"LIMITS TO CONCURRENCY"

by

G.M. HOEDEMAKER*
J.D. BLACKBURN**
and
L.N. VAN WASSENHOVE†

95/27/TM

* Research Assistant at INSEAD, Boulevard de Constance, 77305 Fontainebleau Cedex, France.

** Owen Graduate School of Management, Vanderbilt University, Nashville, TN 37203 USA.

† Professor of Operations Management and Operations Research at INSEAD, Boulevard de Constance, 77305 Fontainebleau Cedex, France.

A working paper in the INSEAD Working Paper Series is intended as a means whereby a faculty researcher's thoughts and findings may be communicated to interested readers. The paper should be considered preliminary in nature and may require revision.

Printed at INSEAD, Fontainebleau, France
Limits to Concurrency

by

Geert M. Hoedemaker
INSEAD
Fontainebleau, France

Joseph D. Blackburn
Owen Graduate School of Management
Vanderbilt University
Nashville, TN 37203 USA

Luk N. Van Wassenhove
INSEAD
Fontainebleau, France

January, 1995
Limits to Concurrency

Abstract

In the global race to bring new products to market, many firms have adopted concurrent engineering as a technique to shrink development lead time. Due to the many concurrent engineering success stories in the business and engineering literature, a common misconception has grown that more concurrency is always better. The major contribution of this paper is a rigorous mathematical proof that limits to concurrency exist even in the simplified situation in which concurrency is modeled as the number of design modules to be executed in parallel. As complexities such as communication linkages between modules are layered onto our basic model, we show that the expected project completion time is minimized at a finite number of modules that decreases with added complexity. In general, the more complex the project, the stricter the limits to concurrency. This strongly suggests that project managers should be cognizant of the potentially adverse effects of pushing concurrency too far.
1. **Introduction**

Concurrent Engineering has become a guiding principle for reducing the time-to-market for new products (Rosenblatt, 1991). With concurrent engineering all activities required to bring a product to market—marketing, design, engineering and manufacturing— are jointly managed to work in parallel, in sharp contrast with the traditional "over the wall" approach. The concurrent process has been characterized as the "rugby team" approach to design instead of the sequential "relay race" approach (Takeuchi and Nonaka, 1986). Many firms have wholeheartedly embraced the concept, although there is no evidence that it fits every environment or that more concurrency is always better (Cordero, 1991).

As with other popular management concepts, the repetition of anecdotal successes can transform a useful principle such as concurrency into a "gospel" for managers: one is either a believer or not. Aggressive managers and consultants push such concepts hard and sometimes oversimplify them in the process. Misconceptions grow that more is always better. However, taking useful management concepts to the extreme has proven before to be counterproductive. For example, JIT excesses have created rigid systems that are excessively vulnerable to changes in external conditions or process variability, and similar excesses have accompanied the adoption of TQM, CIM and MRP systems. As a result, many managers became disenchanted with what was potentially a useful and powerful concept.

The same thing could happen to concurrent engineering—another useful and powerful concept—if there are limits to its effectiveness. If limits exist, managers should be made aware of them and empirical research as to the precise nature of the limits should be carried out. Some researchers, such as Brooks (1975), have suggested potential dangers of extensive segmenting of design activity; however, these limits have neither been spelled out in the general context of concurrent engineering nor proved in a rigorous mathematical fashion.
Questions about the limits of concurrency surfaced to us during empirical studies of the software development process (Blackburn, et. al, 1992). Alcatel, a leading manufacturer of telecommunications systems, described the following problem:

The coding process for large programs to manage telecommunications switching systems software is attacked by subdividing the program into modules to be coded concurrently by teams of software engineers. As the module size is decreased, the degree of parallel activity in coding clearly increases. At the same time, however, inefficiencies arise because of problems created at the interfaces between modules. For instance, more communication is required between engineers working simultaneously on modules. In addition, the potential for interface errors also increases and with it the likelihood of integration problems when the modules are assembled together and tested as a workable system. This strongly suggests that there exist limits for both the size and the number of modules in a software project.

Pushed by time pressure, development managers strive to overlap activities in order to compress the design cycle time. But at what cost? Alcatel had begun to recognize that the size and number of modules designed in parallel was a critical factor in determining project duration. In software, coding efficiency tends to increase if modules are defined small enough to be coded by a programmer working alone, but are these gains illusory? Is there a tradeoff posed by the increasing communication burden among programmers working on related modules? Is time gained by increasing modularization lost due to integration and testing failures? If increasing the number of modules increases the likelihood of rework, then the coding productivity gains may evaporate.

In their struggle with this issue, management at Alcatel was raising fundamental questions about the efficacy of concurrent engineering. For instance, if there are restrictions on the degree to which single-stage design activities can be overlapped, then how formidable are the obstacles to concurrency across stages? The advocates of concurrent engineering are strangely silent on this question.

Subsequently, we have learned from interviews that similar limits to concurrency exist in the development of other new products, not just software. In the design of integrated circuits, for example, subdividing the work into modules smaller than individual components can be
impractical and counterproductive. When the design modules become too small a piece of the puzzle, individual design engineers cannot see the full effect of constraints on their design decisions; infeasibilities only become apparent when the design "pieces" are assembled into a system. Additional constraints on the degree of effective concurrency are imposed when the design of components is subcontracted to suppliers. These complexities thwart the basic objective of concurrency by imposing sharp time penalties on the overall project completion time.

There is a limited literature on model-based studies of concurrent development processes. The engineering design literature contains numerous papers on data requirements and models of the CAD-design interface (O'Grady and Young, 1991). Clark and Fujimoto (1989) model design as a set of overlapping problem-solving activities, and Blackburn (1991) proposes that this model can be implemented by applying Just-in-Time (JIT) concepts to the process. Decomposition models of the concurrent engineering process have been proposed by Kusiak and Park (1990) and Eppinger, Whitney and Gebala (1992). Gebala and Eppinger (1991), Kowal and Shtub (1991) and Ait-Sahlia, Will and Johnson (1991) model the impact of concurrent engineering on design lead time.

Although the concept and successes of concurrent engineering are well documented in the literature, research on the subject is made difficult by the lack of a commonly-accepted definition. Confusion about the process of concurrency occurs because, in development, concurrency actually takes two forms: concurrency in *time* and concurrency in *information* (Blackburn, et. al, 1994). Concurrency in time refers to activities that are performed in parallel by different people or groups, such as simultaneous design activities. Information concurrency refers to the integrated, or team, approach in which all the concerns of the different functions-- the customer, R&D, design engineering, manufacturing, and sales-- are addressed through shared information.

Our model is based on the Alcatel case. It deals with time concurrency, or more precisely with concurrent development of modules, and ignores issues of information concurrency. For simplicity, we ignore complex dependencies between communication, rework and integration.
Instead, we make increased communication, rework probability and integration test problems a simple function of the increasing number of concurrent modules. Therefore, for the purpose of our models, concurrency is reduced to the number of modules designed in parallel and related factors.

We consider a single design stage of a multistage development project. In computer software, the stage could be the coding of a program. As such, the problem can either be solved as a whole, or split into smaller problems (sets)—called modules—to be tackled by distinct teams. Under the modular approach, a decision must be made concerning the size of the modules or the number of modules to be developed in parallel. The number of modules may depend on the number of functions the system is to perform, the logical subsystems into which it can be divided, and on available personnel.

Our analysis seeks to determine the number of modules that will minimize expected overall completion time. To compute the minimal expected completion time, we must specify how different factors influence development lead time. Figure 1 depicts the model of the development process and the three factors assumed to influence that process: (1) communication; (2) rework; (3) integration test time. Communication needs among the design team obviously affect development time. Brooks (1975) has stated that communication in a development project is related to the number of communication links between the project teams. The more modules, therefore, the more time that will be spent on communication among the teams, e.g., to discuss interfaces or design problems. Rework is a second factor influencing overall completion time due to the risk that integration will not be successful. Each module is part of the larger system which must eventually integrate all the modules. Design flaws can occur because of incomplete information transfer when using simultaneous design. The more modules, the higher the likelihood that the modules will not integrate smoothly and must be reworked. Finally, because complexity tends to increase with the number of modules, the system or integration test will take more time.
This simplified model allows us to examine the conjectures of Brooks and the Alcatel development manager by providing the first simple and rigorous proof of limits to concurrency. We show that as additional layers of complexity are added to the problem, the adverse effects of additional concurrency increase—not only is there a finite optimum to the number of concurrent modules but that number decreases with increasing complexity. The latter result also increases our confidence in the robustness and validity of our simple process model. Adding "realism" to the model by introducing more complex links relating rework probability and integration test difficulties to increased communication requirements accompanying increased overlapping would only diffuse the development and would yield the same basic result: there are indeed limits to concurrency.

2. Models of Concurrency

We consider a sequence of models of concurrency starting with a base model and adding complexity with subsequent models. In each case we compute the expected completion time as a function of the number of modules and show that, except for the simplest model, there exists an optimal number of modules that minimizes the expected completion time. Increasing the degree of concurrency beyond that optimal number increases the expected completion time. That is, limits to concurrency exist for all but the most elementary model. We also show that as additional layers of complexity are added, the optimal number of modules decreases.

Section 2.1 introduces the base model that serves as the foundation of this study. We derive the Expected Completion Time, E[T(n)], as a function of the number of modules, n. In Section 2.2 communication time is introduced as a factor that counteracts the advantageous effects of dividing a system into simultaneously-developed modules. Section 2.3 introduces rework as the result of integration problems. It is assumed that if defects are detected, which cause the system to fail the integration test, a fraction of every module must be reworked. Iterations continue until the system passes the integration test. First, this test itself is assumed to be
instantaneous. Subsequently, this assumption is relaxed and integration test time is introduced as an increasing function of the number of modules.

We will assume task times to be exponentially distributed although there is no empirical evidence to support or refute the exponential distribution assumption. The assumption is made for mathematical convenience: it is not essential to arrive at the result but allows for clear and simple derivations. A simple analogy would be the use of the same distribution to show fundamental queueing effects although very few real queueing distributions completely satisfy the exponential assumptions.

2.1 Analysis of the Base Case

Let $X$ denote the time to complete a fully divisible task, in which completion time is exponentially distributed with a mean equal to one. By splitting up the task into $n$ parts, or modules, we create $n$ random variables $X_i$, denoting module completion times, which are assumed to be independent and exponentially distributed with mean $1/n$.

$$X_i: \quad X_i \sim \exp(\frac{1}{n}) \quad i = 1, \ldots, n \quad (1)$$

The completion time of the task is determined by the largest realization of the $n$ variables, i.e. the maximum of the $n$ module completion times, whose distribution is given by the $n^{th}$ order statistic, $Y^n$. This random variable can be shown to have the following density function:

$$g(n, y) = n^r \sigma^{-r} (1-e^{-\sigma y})^{n-1} \quad (2)$$

The expected value of $Y^n$ is the Expected Completion Time, $E[T(n)]$, which can be calculated as:

$$E[T(n)] = \int_0^\infty n^2 y e^{-\sigma y} (1-e^{-ny})^{n-1} \, dy \quad (3)$$
The following theorems establish properties of $E[T(n)]$.

**THEOREM 1:**

If $E[Y^n]$ denotes the expected value of the maximum of $n$ exponentially distributed random variables with mean $1/\mu$, then

$$E[Y^n] = \frac{1}{\mu} \sum_{m=1}^{n} \frac{1}{m}$$  \hspace{1cm} (4)

Proof: The expected value, $E[Y^n]$, can be computed as

$$E[Y^n] = \int_0^\infty n \mu y_n e^{-\mu y_n} \left(1-e^{-\mu y_n}\right)^{n-1} dy_n$$  \hspace{1cm} (5)

Note that the integrand is Riemann integrable on $[0,\infty)$.

Substituting $z = e^{-\mu y_n}$ in (5) gives

$$E[Y^n] = -\int_0^1 \frac{n \mu \ln(z)}{\mu} (1-z)^{n-1} dz$$  \hspace{1cm} (6)

Let $s = z - 1$, then

$$E[Y^n] = -\int_{-1}^0 \frac{n \mu \ln(1+s)(-s)^{n-1}}{\mu} ds$$  \hspace{1cm} (7)

Note that the domain is transformed into $(-1,0]$. For $s$ in $(-1,1]$, $\ln(1+s)$ can be replaced by the series development

$$\ln(s+1) = \sum_{m=1}^{\infty} \frac{(-1)^{m+1} s^m}{m}$$  \hspace{1cm} (8)

Using this expression in (7) gives, after some simplification,

$$E[Y^n] = \int_0^1 \frac{n \mu \sum_{m=1}^{\infty} s^{m+1}}{m} ds$$  \hspace{1cm} (9)

Interchange of integration and summation is allowed if the terms of the series are Riemann integrable on the domain (through the simplification in (9) transformed to $[0,1]$) and the series is uniformly convergent on the domain. It is clear that the first condition holds; the second condition is verified by observing that the series in (9) can be generated from the standard series

$$S(x) = \sum_{n=1}^{\infty} \frac{x^n}{n}$$  \hspace{1cm} (10)
by multiplying each term by $s^{n-1}$. This standard series is known to converge uniformly for $x$ in $[-1,1)$, and so does this transformation of it, since the domain is preserved.

Interchanging the integral and the summation gives

$$E[Y^*] = \frac{n}{\mu} \sum_{m=1}^{\infty} \int_0^1 s^{m+n-1} \, ds$$

Using integration by parts we obtain

$$E[Y^*] = \frac{n}{\mu} \sum_{m=1}^{\infty} \frac{1}{m(n+m)}$$

This infinite series is calculated by defining the sequence $S_m(n)$ as

$$S_m(n) = \frac{1}{\mu} \sum_{m=1}^{M} \frac{1}{m(n+m)}$$

This can be simplified using fractional decomposition

$$S_m(n) = \frac{1}{\mu} \sum_{m=1}^{M} \frac{1}{m} - \frac{1}{\mu} \sum_{m=1}^{M} \frac{1}{n+m} = \frac{1}{\mu} \sum_{m=1}^{M} \frac{1}{m} - \frac{1}{\mu} \sum_{m=M+1}^{M+n} \frac{1}{m}$$

The last term can be bounded between two sequences that converge to the same limit

$$0 \leq \frac{1}{\mu} \sum_{m=M+1}^{M+n} \frac{1}{m} \leq \frac{1}{\mu} \frac{n}{M+1} = S_m(n)$$

We conclude that

$$E[Y^*] = \lim_{M \to \infty} S_m(n) = \frac{1}{\mu} \sum_{m=1}^{n} \frac{1}{m}$$

In the present case $\mu = n$, so

$$E[T(n)] = \frac{1}{n} \sum_{m=1}^{n} \frac{1}{m}$$

**THEOREM 2:**

$E[T(n)]$ is convex and decreasing on the positive integers.

**PROOF:**

$$\Delta E[T(n)] = E[T(n)] - E[T(n-1)]$$

$$\Delta E[T(n)] = \frac{1}{n(n-1)} - \frac{1}{n(n-1)} \sum_{m=1}^{n} \frac{1}{m}$$
which is negative for \( n \geq 2 \). Furthermore \( \Delta E[T(n)] \) is nondecreasing in \( n \), since \( \Delta^2 E[T(n)] = \Delta E[T(n+1)] - \Delta E[T(n)] \) can be expressed as

\[
\Delta^2 E[T(n)] = \frac{1}{n(n+1)} - \frac{1}{n(n+1)} \sum_{m=1}^{n} \frac{1}{m} - \left( \frac{1}{n(n-1)} - \frac{1}{n(n-1)} \sum_{m=1}^{n} \frac{1}{m} \right)
\]

\[
= \frac{1}{n(n+1)} - \frac{1}{n(n-1)} - \frac{1}{n(n-1)(n+1)} - \left( \frac{1}{n(n+1)} - \frac{1}{n(n-1)} \right) \sum_{m=1}^{n+1} \frac{1}{m}
\]

\[
= -\frac{3}{n(n^2-1)} + \frac{2}{n(n^2-1)} \sum_{m=1}^{n} \frac{1}{m}
\]  

which is nonnegative on the positive integers. Hence, \( E[T(n)] \) is convex in \( n \), and strictly convex for \( n > 2 \), completing the proof.

We conclude that, in the absence of adverse side effects, a decomposition of the task into an infinite number of modules would minimize expected task completion time; that is, the more parallel activity, the better. In this case, then, there is no limit to concurrency (\( n^* = \infty \)). However, the convexity of \( E[T(n)] \) indicates that the gain from further subdivision decreases with \( n \).

### 2.2 Communication Loss

We now introduce the assumption that modules are not performed independently of one another. As \( n \) increases, additional communication among the modules is required and this will increase execution times. The expected completion time, \( E[T_c(n)] \), becomes a function of the time spent performing the actual work plus the time spent on communication. In our model we have chosen to include communication time into the distribution of the completion time of module \( i \). Letting \( C(n) \) denote the communication function, the completion time of module \( X_i \) is defined as having the following distribution:

\[
X_i \sim \exp\left(\frac{1}{n} + C(n)\right)
\]  

Using (4) we can rewrite the expected completion time of the task as

\[
E[T_c(n)] = \left[\frac{1}{n} + C(n)\right] \sum_{m=1}^{n} \frac{1}{m} = E[T(n)] + nC(n)E[T(n)]
\]
Theorem 3 states sufficient conditions that a subdivision into an infinite number of modules is no longer optimal: there exists a finite \( n_C^* \) such that \( E[T_C(n)] \) is minimized for \( n \) equal to \( n_C^* \).

**THEOREM 3:**

If \( C(n) \) is nondecreasing and \( nC(n) \) is convex and increasing, then

1) \( E[T_C(n)] \) is convex;
2) \( E[T_C(n)] \) assumes a minimum at a finite \( n_C^* \).

**PROOF:**

We use the following properties of discrete convex functions:

a) If two functions \( f \) and \( g \) are positive and convex, then \( f \cdot g \) is convex;

b) If functions \( f \) and \( g \) are convex, then \( f+g \) is convex;

c) If \( f \) is convex and increasing, then \( f \) is unbounded.

We have proved in § 2.1 that \( E[T(n)] \) is convex in \( n \). Assume that \( nC(n) \) is convex and increasing, then it follows from a) that \( nC(n)E[T(n)] \) is convex and from b) that \( E[T_C(n)] \) is convex. This proves the first part of the theorem. If \( C(n) \) is nondecreasing and \( nE[T(n)] \) is increasing, we have that \( nC(n)E[T(n)] \) is increasing. From c) it follows that \( nC(n)E[T(n)] \) is unbounded. Since \( E[T(n)] \) is nonnegative, it follows from (21) that \( E[T_C(n)] \geq nC(n)E[T(n)] \), so \( E[T_C(n)] \) is also unbounded. Since \( E[T_C(n)] \) is convex and unbounded, it follows that either \( E[T_C(n)] \) is increasing on the positive integers, i.e. \( n_C^* = 1 \), or \( E[T_C(n)] \) is initially decreasing and assumes a minimum at a finite \( n_C^* \), which concludes the proof.

2.3 Rework and Integration Tests

Communication is rarely faultless, and faulty communication can introduce design flaws which remain undetected until an integration test is conducted. Correction of defects makes necessary rework of the design modules and this can extend the project completion time. Since
rework and integration testing are indirectly related to increasing the number of modules and, consequently, the communication requirements, we examine these effects on project completion time under a set of simplifying assumptions. As with the base model, the assumptions permit the development of precise mathematical results concerning the effects of increased complexity; a degree of realism is sacrificed to achieve unambiguous conclusions.

Specifically, we assume that an integration test is performed (initially assumed to be instantaneous) when all the modules are completed. The test is perfect in that any flaws are certain to be detected. The system is defective and fails the test with probability \( p(n) \), an increasing function of the number of modules, \( n \). If flawed, the design must be completely reworked; however, the module completion time will tend to decrease with each iteration: when reworking a module, the completion time is exponentially distributed with a mean that is only a fraction \( r (<1) \) of the mean in the previous iteration.

The expected completion time, including rework, \( E[T_{CR}(n)] \), then becomes:

\[
E[T_{CR}(n)] = E[T_c(n)] + p(n)\{rE[T_c(n)]\} + p(n)^2\{r^2 E[T_c(n)]\} + ... \\
= E[T_c(n)] + E[T_c(n)]\{rp(n)^2\} + E[T_c(n)]\{rp(n)^2\} + ... \\
= \frac{E[T_c(n)]}{1-rp(n)} = E[T_c(n)] + \frac{rp(n)}{1-rp(n)} E[T_c(n)] \\
= E[T_c(n)] + \text{Rework}(n)
\]

(22)

To study the effect of rework, we express the first difference of \( E[T_{CR}(n)] \) as:

\[
\Delta E[T_{CR}(n)] = \Delta E[T_c(n)] + \Delta \text{Rework}(n) \quad \text{with} \quad \Delta \text{Rework}(n) = \frac{rp(n+1)(1-rp(n))E[T_c(n+1)] - rp(n)(1-rp(n+1))E[T_c(n)]}{(1-rp(n+1))(1-rp(n))}
\]

(24)

From Section 2.2, \( E[T_c(n)] \) is minimized at \( n_C^* \). Since \( p(n) \) is increasing in \( n \), (24) implies that \( \Delta \text{Rework} \) is positive for \( n \geq n_C^* \). Then (23) implies that \( \Delta E[T_{CR}(n)] \) is positive for \( n \geq n_C^* \), that is \( E[T_{CR}(n)] \) is increasing for \( n > n_C^* \). Hence

\[
n_{CR}^* \leq n_C^*.
\]
In other words, the optimum number of modules in the model that includes rework is no larger than the optimum number in the model including only communication loss.

We now remove the simplifying assumption that the integration test is instantaneous by introducing a function \( I(n) \) that relates integration test time to the number of modules. Since reworking tasks implies running the risk of creating regression errors (new errors created by fixing an old error), several iterations may be necessary. For simplicity we assume integration test time to be the same at each iteration of the project.

If we include integration test time the expected completion time, \( E[T_{CRI}(n)] \), becomes:

\[
E[T_{CRI}(n)] = E[T(n)] + \frac{I(n)}{1-p(n)} = E[T(n)] + \frac{I(n)}{1-rp(n)}
\]  
(25)

If we assume that \( p(n) \) is increasing in \( n \) and \( I(n) \) is nondecreasing in \( n \), then the introduction of the integration test time involves the addition of a factor that increases with \( n \). As in the previous section, if \( n_{CRI}^* \) minimizes \( E[T_{CRI}(n)] \), then it follows that

\[ n_{CRI}^* \geq n_{CR}^* \text{.} \]

This implies again that layering on additional complexity leads to an optimum number of modules that is no larger than in the simpler model which preceded it.

We are now in a position to summarize the results of our analysis with the following formula:

\[
E[T_{CRI}(n)] = E[T(n)] + \frac{(nC(n) + rp(n))}{1-rp(n)} E[T(n)] + \frac{I(n)}{1-p(n)}
\]  
(26)

The expected completion time depends on the extent of concurrency as reflected in the number of modules, \( n \), and is adversely affected by communication \( (C(n)) \), probabilities \( (p(n)) \) of rework \( (r) \) iterations, and integration test time \( (I(n)) \). The critical numbers that specify the optimal levels of concurrency are subject to the following inequality:

\[ n_{CRI}^* \leq n_{CR}^* \leq n_C^* < n^{***} \]

3. Numerical Illustration
The impact on project completion time of the above-mentioned factors is illustrated by an example whose results are shown in Figure 2.

Following Brooks (1975), the communication time is assumed to be proportional to the number of communication links between the modules:

\[ C(n) = \frac{\alpha}{2} n(n-1) \]  

(27)

In (27) \( \alpha \) denotes a scaling parameter. For this example we chose \( \alpha = 0.000077 \) such that for \( n=30 \), 50% of the expected completion time of the longest module is spent on communication (Abdel-Hamid and Madnick, 1992). Observe in Figure 2 that the expected completion time is indeed a convex function of \( n \), because \( C(n) \) is convex and increasing in \( n \). In this example, the minimizing value is achieved at \( n_C^* = 21 \).

Note that \( p(n) \) is an increasing function of the number of modules. Letting \( \beta \) equal 0.02, then the optimum number of modules is reduced to \( n_{CR}^* = 16 \) as shown in Figure 2. This example illustrates the general result that the inclusion of rework, defined as \( E[T_{CR}(n)] - E[T_C(n)] \), causes the optimum number of modules to decrease.

Integration test time is assumed here to be a linear function of the number of modules:

\[ I(n) = \begin{cases} 0 & n = 1 \\ \gamma n & n > 1 \end{cases} \]  

(17)

Inclusion of this function, with \( \gamma \) equal to 0.005, leads to an optimum number of modules \( n_{CRI}^* = 12 \).

We conclude that, for this example, when communication is introduced into the model, a finite optimal number of parallel modules occurs at \( n_C^* = 21 \). The optimal number of modules decreases as failure rate functions (\( n_{CR}^* = 16 \)) and integration test time (\( n_{CRI}^* = 12 \)) are included.
4. Conclusions

There is substantial research showing that concurrent engineering practices can dramatically reduce product development time. These practices include (a) information concurrency: early involvement of manufacturing, marketing, engineering, and other "functions" to anticipate problems and deal with them peremptorily, and (b) concurrency in time: development and subdivision of tasks into modules, to be solved in parallel (Blackburn, et al., 1994) These practices can be very effective when carried out by communicative, cross-functional teams.

This investigation demonstrates that concurrent engineering also involves tradeoffs. Through simple analytical models of the design process modeled after a problem at Alcatel, we show that simultaneous activities, as represented by the subdivision of tasks into parallel modules, ultimately impose penalties that can cause a net increase in project completion time. Concurrent engineering has limits.

In this paper we consider one facet of concurrency: time concurrency in the form of subdivision of a task into modules. We show that, in the ideal situation of infinite task divisibility with no penalties, there are no limits to concurrency. However, as the model moves closer to reality—with communication losses, integration test time and probability of rework—these factors impose limits on concurrency. The returns from additional simultaneity diminish, turn negative and eventually increase expected project completion time. The precise limit to effective concurrency is, of course, situation dependent and is not the point of the paper. The point is to provide a rigorous proof that these limits exist and that they are a function of project complexity. This is extremely relevant for management because it refutes the popular misconception that, in the pursuit of faster development, more concurrency is always better. Project managers should understand that the diminishing returns that come from continuing efforts at concurrency are a natural effect that is endemic to the process, not a deficiency in management.
This research does not question the value of concurrent engineering. On the contrary, we argue that, based on our models, initial efforts at concurrent project management should have a large payoff in time compression. A firm that has been using traditional, sequential development should see quick success upon adoption of principles of simultaneity. However, we also show that managers, at some point, will encounter diminishing returns from their concurrency efforts; therefore, the organization should be prepared to deal with this potentially-frustrating result.

Future research could go in many directions. On the technical side the effects of task time distributions other than the exponential could be investigated, as well as other forms of penalty functions that occur in testing and rework. Our work suggests a new line of empirical research into the nature of factors limiting concurrency under different development environments. A better understanding of the complex interrelations among team communications, testing and rework, for example, may allow us to develop more realistic models that can predict results in specific cases. In addition, this study did not try to model other forms of concurrency such as early involvement of the different organizational functions on project completion time (information concurrency), nor have we investigated the effects of overlapping stages (another form of time concurrency). We conjecture that these aspects of concurrency would also have limits, but this would be a question for subsequent research.

Acknowledgment

This research was supported by the Dean's Fund for Research, Owen Graduate School of Management, Vanderbilt University and by the INSEAD Research & Development Fund. Additional support was provided by the Air Force Office of Scientific Research, Grant number AFOSR-91-0440 and by the Center for U.S.-Japan Technology Management.
References:


Adverse Influences on Concurrent Engineering

Figure 1

System decomposed into modules

parallel module development

integration into system

integration test time

communication

work

integration testing
Figure 2
A Numerical Illustration of the Simultaneity Model