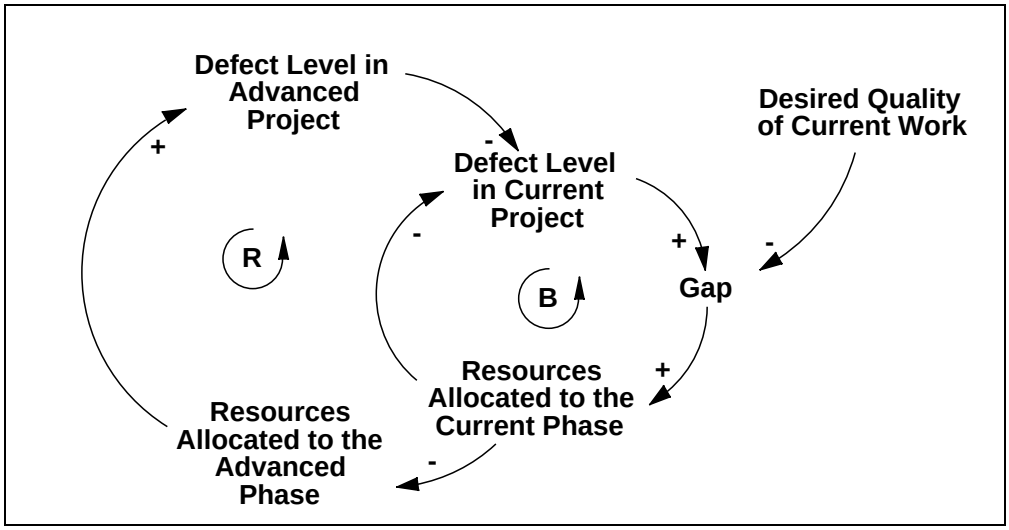

Figure 13 Arrows indicate the direction of causality. Signs ('+' or '-') at arrow heads indicate the polarity of relationships: a '+' denotes that an increase

in the independent variable causes the dependent variable to increase, ceteris paribus (and a decrease causes a decrease). That is, $X \rightarrow^{+} Y \Leftrightarrow \partial Y/\partial X > 0$. Similarly, '-' indicates that an increase in the independent variable causes the dependent variable to decrease; that is, $X \rightarrow^{-} Y \Leftrightarrow \partial Y/\partial X < 0$. See Richardson and Pugh 1981.

Allocating additional effort to fix defects in the current project also reduces the effort allocated to the advanced project. Fewer resources allocated to the advanced phase means that projects will have defects needing correcting when they enter the current phase. These links create the reinforcing loop labeled with the R. This loop makes the resource dependent product development system unstable and prone to tilting. If the loop works in the virtuous direction, the system is driven towards higher capability. If the loop works in the opposite direction, the system is driven to low levels of capability.

The introduction of a new methodology, if it increases the initial time required per task, can unintentionally ignite the downward spiral represented in the loop R. The reduction in capacity caused by the increase in hours per task reduces the amount of work accomplished in the advanced phase. If less work is accomplished in the advanced phase, more must be done in the current phase to achieve the desired quality level further starving the advanced product and creating the downward spiral of capability. The poor behavior is caused by the myopic decision rule used to determine the allocation of resources between the current and advanced phases. Favoring the project nearest to launch produces undesirable behavior when resources are scarce.

To complement the introduction of the new methodology, a new implementation strategy is added. An organization has a number of ways to compensate for its low process capability including fixing defects as they occur and investing in long run improvements

that eliminate the defects in the first place. Frequently both paths cannot be pursued simultaneously and the organization must allocate scarce resources between fixing current problems and preventing future ones. Repenning and Sterman (1997) discuss in some detail why there is a strong bias in many organizations towards correction. To counteract this bias, Repenning (*in progress*) introduces the concept of *limiting strategies*. Limiting strategies are based on a simple idea: effective implementation of new improvement methods focused on prevention requires progressively eliminating the organization's ability to use the short-cut correction methods. The archetypal example of such a strategy is the use of inventory reduction to drive fundamental improvements in machine up-time and yield in manufacturing operations. As the amount of material released to the production system is reduced, operators and supervisors are less able to compensate for low yield and up-time by running machines longer and holding protective buffer inventories. In such a situation the only way to meet production objectives is through fundamental improvements.

The first step in using a limiting strategy is to identify the key mechanisms through which the organization compensates for its low underlying capability. In product development systems one way to compensate for low capability is to redo existing work to correct defects. In the model, this is represented by multiple iterations through the re-work cycle prior to product launch. To use a limiting strategy to implement the new methodology, the allocation of resources to the current phase re-work cycle must be constrained. In the model this is operationalized in a very simple way. At some point during the current phase, a design freeze is introduced and, from that date forward, no resources are allocated to re-work.

In the simulations shown below (figure 16 through 18), both the methodology and the freeze date policy are introduced in month 192. Figure 16 shows the fraction of work dedicated to current programs for a variety of freeze date policies. The 'no-freeze'(freeze at 0) is identical to the adherence only policy discussed earlier. The system never makes the required transition to a higher percentage of advanced work (figure 16), and never develops enough experience with the methodology to reap its benefits (figure 17). The system's performance is significantly worse than if the methodology had not been introduced (figure 18).

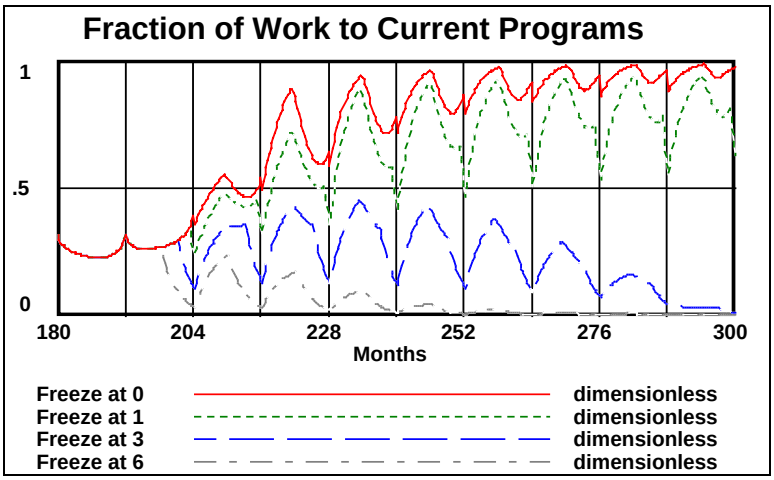

Figure 14

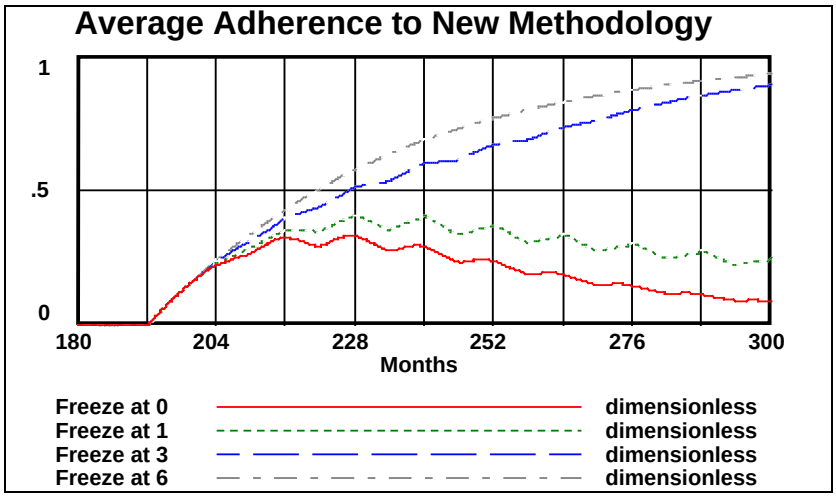

Figure 15

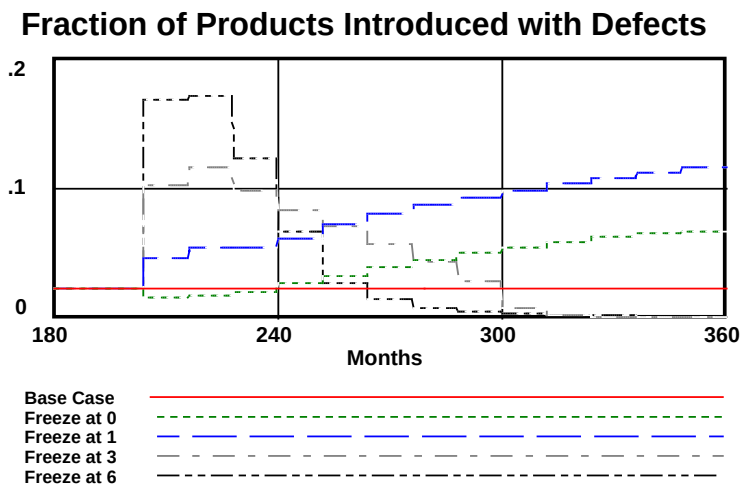

Figure 16

The results are even worse if the freeze date is one month prior to launch. The system fails to make the transition to an increased fraction of work done up-front (figure 16), adherence to the new methodology never reaches above 50% (figure 17) and an even higher fraction of defective products are introduced to the market (figure 18).

In contrast to the no-freeze and one month freeze policies, freezing at three or six produces a different behavior pattern. In both cases the system makes the transition to the regime in which the vast majority of work is done in the advanced phase (figure 16). With a six month freeze date this happens quickly, while with a three month date there is an extended transition period in which the fraction of resource dedicated to the current project is increased. As a consequence, in both cases the adherence to the methodology increases monotonically (figure 17). The limiting strategy is effective in insuring the methodology is implemented.

However, the long run success does come with a cost. The freeze policy creates a substantial 'worse-before-better' dynamic. In both the three and six month freeze cases the fraction of defective products introduced to the market increase substantially after the policy is put in place (figure 18). The three month date produces a smaller initial increase but also a slower transition to the higher capability system. The six month date creates a substantial increase in the defect introduction rate, but leads to a faster transition to the new process.

4. Discussion:

a. Dynamic Complexity

Developing new products can be a very complex process. The product being created may have thousands of components, be composed of new and unproven technologies, and require significant advances in process capability. The trend towards product development processes that include participants from areas outside engineering only increases the level of complexity: a well designed development effort cuts across many of the traditional functional boundaries within an organization. Dozens of different departments including sales, marketing, product design, product engineering, manufacturing engineering, purchasing, and field service and support may be included, and there can be hundreds, if not thousands of different people whose input is needed to successfully develop and introduce a new product.

Decomposition is a popular problem solving strategy is such complex situations (Simon 1969), so, given this immense level of detail complexity, it is not surprising that organizations often treat development projects as independent entities. The rise of the matrix organization within many product design and engineering departments can be seen

as an attempt to manage the complexity of the process by decomposing the process by both function and product. However, while an increased focus on projects may help managers cope with the detail complexity, it has also greatly increased the dynamic complexity. An organizational structure composed of both projects and functional areas has high degree of resource dependence. Projects must compete for the scarce attention of all of the different functional departments.

Many authors have recognized that managers who treat development projects as independent entities do so at their peril. One of the major thrusts of Wheelwright and Clark (1992, 1995) is that senior leaders should focus on managing the portfolio of development efforts rather than individual projects. It has become an article faith among business writers that managers should focus on whole systems and processes not functions and specific pieces (Garvin 1995, Senge 1990, de Geuss 1988, Stata 1989). This shift in perspective is obviously important, but the literature contains less guidance on how to put such thoughts into action. Unfortunately, more is needed than simply recognizing that product development processes are complex systems with interdependent elements. A substantial body of research shows that human decision makers perform poorly in decision making tasks in dynamic environments (Sterman 1989a, 1989b; Brehmer 1992; Funke 1991). In experiments with even modest levels of dynamic complexity, subjects are often outperformed by simple decision rules that account for a fraction of the available information (Paich and Sterman 1993, Diehl and Sterman 1995).

Decision making processes are the major focus of much of the literature on managing product development (Wheelwright and Clark 1992). An alternative approach to

improvement is to focus not on the decisions, but on the structure of the process itself. Numerous authors have argued that senior managers have the most impact on a given project very early in its life-cycle (e.g. Gluck and Foster 1975). Similarly, the high leverage point in improving product development processes may lie in improving the structure of the process rather than the day-to-day decisions made within that structure.

The analysis presented here focuses on one important decision, the allocation of resources between current and advanced projects. There is a growing body of case study evidence and theory that suggests that people behave myopically in these contexts, specifically over-investing in current project and ignoring longer term investments (Repenning and Sterman 1997, Repenning 1996, Jones 1997). Combining resource dependence and such myopic behavior creates a system with a number of undesirable characteristics: the system is not robust to variations in work load, and new tools and processes are difficult to implement. A limiting strategy, which is operationalized as fixing resource allocation rather than leaving it to the discretion of the manager, is a simple, easily implementable solution to this problem that greatly increases the robustness of the process to variations in workload, and improves the effectiveness of new tools that are introduced.

b. Future Research: Designing Robust Organizations

Placing constraints on the allocation of resources between projects is only one small example of how processes can be redesigned to be more robust and produce consistently better results. The methodology of robust design has been widely propagated through the product development community with great success in many areas. Substantial improvements have been made in the quality and reliability of many projects through

designs that minimize the sensitivity of the final item's performance to variations in the inputs.

The core idea of robust design is equally applicable to the design of processes and organizations. In most cases it is prohibitively costly to run the needed experiments in actual organizations and, unlike many product development processes, prototyping is not an option. Thus the development of models to capture the dynamics of such processes is critical to understanding which policies are robust to changes in the environment and which are not. The simulation methodology used here provides one means to generate useful models of organizations and processes focused on improving the design of organizations (Forrester 1961). Similar methodologies has been used successfully to improve the performance of a wide range of business systems including preventive maintenance programs (Carroll, Markus and Sterman 1997), process improvement programs in manufacturing, (Sterman, Repenning and Kofman 1997, Repenning 1997), and supply chain management. Much work, however, remains. Developing models to aid in the understanding and improvement of product development processes, using a wide range of methods, remains a fruitful and productive area for research.

References

Brehmer, B., (1992). Dynamic Decision Making: Human Control of Complex Systems, *Acta Psychologica*, 81,211-241.

Carroll, J., J. Sterman, and A. Markus (1997). Playing the Maintenance Game: How Mental Models Drive Organization Decisions. R. Stern and J. Halpern (eds.) *Debating Rationality: Nonrational Elements of Organizational Decision Making*. Ithaca, NY, ILR Press.

Clark, K. and T. Fujimoto (1991). *Product Development Performance: Strategy, Organizations, and Management,*

Dean, J. W. and D. Bowen (1994). "Management Theory and Total Quality: Improving Research and Practice Through Theory Development, *Academy of Management Review*, 19(3): 392-418.

de Geus, Arie (1988). "Planning as Learning," Harvard Business Review, March-April, 70-74.

Diehl, E. and J.D. Sterman (1995). Effects of Feedback Complexity on Dynamic Decision Making, *Organizational Behavior and Human Decision Processes,* 62(2):198-215.

Dertouzos, M., R. Lester, and R. Solow (1989). *Made in America: Regaining the Productive Edge,* The MIT Press, Cambridge, MA.

Forrester, J. W. (1961). *Industrial Dynamics.* Cambridge, MA: The MIT Press.

Forrester, J. W. (1991). "From the Ranch to System Dynamics: An Autobiography", in Bedeian, Arthur (ed) *Management Laureates: A Collection of Autobiographical Essays,* JAI Press.

Funke, J. (1991). Solving Complex Problems: Exploration and Control of Complex Systems, in R. Sternberg and P. Frensch (eds.), *Complex Problem Solving: Principles and Mechanisms.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Gluck, F. and R. Foster (1975). "Managing Technological Changes: A Box of Cigars for Brad", *Harvard Business Review,* Sept.-Oct., 139-150.

Hauser, J. and D. Clausing(1988). "The House of Quality", *Harvard Business Review*, Vol.66, No. 3:66-73.

Huber, G.P. and W.H. Glick (1993), *Organizational Change and Redesign: Ideas and Insights for Improving Performance.* New York, Oxford University Press.

Jones, A.P. (1997). Sustaining Process Improvement in Product Development: The Dynamics of Part Print Mismatches, unpublished MS Thesis, Department of Civil and Environmental Engineering, Massachusetts Institute of Technology, Cambridge, Ma.

Kanter, R.M., T.D. Jick, and R. A. Stein (1992). *The Challenge of Organizational Change*. New York, Free Press

Krahmer, E. & R. Oliva (1996). Improving Product Development Interval at AT&T Merrimack Valley Works. Case history available from authors, MIT Sloan School of Management, Cambridge, MA 02142.

Paich, M. and Sterman, J. (1993). Boom, Bust, and Failures to Learn in Experimental Markets. *Management Science*, 39(12), 1439-1458.

Repenning, N. (*in progress*). Improvisational Change, Process Improvement, and Limiting Strategies. Working Paper Available from author, Sloan School of Management, MIT, Cambridge, MA 02142.

Repenning, N. (1997b). Successful change sometimes ends with results: Exploring the Improvement Paradox. Working Paper available from author, MIT Sloan School of Management, Cambridge MA 02142.

Repenning N. and J. Sterman (1997). Getting Quality the Old-Fashion Way: Self-Confirming Attributions in the Dynamics of Process Improvement, to appear in *Improving Research in Total Quality Management*, National Research Council Volume, Richard Scott and Robert Cole, eds.

Repenning, N. (1996). Reducing Product Development Time at Ford Electronics, Case Study available from author, MIT Sloan School of Management, Cambridge MA 02142.

Richardson, G. P. and Alexander Pugh (1981). *Introduction to System Dynamics Modeling with DYNAMO*, MIT Press, Cambridge, Ma:

Senge, P. (1990). *The Fifth Discipline: The Art and Practice of the Learning Organizations*, Doubleday, New York, NY.

Simon, H.A. (1962) The Architecture of Complexity. *Proceedings of the American Philosophical Society*, 106, No.6, 467-82.

Simon, H.A.(1969). The *Sciences of the Artificial.* Cambridge, MA: The MIT Press.

Stata, R. (1989). "Organizational Learning: The Key to Management Innovation," *Sloan Management Review*, 30(3), Spring, 63-74.

Sterman, J. D. (1989a). Misperceptions of Feedback in Dynamic Decision Making. *Organizational Behavior and Human Decision Processes* 43 (3): 301-335.

Sterman, J. D. (1989b). Modeling Managerial Behavior: Misperceptions of Feedback in a Dynamic Decision Making Experiment. *Management Science* 35 (3): 321-339.

Sterman, J., N. Repenning, and F. Kofman (1997). Unanticipated Side Effects of Successful Quality Programs: Exploring a Paradox of Organizational Improvement. *Management Science*, April, 503-521.

Van de Ven, A., and M.S. Poole (1995). Explaining Development and Change and Organizations, *Academy of Management Review*, Vol. 20, No. 3, 510-540

Weick, K. E. (1993). "Organizational Redesign as Improvisation" in Huber, G.P. and W.H. Glick (eds.) *Organizational Change and Redesign*., New York, Oxford University Press.

Weick, K.E. (1979). *The Social Psychology of Organizing*, Second Edition, New York, Random House.

Wheelwright, S. and K. Clark (1993). *Revolutionizing Product Development: Quantum Leaps in Speed Efficiency and Quality*, The Free Press, New York, NY.

Wheelwright, S. and K. Clark (1995). *Leading Product Development,* The Free Press, New York, NY.

Ulrich, K. and S. Eppinger (1995). *Product Design and Development*, McGraw-Hill Inc., New York, NY.

Zangwill, W. (1993). *Lightning Strategies for Innovation: how the world's best create new projects*, Lexington Books, New Your, NY.