INFORMATION HIDING IN PRODUCT DEVELOPMENT: THE DESIGN CHURN EFFECT

Ali Yassine, Nitin Joglekar, Dan Braha, Steven Eppinger, Daniel Whitney

Information Hiding in Product Development:
The Design Churn Effect

Ali Yassine[1] . Nitin Joglekar[2] . Dan Braha[1] . Steven Eppinger[1] . Daniel Whitney[1]

[1]Massachusetts Institute of Technology, Cambridge, Massachusetts, 02139

[2]Boston University, Boston, Massachusetts, 02215

Abstract

Execution of a complex product development project is facilitated through its decomposition into an interrelated set of localized development tasks. When a local task is completed, its output is integrated through an iterative cycle of system-wide integration activities. Integration is often accompanied by inadvertent information hiding due to the asynchronous information exchanges. We show that information hiding leads to persistent recurrence of problems (termed as the design churn effect) such that progress oscillates between being on schedule and falling behind. The oscillatory nature of the PD process confounds progress measurement and makes it difficult to judge whether the project is on schedule or slipping. We develop a dynamic model of work transformation to derive conditions under which churn is observed as an unintended consequence of information hiding due to local and system task decomposition. We illustrate these conditions with a case example from an automotive development project and discuss strategies to mitigate design churn.

(Product Development, Design Process Modeling, Decomposition and Integration, Component and System Performance Generation, Information Hiding, Design Churn)

1. Introduction

"We just churn and chase our tails until someone says that they won't be able to make the launch date." Anonymous product development manager at an automobile manufacturer

The difficulty to accurately measure individual activity progress within the context of the overall program goals is well understood by product development (PD) managers. The above quote is taken from a study of PD management practices at a large automotive company (Mar

1999). Progress oscillates between being on schedule (or ahead of schedule) and falling behind. In many instances, development tasks are repeated and no one knows why. This is a universal phenomenon in PD settings. For instance, in the software development realm, Cusumano and Selby (1995) report that the progress is measured by the number of bugs that testers report to developers during the development process. They show a bug report (Figure 1) oscillating from a high number of bugs to a low number and back to a high number and so on. Other histories showing oscillatory behavior in PD processes have been observed in aerospace (Browning et al. 2000), automotive (McDaniel 1996; Mar 1999), electronics (Wheelwright and Clark 1992), and information system development (Joglekar 2001) settings.

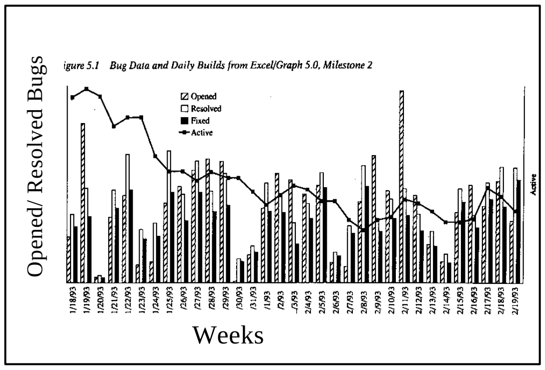


Figure 1: Evidence of Design Churn - Microsoft Excel (Cusumano and Selby 1995)

The Motivation for studying the churn phenomenon is abundant. The oscillatory nature of PD progress makes it hard to measure actual development progress and ultimately difficult to judge whether the project is on schedule or slipping. Other unfortunate consequences of churn may include significant increase in development times, organizational memory lapses regarding PD problem solving know-how, and deteriorated morale amongst developers. There are few managerial guidelines available for dealing with churn. Typically, a lack of understanding for the underlying causes of churn leads to myopic resource allocation decisions.

In this paper, we take an information-processing view of PD by characterizing the development process as a sequence of problem solving activities (Clark and Fujimoto 1991). Design churn is defined as a scenario where the total number of problems being solved (or progress being made) does not reduce (increase) monotonically as the project evolves over time.

There are several possible explanations for churn and this paper investigates one of them. We focus on the structural reason for churn; namely, delays associated with information dependencies. The information processing view postulates design decomposition to be a nested series of generation and testing activities (Simon 1996). If testing occurs simultaneously with the generation activities, then the process will not churn.[1] In reality, generation-testing cycles have built-in delays. This paper develops a generation-testing model with the capability to consider integration of several generation groups in the presence of delays.[2] The structure of the development process inherently results in some of the information related to the design tasks being sometimes hidden from other developers and managers.[3] Our premise is that in many development scenarios, design churn becomes an unintended consequence of information hiding.

Performance variation (i.e. imperfect evaluation) in the test activity may also cause churn. For instance, some systems exhibit non-monotonic reduction in either the variance or the expected value (and sometimes both) of design parameters due to uncertainty in performance evaluation (Browning et al. 2000). In order to avoid the confounding effects of variability (as it will only exacerbate churn), we deal only with the expected values and exclude performance variation as a plausible source of churn.[4] Other explanations for churn are also possible. Exogenous changes (e.g. a change in customer requirements) to design objectives also lead to churn (Mar 1999). Again, such changes will only confound the analysis of our basic premise and are excluded from the model. Furthermore, oscillatory allocation of resources as in "fire-fighting" models (e.g. Repenning et al. 2001) and in behavioral choice models (e.g. Ford and Sterman, 1999) exhibit churn-like behavior. These explanations are also excluded from our model based on similar rationale.

We explore our premise by developing a model for tracking the progress of PD processes while accounting for information hiding. Our model divides the development process into two

[1] It is customary in the PD literature to presume that a fully concurrent generation-testing cycle does not create more problems than it solves (Smith and Eppinger, 1997).
[2] In section 3, we propose a generalized decomposition model, where the generation activities are assigned to local or specialized groups, while testing is conducted by system-wide test and integration groups.
[3] Wheelwright and Clark (1992) describe how PD projects fail to meet their original potential due to intrinsic characteristics of the process and not due to a lack of creative people, technical skills, or management skills within the PD organization.
[4] For assessment of variability in PD refer to congestion models of development (e.g. Adler et al., 1995).

interdependent task sets: local and system. The structure of this problem solving process is set up such that local tasks, by definition, cannot hide information from system tasks about their individual progress and problems. On the other hand, system tasks may withhold information (gathered from local tasks) for limited periods of time before releasing it to local tasks. Between these releases, the information is hidden from local tasks, which work based on previously released information. Our model focuses on churning that is caused by these episodic releases of information.

For instance, the product architecture for a laptop computer enables the development team to decompose the laptop development project into local tasks such as the main board, LCD, and packaging design (Baldwin and Clark 2000). These local tasks, when completed, feed information into system testing and integration tasks. The integration tasks evaluate this information (based on system considerations) and provide individual feedback to the local tasks, which may require that local tasks perform extra iterations. System testing for the various pieces of the development process may take different times to process, and thus system feedback will be provided to local teams at different times. In effect, the inherent delays associated with generating the test results of system integration amount to information hiding. Conversely, if the laptop architectural decomposition required minimal (or no) interaction between the different local development tasks, then the system integration tasks may not be required to give feedback to local tasks and hence information hiding is non-existent.

Analysis of churn due to information hiding raises interesting questions about the convergence of the underlying system. We define PD convergence as a process in which problem solving activities result in a technically feasible design within a specified time frame.[5] That is, the total number of problems being solved falls below an acceptable threshold. The main results obtained from the analysis of this model are summarized as follows:

(1) The existence of design churn is a fundamental characteristic of the decomposition and integration of design between local and system teams. More specifically, it is shown that design churn may be attributed to two modes. The first mode reflects the 'fundamental

[5] A formalization of convergence in terms of conditions for stability is presented in section 4.3.

churn' of the design process, and the second mode, termed 'extrinsic churn', may be present depending on the relative rates of work completion and the rework induced between system and local tasks.

(2) It is possible for development processes to exhibit churning behavior under both converging and diverging scenarios. Conditions under which the total number of design problems associated with the system and local tasks converges to zero as the development time increases are presented.

The rest of the paper is organized as follows. In the next section, we discuss the literature relevant to information hiding and design churn. In Section 3 we propose a model for asynchronous information exchanges in a development environment. In Section 4.1, we introduce a PD model that involves a single local development team and single system integration team, and that accounts for information hiding. The basic model is formulated and analyzed in the rest of Section 4, where conditions for the convergence of the design process as well as "pure design churn" are presented. In Section 5, we present a generalized model that involves multiple local development teams that exchange information, under more general information release policies, with a corresponding system integration team. In Section 6, we apply the findings of the model to analyze the appearance design process for an automotive product development project. In Section 7, we discuss the managerial implications by identifying mitigation strategies to counter design churn in complex development processes.

2. Literature Review

Information hiding is not a new concept in management science. For instance, in the supply chain management literature, information hiding has been justified on grounds of either asymmetrical or distorted availability of information (Lee et al. 1997). Similar ideas have been explored in a segment of PD literature. For instance, in software development projects information hiding refers to the practice of keeping the implementation details of a software module hidden from other modules in the program (Sullivan et al. 2001). Typically, such practices are justified by the desire to reduce the coordination burden. However, formal models for capturing the effects of information hiding are rare in the PD literature.

There are several management science models that relate to one or more aspect of PD design churning. We group these models into the following categories: Set-based concurrent engineering, resource allocation, and information dependency.

2.1 Set-Based Concurrent Engineering:

Sobek et al. (1999) describe a method to model convergence in Toyota's PD process, called set-based concurrent engineering (SBCE). With SBCE, Toyota's designers think about sets of design alternatives, rather than pursuing one alternative iteratively. As the development process progresses, they gradually narrow the set until they come to a final solution. This set narrowing technique is also utilized in Krishnan's et al. (1997) model of design iteration. For example, Figure 2(a) shows that design parameter X converges with time. SBCE literature does not focus on instances where design churn is possible as shown in Figure 2(b). However, it is possible to extend these concurrency models to demonstrate and study churn (Mihm et al. 2001).

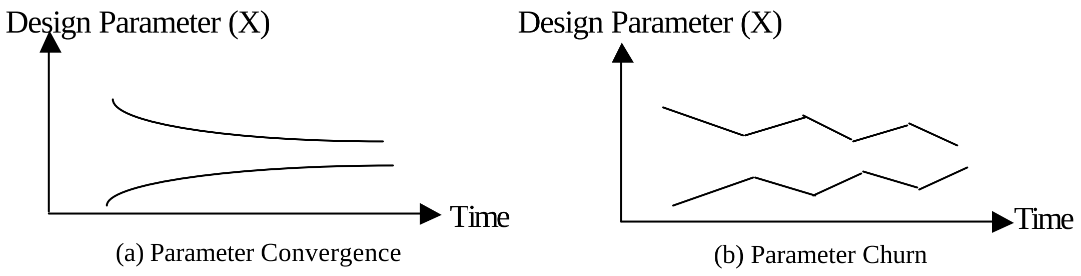


Figure 2: Parameter Convergence and Churn

2.2 Resource allocation:

Resource allocation has been identified as a managerial lever for controlling the rate of PD process completion (Ahmadi and Wang 1999). Bohn (2000) and Repenning et al. (2001) define the "firefighting" syndrome as the preemption of important, but not urgent, development activities due to an imminent necessity or problem (referred to as a "fire") in another part of the same development project (or another development project). Moving resources from one part of the project to another (or from one project to another) may trigger a vicious cycle of firefighting.

As a result, PD performance will oscillate. Conventional PD resource allocation studies (Adler et al. 1995; Loch and Terwiesch 1999) model waiting effects without focusing on design churn.

2.3 Information Dependency:

Information interdependency between development activities is an important feature of complex product development processes (Eppinger et al. 1994). Interdependency is manifested and measured by the amount of iteration and rework inherent in a PD process. The Design Structure Matrix (DSM) provides a simple mapping to capture interdependencies within a development process (Eppinger 2001). It is worth noting that DSM models may exhibit divergent churn behavior; however, both Smith and Eppinger (1997) and Browning and Eppinger (1998) artificially suppress this behavior.

Our treatment of design churn builds on the PD literature of task concurrency, resource allocation, and information dependency constructs. In particular, we use a DSM model as a building block to expand upon by introducing asynchronous information delays with these constructs.[6] In the next section, we will establish the linkages between asynchronous interdependencies and the DSM.

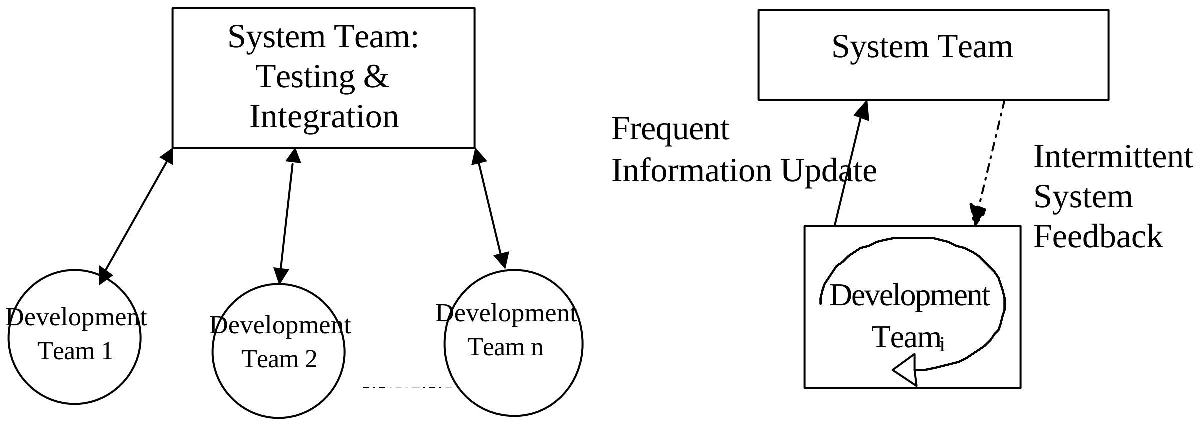
3. Asynchronous Information Interdependency in Design Processes

In a large and complex PD project, different development groups work concurrently on multiple aspects of the process (Joglekar et al. 2001). Work progresses within each group through internal iteration. Coordination between groups takes place through system level testing or an integration group. Individual (i.e., local) groups provide status updates to the system group. This information is processed based on global considerations, which may result in rework for some of the individual groups. Figure 3 shows a schematic of the information exchanges within the PD process described above. In the left side of the figure, we describe how a set of local development teams, working concurrently on a common project, interact through a system level team that coordinates and orchestrates their individual development efforts. The double-headed

[6] A control theory based matrix formulation using the DSM is a convenient approach to build our argument. However, the core ideas can be built using alternative approaches. See, for instance, Mihm el al (2001) for a selective evolutionary based exposition of related PD decisions.

arrow demonstrates the two-way communication that takes place. The right side of the figure depicts the interaction process between a single local development team and the system team. The solid arrow indicates that local teams frequently provide the system with updates regarding their progress, while the dotted arrow indicates that system team provides intermittent feedback to the local team.

The frequency of system level feedback might depend on either exogenous considerations (such as suppliers' ability to provide updates) or endogenous considerations such as system level


System Team: Testing & Integration
Development Team 1
Development Team 2
Development Team n
System Team
Frequent Information Update
Intermittent System Feedback
Development Team$_i$

(a) Interaction between local and system teams

(b) Synchronization between local and system teams

Figure 3: Local and System Bifurcation of Information

test requiring a minimal turn around time for a desired fidelity (Thomke and Bell 2001). If the synchronization is effectively instantaneous, for example during daily builds of Microsoft's Development Cycles, then we can think about the whole process in terms of a unified (combining local and system level) structure. Smith and Eppinger (1997) have developed a method using linear systems theory to analyze such models and identified controlling features of a unified iteration process. Unified iteration does not allow for information delays between local and system task execution. However, many of PD processes are characterized by intermittent system feedback.[7] Hence, we explore the management of multiple development teams

[7] This is a common PD observation since system teams need time to absorb and integrate all the local information they receive before sending feedback. Consequently, there is a delay from the time system teams receive local information until the time they send it back to local teams. Furthermore, information hiding and delays occur due to the fact that local teams, once they receive system feedback, do not usually drop all things at hand and immediately act on or respond to this new information. Usually, this new information is queued or batched with other updates.

coordinated through a system integration team and subject to periodic feedback (Joglekar and Yassine 2001).

The DSM shown in Figure 4 captures the above development setup. The DSM is composed of blocks that represent several local development teams and a system integration team. The system team facilitates interactions between local teams as represented by the solid arrows in the figure. The local DSMs are internally updated at every time step (ΔT); and provide status information to the system DSM at $t_{i,S}$ periodic intervals. The system DSM provides updates to the local DSMs at periodic intervals $T_1, T_2, \ldots, T_m$. The local and system update periods (i.e., $t_{i,S}$'s or T_i's) may or may not be synchronous; e.g., $T_1 = k_1?T, \ldots, T_m = k_m?T$ where k_i are integer constants for all i's. In addition, the dotted arrows demonstrate an instance where local teams are allowed to interact directly (i.e., without the facilitation of the system team); in which case, the local DSM L_i provides status information to other local DSMs at periodic intervals $t_{i,1}, t_{i,2}, \ldots, t_{i,m}$.

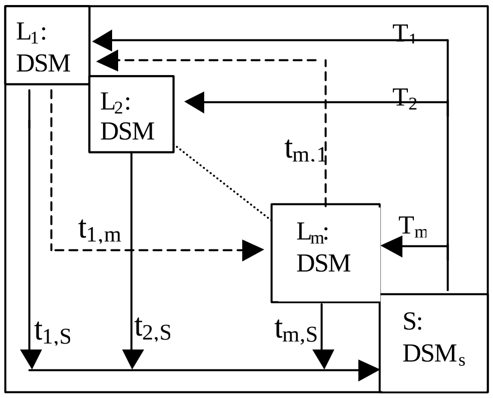


Figure 4: DSM Representation of a PD Process Showing Local and System Teams
(Li: represent a local development team, and S: represent a system team)

This type of DSM is not a pathological case. Numerous researchers have documented the existence of this local/system bifurcation (Sosa et al. 2000). The problem cannot be treated as a single DSM to study the churning properties of the development process due to time delays and asynchrony in information transfers between the system and different local groups.

4. Asynchronous Work Transformation Model: Single Local DSM Case

4.1 Model Formulation

First, we study a simplified version of the problem. We assume, without loss of generality, that there exists a single local DSM (containing the local tasks) that exchanges information with a corresponding system DSM at every time step. The system DSM releases information every T time steps.[8] Consistent with Smith and Eppinger (1997), we specify that all the tasks associated with the local and system DSMs are internally updated at each iteration step. We label $L(k)$ as the vector for the amount of unfinished work in the local tasks at time k. Absent all system feedback, the progress of $L(k)$ is given by:

$$L(k) = \mathrm{W}^{\mathrm{L}} L(k-1) \qquad k = 1, 2, ... \qquad (1)$$

where W^{L} is the work transformation matrix that captures the fraction of rework created within a local group of tasks (Smith and Eppinger 1997). Equation (1) describes the work transformation during each iteration stage as follows. Local tasks finish a fraction of their own work, given a constant completion rate specified in the diagonal of W^{L}. However, this work causes some rework to be created to other dependent tasks. The off-diagonal elements of W^{L} document such dependencies. The construction of W^{L} is detailed in Appendix A.

We augment the state space for the above model by introducing two more vectors: $S(k)$ and $H(k)$. The vector $S(k)$ represents the amount of unfinished work in all system tasks at time step k, and $H(k)$ is a vector for the amount of finished system work at time step k that is ready to be transmitted to local tasks but remains hidden until it is released. We also define a matrix W^{S} that corresponds to $S(k)$ in a manner analogous to the relation between W^{L} and $L(k)$, that is

$$S(k) = \mathrm{W}^{\mathrm{S}} S(k-1) \qquad k = 1, 2, ... \qquad (2)$$

Combining both state equations (1 and 2) and incorporating both types of information exchanges (from local to system and vice versa), we obtain the state equation (3). This equation assumes that the system transmits all the work withheld up until the last moment before data transmittal to local tasks.

[8] The model is capable of accommodating multiple local DSMs as discussed in Section 5. Furthermore, for the sake of simplicity and ease of exposition, we assume that these local DSMs and the system DSM have the same rank. Finally, the system can release information once or in multiple periods.

$$\begin{bmatrix} L(k+1) \\ S(k+1) \\ H(k+1) \end{bmatrix} = \underbrace{\begin{bmatrix} \mathrm{W}^{\mathrm{L}} & 0 & 0 \\ \mathrm{W}^{\mathrm{LS}} & \mathrm{W}^{\mathrm{S}} & 0 \\ 0 & \mathrm{W}^{\mathrm{SH}} & I \end{bmatrix}}_{\mathbf{A}^{\mathbf{Hold}}} \begin{bmatrix} L(k) \\ S(k) \\ H(k) \end{bmatrix} (1-d_T(k)) + \underbrace{\begin{bmatrix} \mathrm{W}^{\mathrm{L}} & \mathrm{W}^{\mathrm{SL}} & \mathrm{I} \\ \mathrm{W}^{\mathrm{LS}} & \mathrm{W}^{\mathrm{S}} & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{\mathbf{A}^{\mathbf{Release}}} \begin{bmatrix} L(k) \\ S(k) \\ H(k) \end{bmatrix} d_T(k) \qquad (3)$$

In Equation (3), $d_T(k) = \sum_{k=0}^{\infty} d(k - jT)$ is the periodic impulse train function, where $d(k-n)$ is the unit impulse (or unit sample) function defined as:

$$d(k-n) = \begin{cases} 1 & k = n \\ 0 & k \neq n \end{cases} \qquad (4)$$

The matrix $\mathrm{A}^{\mathrm{Hold}}$ is active at each iteration step except for every T periods when the system team releases its feedback to the local team and the matrix $\mathrm{A}^{\mathrm{Release}}$ becomes active. W^{LS} is a matrix that captures the rework fraction created by local tasks $L(k)$ for the corresponding system tasks $S(k)$. Similarly, when information is released by the system, the matrix W^{SL} captures the rework fraction created directly by the system tasks $S(k)$ for the local tasks $L(k)$. W^{SH} is a matrix that captures the rework created for the local tasks by the system tasks, and is placed in a hidden (or holding) state until it is time to be transmitted to local tasks. When no information is being released by the system to local tasks, the identity sub-matrix in $\mathrm{A}^{\mathrm{Hold}}$ guarantees that finished system work is carried over to the next period. The identity sub-matrix in $\mathrm{A}^{\mathrm{Release}}$ guarantees that finished system work is transmitted to local tasks, through *H(k),* every T time steps. Consequently, $H(k)$ gets set to zero each T steps and is rebuilt in between. The construction of the work transformation matrices W^{L}, W^{S}, W^{LS}, W^{SH} and W^{SL} is dependent on the structure of the information exchanged within the development process. In Appendix A, we specify (consistent with the case study presented in Section 6) the work transformation matrices based on the local and system DSMs Ω^{L}, Ω^{S}; as well as the *inter-component dependency* matrices Ω^{LS}, Ω^{SL}, which represent the interaction between local and system teams.[9]

Individual elements within the L, S, and H vectors refer to the same task. To illustrate the concept, consider the following two tasks : door trim design and garnish trim design related to

[9] The local and system DSMs as well as the inter-component dependency matrices represent the amount of rework created for each task *based on work done on the other tasks in the previous period.*

the development of a car door. The state equations for this problem are shown in Equations (5) and (6) for the case when no information is being released by the system (e.g., the 'body' integration team) to local tasks (e.g., the 'door' design team), and for the case when information is released by the system, respectively.

In this example, $L_1(k)$ and $S_1(k)$ designate the number of design problems or open issues associated with the door trim task, which are being worked by the local design team and system integration team, respectively. $H_1(k)$ refers to the number of door trim problems resolved by the system integration team that are waiting to be released for future work by the local design team. Any problem associated with the door trim design can reside in only one of these three states until it is fully resolved. Note that $1-w_{11}^{L}$ and $1-w_{22}^{L}$ are the fractions of L_1 and L_2 respectively that can be completed in an autonomous manner in every time step. Furthermore, $w_{12}^{L}L_2(k)$ and $w_{21}^{L}L_1(k)$ are the amounts of rework that get created for task L_1 and L_2, respectively, as a consequence of the autonomous progress. Similar interpretations can be made for the system matrix (i.e., w_{ij}^{S}).

$$\begin{bmatrix} L_1(k+1) \\ L_2(k+1 \\ S_1(k+1) \\ S_2(k+1) \\ H_1(k+1) \\ H_2(k+1) \end{bmatrix} = \begin{bmatrix} w_{11}^{L} & w_{12}^{L} & 0 & 0 & 0 & 0 \\ w_{21}^{L} & w_{22}^{L} & 0 & 0 & 0 & 0 \\ w_{11}^{LS} & w_{12}^{LS} & w_{11}^{S} & w_{12}^{S} & 0 & 0 \\ w_{21}^{LS} & w_{22}^{LS} & w_{21}^{S} & w_{22}^{S} & 0 & 0 \\ 0 & 0 & w_{11}^{SH} & w_{12}^{SH} & 1 & 0 \\ 0 & 0 & w_{21}^{SH} & w_{22}^{SH} & 0 & 1 \end{bmatrix} \begin{bmatrix} L_1(k) \\ L_2(k) \\ S_1(k) \\ S_2(k) \\ H_1(k) \\ H_2(k) \end{bmatrix} \tag{5}$$

$$\begin{bmatrix} L_1(k+1) \\ L_2(k+1 \\ S_1(k+1) \\ S_2(k+1) \\ H_1(k+1) \\ H_2(k+1) \end{bmatrix} = \begin{bmatrix} w_{11}^{L} & w_{12}^{L} & w_{11}^{SL} & w_{12}^{SL} & 1 & 0 \\ w_{21}^{L} & w_{22}^{L} & w_{21}^{SL} & w_{22}^{SL} & 0 & 1 \\ w_{11}^{LS} & w_{12}^{LS} & w_{11}^{S} & w_{12}^{S} & 0 & 0 \\ w_{21}^{LS} & w_{22}^{LS} & w_{21}^{S} & w_{22}^{S} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} L_1(k) \\ L_2(k) \\ S_1(k) \\ S_2(k) \\ H_1(k) \\ H_2(k) \end{bmatrix} \tag{6}$$

4.2 Model Analysis

In this section, we explore the fundamental characteristics of the model described in Equation (3). All proofs are presented in Appendix B.

First, we notice that Equation (3) can be rewritten as follows:

$$x(k+1) = A(k)x(k) \qquad (7)$$

where $x(k) = \begin{bmatrix} L(k) \\ S(k) \\ H(k) \end{bmatrix}$ and $A(k) = \begin{bmatrix} \mathrm{W}^{\mathrm{L}} & d_T(k)\mathrm{W}^{\mathrm{SL}} & d_T(k)\mathrm{I} \\ \mathrm{W}^{\mathrm{LS}} & \mathrm{W}^{\mathrm{S}} & 0 \\ 0 & (1-d_T(k))\mathrm{W}^{\mathrm{SH}} & (1-d_T(k))I \end{bmatrix}$

Thus, the model described in Equation (3) is a *homogenous linear difference system* that is *nonautonomous*, or *time-variant*. Moreover, since the impulse train function $d_T(k)$ is periodic with period T (recall that the system DSM releases information every T time steps), we conclude that for all $k \in \mathrm{Z}$ (where Z is the set of all positive integers), $A(k+T) = A(k)$. That is, the model described in Equation (7) is a *linear periodic system*.

We now present some results obtained using Floquet theory (Richards 1983) for the linear periodic system given in Equation (7).[10]

Definition 1. Matrix $C = A(T-1)A(T-2)\cdots A(0)$ is referred to as the *monodromy matrix* of (7).

In the following we assume that the monodromy matrix is diagonalizable.[11] C is diagonalizable if and only if it has linearly independent eigenvectors. A sufficient condition for C to be diagonalizable is that it has distinct eigenvalues (Strang 1980). We cite the following result from Richards (1983) as Lemma 1, Theorem 1, and corollary 1 to set up further analysis.

Lemma 1. Let C be a diagonalizable $n \times n$ matrix, and let T be any positive integer. Let us decompose C as $C = S_C \Lambda_C S_C^{-1}$, where Λ_C is a diagonal matrix of the eigenvalues of C, and S_C is the corresponding eigenvector matrix. Then, there exists some $n \times n$ matrix B such that $B^T = C$. Moreover, $B = S_C \Lambda_B S_C^{-1}$, where $\Lambda_B = \sqrt[T]{\Lambda_C}$.

The following result indicates that the analysis of the periodic system described in Equation (7) is reduced to the study of a corresponding autonomous linear system.

Theorem 1. If $y(k)$ is a solution of the autonomous linear system

$$y(k+1) = By(k) \qquad (8)$$

Then, the general solution $x(k)$ of the linear periodic system (7) is given as follows

[10] Floquet theory has been mainly applied in the mathematical and the physical sciences (Kuchment, 1993). However, to the best of our knowledge, Floquet theory has not been applied in the social and management sciences.
[11] As observed in Smith and Eppinger (1997), the diagonalization assumption reflects reality. The qualitative results, however, will remain invariant in the general case; though the computation of the underlying matrices becomes more complicated.

$$x(k) = P(k)B^k g \qquad (9)$$

where $P(k)$ is a nonsingular periodic matrix of period T, and $g \in \mathbb{R}^n$ is a constant vector.[12]

Corollary 1. The general solution $x(k)$ of the linear periodic system (7) is given by

$$x(k) = P(k)y(k) \qquad (10)$$

where $y(k)$ is the general solution of the autonomous linear system (8).

Corollary 1 has the following interesting interpretation for the information hiding problem in PD. We note that there are two sources of oscillation that govern the development of the total number of problems being solved as the project evolves over time. The first source is associated with the periodic matrix $P(k)$ in equation (10), and reflects the 'fundamental churn' of the process. This 'fundamental churn' may be attributed to the intrinsic characteristic of information delays between local and system task execution. The second source of oscillation, termed 'extrinsic churn,' is associated with the properties of the linear autonomous system (8) as discussed in Smith and Eppinger (1997). More specifically, positive real eigenvalues of B correspond to non-oscillatory behavior of the solution $y(k)$. Negative and complex eigenvalues of B describe damped oscillations. The overall property of the linear periodic system (7) is thus the combined effect of both sources of oscillation.

Corollary 1 allows the development of conditions under which the linear periodic system (7) converges (i.e., as the time increases to infinity the total number of design problems associated with the system and local tasks converges to zero). We show in Section 4.3 that the eigenvalues and the eigenvectors of the matrix B determine conditions of convergence.

4.3 Conditions for Stability

In this section, we present conditions under which the total number of design problems associated with the system and local tasks converges to *zero* as the time increases to infinity.

First, we note that the zero solution is an *equilibrium point*[13] of (7). Next we introduce the definitions of stability of the equilibrium point.

Definition 2. The equilibrium point x^* is

[12] Any solution of (8) may be obtained from the general solution by a choice of vector g based on initial conditions.

[13] A point x^* is called an equilibrium point of (8) if $x^* = A(k)x^*$ for all $k \geq 0$.

(1) *stable* if given $e > 0$ there exists $d = d(e)$ such that $\left\| x_0 - x^* \right\| < d$ implies $\left\| x(k) - x^* \right\| < e$ for all $k \geq 0$. x^* is *unstable* if it is not stable.

(2) *globally attracting* if $\lim_{k \to \infty} x(k) = x^*$ for any initial work vector x_0.

(3) *asymptotically stable* if it is stable and globally attracting.

Intuitively, the zero solution is stable if the total number of design problems associated with the system and local tasks remains bounded as the project evolves over time. Asymptotic stability requires the additional condition that the total number of design problems associated with the system and local tasks converges to the origin for any initial work vector.

When the PD process involves time delays and asynchrony in information transfer between the system and local group, conditions for the convergence of the development process are of vital importance for PD management. Before we present stability conditions for the asynchronous work transformation model, we introduce the so-called '*Floquet exponents*' and '*Floquet multipliers*' of the linear periodic system (7). Floquet exponents are the eigenvalues l of B; while the corresponding eigenvalues l^T of the monodromy matrix (C) are the Floquet multipliers. We have the following result:

Theorem 2. The zero solution of (7) is stable *if and only if* the Floquet exponents have magnitude less than or equal to 1, and asymptotically stable *if and only if* all the Floquet exponents have magnitude less than 1.

The following provides an additional result that explains the behavior of solutions of the asynchronous work transformation model:

Corollary 2 The zero solution of (7) is stable *if* the Floquet multipliers have magnitude less than or equal to 1 and asymptotically stable *if* all the Floquet multipliers have magnitude less than 1.

A direct consequence of Theorem 2 is that the Floquet exponents and their corresponding eigenvectors (i.e., eigenvectors of B) determine the rate and nature of convergence of the design process. Consistent with Smith and Eppinger (1997), we use the term *design mode* to refer to an eigenvalue of B along with its corresponding eigenvector.[14] The magnitude of each eigenvalue

[14] For autonomous linear systems (i.e., $A(k) = A$), the period of the matrix $A(k)$ is $T = 1$, the monodromy matrix $C = A$, and the Floquet multipliers are simply the eigenvalues of A. Thus, the Smith and Eppinger (1997) model is a special case of equation (7).

of B determines the geometric rate of convergence of one of the design modes; while the corresponding eigenvector identifies the relative contribution of each of the various constituent tasks to the amount of work that jointly converges at the given geometric rate (Smith and Eppinger 1997). The eigenvector corresponding to the largest magnitude eigenvalue of B (most slowly converging design mode) provides useful information regarding design tasks that require significant amount of work. More specifically, the larger the magnitude of an element in that eigenvector, the stronger the element contributes to the slowly converging design mode.

4.4 Conditions for "Pure Churn"

"Pure design churn" is defined as a scenario where the total number of problems being solved oscillates freely as the project evolves over time and *neither* convergence nor divergence occurs. "Pure design churn" means that the amount of unfinished work does not decrease simultaneously for all of the tasks. Instead, the amount of unfinished work shifts from task to task as the project unfolds. The above scenario is represented by particular solutions that are *periodic*; i.e., solutions $x(k)$ where for all $k \in \mathbb{Z}$, $x(k+N) = x(k)$ for some positive integer N. The following results hold:

Theorem 3.

(i) The linear system (7) has a periodic solution of period T if the monodromy matrix C has an eigenvalue of equal to 1.

(ii) The linear system (7) has a periodic solution of period $2T$ if the monodromy matrix C has an eigenvalue equal to -1.

(iii) If the largest magnitude eigenvalue of the monodromy matrix C equals to 1 and is strictly greater (in absolute value) than any other eigenvalue, then the limiting behavior of the general solution of the linear system (7) is periodic with period T .

5. Asynchronous Work Transformation Model: Multiple Local DSM Case

In this section, we consider the general case where multiple local teams are coordinated through a system integration team and subject to periodic feedback. More specifically, the m local DSMs are internally updated and provide status information to others (local and system DSMs) at every time step. The system DSM provides updates to the m local DSMs at periodic intervals $T_1, T_2, ..., T_m$ as shown in Figure 4.

We label L_i as the vector that designates the amount of unfinished work of the tasks of local team i ($i = 1,...,\mathrm{m}$) at time k. Let n_i denote the number of local tasks in local team i, and let $n = \sum n_i$ denote the total number of tasks in all of the local teams. Individual elements within the L_i ($i = 1,...,m$), S_i, and H_i vectors refer, correspondingly, to the same task. In general, the system of equations is written as follows:

$$\underbrace{\begin{bmatrix} L_1(k+1) \\ \vdots \\ L_m(k+1) \\ S_1(k+1) \\ \vdots \\ S_m(k+1) \\ H_1(k+1) \\ \vdots \\ H_m(k+1) \end{bmatrix}}_{x(k+1)} = \underbrace{\begin{bmatrix} \mathrm{W}^{\mathrm{L}_1} & \cdots & \mathrm{W}^{\mathrm{L}_m\mathrm{L}_1} & d_{T_1}(k)\mathrm{W}^{\mathrm{S}_1\mathrm{L}_1} & \cdots & d_{T_1}(k)\mathrm{W}^{\mathrm{S}_m\mathrm{L}_1} & d_{T_1}(k)I & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & 0 & \ddots & 0 \\ \mathrm{W}^{\mathrm{L}_1\mathrm{L}_m} & \cdots & \mathrm{W}^{\mathrm{L}_m} & d_{T_m}(k)\mathrm{W}^{\mathrm{S}_1\mathrm{L}_m} & \cdots & d_{T_m}(k)\mathrm{W}^{\mathrm{S}_m\mathrm{L}_m} & 0 & 0 & d_{T_m}(k)I \\ \mathrm{W}^{\mathrm{L}_1\mathrm{S}_1} & \cdots & \mathrm{W}^{\mathrm{L}_m\mathrm{S}_1} & \mathrm{w}^{\mathrm{S}}_{11} & \cdots & \mathrm{w}^{\mathrm{S}}_{1n} & 0 & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & 0 & 0 & 0 \\ \mathrm{W}^{\mathrm{L}_1\mathrm{S}_m} & \cdots & \mathrm{W}^{\mathrm{L}_m\mathrm{S}_m} & \mathrm{w}^{\mathrm{S}}_{n1} & \cdots & \mathrm{w}^{\mathrm{S}}_{nn} & 0 & 0 & 0 \\ 0 & 0 & 0 & (1-d_{T_1}(k))\mathrm{W}^{\mathrm{S}_1\mathrm{H}_1} & \cdots & (1-d_{T_1}(k))\mathrm{W}^{\mathrm{S}_m\mathrm{H}_1} & (1-d_{T_1}(k))I & 0 & 0 \\ 0 & 0 & 0 & \vdots & \ddots & \vdots & 0 & \ddots & 0 \\ 0 & 0 & 0 & (1-d_{T_m}(k))\mathrm{W}^{\mathrm{S}_1\mathrm{H}_m} & \cdots & (1-d_{T_m}(k))\mathrm{W}^{\mathrm{S}_m\mathrm{H}_m} & 0 & 0 & (1-d_{T_m}(k))I \end{bmatrix}}_{A(k)} \underbrace{\begin{bmatrix} L_1(k) \\ \vdots \\ L_m(k) \\ S_1(k) \\ \vdots \\ S_m(k) \\ H_1(k) \\ \vdots \\ H_m(k) \end{bmatrix}}_{x(k)} \quad (11)$$

In the above expression, $\mathrm{W}^{\mathrm{L}_i}$ is a work transformation matrix that captures the fraction of rework created within the group of tasks of local team i. W^{S} is the work transformation matrix that captures the fraction of rework created within the system tasks. $\mathrm{W}^{\mathrm{S}_i\mathrm{H}_j}$ is a $n_j \times n_i$ matrix that captures the fraction of finished system work created by system tasks $S_i(k)$ for the local tasks $L_j(k)$, and is held in $H_j(k)$ until the next scheduled information release. $\mathrm{W}^{\mathrm{L}_i\mathrm{L}_j}$ is a $n_j \times n_i$ matrix that captures the fraction of rework created by local tasks $L_i(k)$ for the local tasks $L_j(k)$. $\mathrm{W}^{\mathrm{L}_i\mathrm{S}_j}$ is a $n_j \times n_i$ matrix that captures the fraction of rework created by local tasks $L_i(k)$ for the system tasks $S_j(k)$. Since information is released by the system to the local team i only at periodic intervals of T_i, the $n_i \times n_i$ diagonal sub-matrix $(1 - d_{T_i}(k))I$ guarantees that finished system work is carried over to the next period. When information is released by the system to local team j, the $n_j \times n_i$ matrix $d_{T_j}(k)\mathrm{W}^{\mathrm{S}_i\mathrm{L}_j}$ captures the fraction of rework created *directly* by the system tasks $S_i(k)$ for the local tasks $L_j(k)$. The $n_i \times n_i$ diagonal sub-matrix

$d_{T_i}(k)I$ indicates that information is transmitted to the local tasks $L_i(k)$ *indirectly* through the holding state $H_i(k)$.

The next result shows that the model described in Equation (11) is a special case of a *linear periodic system*. Once the period of the matrix $A(k)$ is identified, the monodromy matrix C can be determined, and the results presented in Section 4 can be readily employed.

Theorem 4. If the system team provides updates to m local teams at periodic intervals $T_1, T_2, ..., T_m$, then the fundamental period T of the linear matrix $A(k)$ is the *least common multiple* of $T_1, T_2, ..., T_m$; i.e., $T = \mathrm{lcm}(T_1, T_2, ..., T_m)$.

Following a similar reasoning as in Theorem 4, it can be shown that any *periodic information release policy* will lead to a linear periodic system, and thus can be analyzed using the tools presented in Section 4. For example, the local teams may provide status information to others (local and system teams) at periodic intervals $t_1, t_2, ..., t_m, t_{\text{system}}$, rather than at every time step; or any team (local or system) may provide information status to others (local or system teams) at *non-uniform* (but periodic) intervals. Indeed, any such periodic information release policy can be transformed to a model, where all elements $a_{ij}(k)$ of the linear matrix $A(k)$ are periodic functions (with possibly *non-identical* periods). In this case, Theorem 4 can be adapted by letting the fundamental period T of the linear matrix $A(k)$ to be the least common multiple of the periods of the elements $a_{ij}(k)$.

6. Case Study: The Automotive Appearance Design Process

In this section, an illustration of the asynchronous work transformation model in a real product development process, previously reported by McDaniel (1996), is presented. We intend to demonstrate internal process dynamics, show that oscillatory patterns arise in an asynchronous PD project, and assess several mitigation strategies by exploiting the results developed in the paper. In Section 6.1, we provide a general overview of the automotive appearance design process. Section 6.2 demonstrates how to construct the underlying work transformation matrices. Then, in Section 6.3 we analyze the base case model. Section 6.4 assesses the efficacy of churn mitigation strategies based on three operational scenarios. Finally, results of sensitivity analysis are presented in Section 6.5.

6.1 Appearance Design Process Overview

Appearance design refers to the process of designing all interior and exterior automobile surfaces for which appearance, surface quality and operational interface is important to the customer. Such design items include, for example, exterior sheet metal design and visible interior panels. Appearance design is the earliest of all physical design processes, and changes in this stage easily cascade into later development activities causing costly rework. This is avoided by allowing "stylists" (from the industrial design group) to work closely with "engineers" (from the engineering design group). While stylists are responsible for the appearance of the vehicle, engineers are responsible for the feasibility of the design by ensuring that it meets some functional, manufacturing, and reliability requirements. Figure 5 shows the industrial design process within the context of the overall automotive product development process. The industrial design portion is allotted approximately 52 weeks for completion in a typical vehicle program.

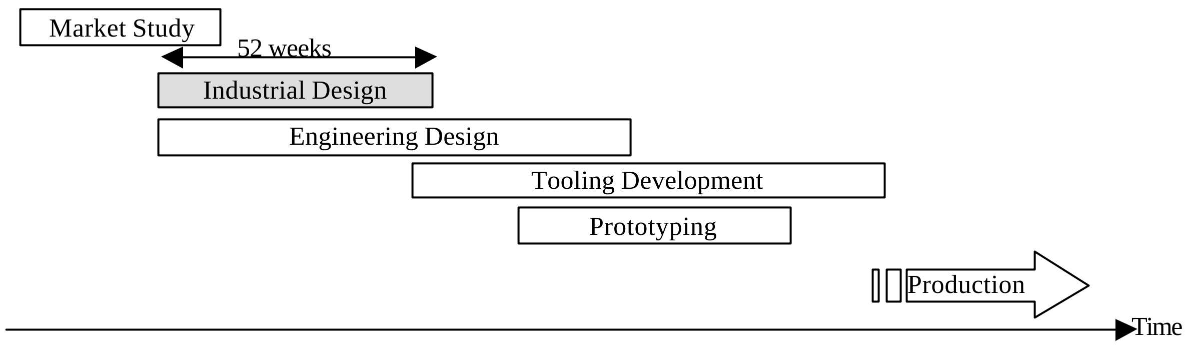


Figure 5: Appearance Design in Relation to Total Development Process

Records from the study company, shown in Figure 6, indicate churning behavior for a specific vehicle program. While the curves presented in the figure show churn in both interior and exterior subsystem development, our analysis of the churn phonemenon will be limited to the interior design process involving the styling and engineering developement organizations.

Information exchanges from styling to engineering take the form of wireframe CAD data generated from clay model scans; referred to as scan transmittals of surface data. Scan transmittals are scheduled at roughly six weeks intervals (i.e., T=6). Information exchanges between engineering and styling occur on a weekly basis through a scheduled feasibility meeting. During these meetings various engineering groups provide feedback to styling on infeasible design conditions. Therefore, with this information transfer setup engineering will be the local team, as defined in our model, and styling will be the system team.

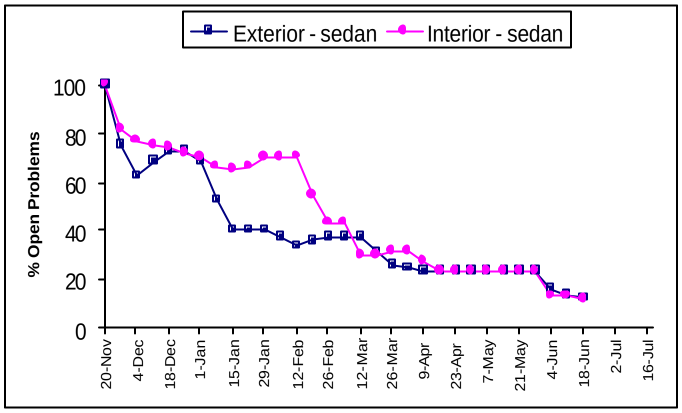


Figure 6: Churning Behavior Observed in a Family of Vehicle Programs (McDaniel 1996)

In addition to the cross-functional information exchanges between styling and engineering, information flows also occur within functional groups. For example, within engineering, a hand clearance study would compile information about the front door trim panel and the front seat to determine whether the two components physically interfere, and whether the space between them meets minimum acceptable requirements.

6.2 Construction of Work Transformation Matrices

From the program management perspective, the vehicle interior is segmented into sub-systems, or components. These components represent major sub-assemblies of the interior, and include typical components such as the instrument panel, the front door trim panels, and the center console. This level of component aggregation is used primarily because these components have been the unit of management and budgetary control for engineering design work, and because the company defined a number of standard engineering design studies to be performed on each component at this level. The DSMs (Ω^L, Ω^S) for the engineering and industrial design processes are shown in Figure 7(a) and 7(b), respectively. The transformation of component-level design information to system-level design information, as used within the industrial design group, is captured by the 'dependency' matrix Ω^{LS} in Figure 7(c). This transformation is typically performed on a weekly basis, when the engineering group provides feedback to the industrial design on infeasible conditions. Similarly, the ‘dependency’ matrix Ω^{SL} in Figure 7(d) captures the impact of industrial design on the engineering process at each scan transmittal (on a six-week interval).

			1	2	3	4	5	6	7	8	9	10
1	L_1	Carpet	0.85	0.12	0.02	0.06	0.06				0.06	
2	L_2	Center Console	0.1	0.53	0.04			0.3	0.02		0.24	0.02
3	L_3	Door Trim Panel	0.02	0.04	0.47	0.08		0.24	0.02		0.18	0.02
4	L_4	Garnish Trim	0.06		0.18	0.68		0.14	0.1	0.02	0.08	
5	L_5	Overhead System	0.04				0.83					
6	L_6	Instrument Panel		0.3	0.26	0.16		0.28	0.06		0.02	0.2
7	L_7	Luggage Trim		0.02	0.02	0.1		0.06	0.76	0.06	0.04	
8	L_8	Package Tray				0.1			0.06	0.83	0.16	
9	L_9	Seats	0.08	0.24	0.18	0.08		0.04	0.04	0.16	0.63	0.2
10	L_{10}	Steering Wheel		0.02	0.02			0.26			0.2	0.7

(a) Local DSM- Ω^L (i.e., engineering)

			1	2	3	4	5	6	7	8	9	10
1	S_1	Carpet	0.2									
2	S_2	Center Console		0.2								
3	S_3	Door Trim Panel			0.2							
4	S_4	Garnish Trim				0.2						
5	S_5	Overhead System					0.2					
6	S_6	Instrument Panel						0.2				
7	S_7	Luggage Trim							0.2			
8	S_8	Package Tray								0.2		
9	S_9	Seats									0.2	
10	S_{10}	Steering Wheel										0.2

(b) System DSM- Ω^S (i.e., industrial design)

		1	2	3	4	5	6	7	8	9	10
1	Carpet	▨									
2	Center Console		▨	0.09	0.17	0.21	0.09	0.14	0.42	0.29	0.38
3	Door Trim Panel		0.12	▨	0.6	0.24	0.1	0.16	0.49	0.34	0.44
4	Garnish Trim		0.06	0.15	▨	0.12		0.16	0.49	0.08	0.22
5	Overhead System		0.05		0.08	▨					
6	Instrument Panel		1	0.87	0.58		▨	0.94	1.41	0.49	3.81
7	Luggage Trim		0.07	0.06	0.25			▨			
8	Package Tray				0.08				▨	0.07	
9	Seats		0.14	0.12	0.12				0.58	▨	
10	Steering Wheel				0.05						▨

(c) Ω^{LS} (converting local issues to system issues)

		1	2	3	4	5	6	7	8	9	10
1	Carpet	0.15									
2	Center Console		0.15								
3	Door Trim Panel			0.15							
4	Garnish Trim				0.15						
5	Overhead System					0.15					
6	Instrument Panel						0.15				
7	Luggage Trim							0.15			
8	Package Tray								0.15		
9	Seats									0.15	
10	Steering Wheel										0.15

(d) Ω^{SL} (converting system issues to local issues)

Figure 7: Local, System DSMs, and System/Local conversion matrices

The average autonomous completion rates per component are shown along the diagonal of the local and system DSMs (i.e., Ω^L and Ω^S, respectively).[15] To set a base level of normalized resource usage for each component, engineers defined the resource usage intensity required to accomplish the autonomous completion rates presented in Figure 7 as *one resource-week*. The DSMs for styling and engineering were obtained by circulating a survey instrument, to both groups. Respondents were asked to populate the DSM by estimating the pairwise coupling (i.e., dependency strength) between components using S, M, W, or N ratings (i.e., strong, medium, and weak, or none respectively). These estimates were converted into numerical values (by assigning a probability of 0.3, 0.2, 0.1, and 0 for the S, M, W, and N respectively). Local and system DSMs, as determined by the average of responses of the surveys, are shown in Figure 7(a-b). A complete explanation of the DSM and ‘dependency’ matrices in Figure 7 is given in (McDaniel 1996).

[15] These rates are obtained by estimating the autonomous completion time for each component and using an exponential decay function.

6.3 Base Case Analyses

For the base case, the largest magnitude eigenvalue of B is 0.9943. Because this eigenvalue is so close to 1, this means that the system is stable, under the above operating conditions, and converges very slowly (see Theorem 2). By inspecting the eigenvector corresponding to the largest magnitude eigenvalue of B, we observe that the magnitudes (in descending order) of the elements are as shown in Figure 8.

Element	S_6	S_3	S_2	H_6	L_6	S_4	S_9	S_7	L_3	L_2	H_3	L_9	S_8	L_{10}	H_2
Magnitude	0.925	0.227	0.158	0.141	0.131	0.098	0.078	0.055	0.044	0.043	0.035	0.033	0.026	0.025	0.024
Cumulative Work	388.2	98.7	68.8	27.8	79.2	43.6	35	22.9	25.9	24.3	7.1	18.2	10.8	12.9	4.9
Element	L_4	S_5	S_{10}	H_4	H_9	L_7	H_7	L_1	L_8	H_8	H_5	H_{10}	L_5	S_1	H_1
Magnitude	0.022	0.022	0.018	0.015	0.012	0.009	0.009	0.007	0.007	0.004	0.003	0.003	~ 0	~ 0	~ 0
Cumulative Work	13.2	8.9	7.2	3.1	2.5	5.7	1.6	3.9	4.3	0.7	0.6	0.5	0.7	~ 0	~ 0

Figure 8: Eigenvector and Corresponding Total Work

The interpretation of the ranking, in Figure 8, is that the larger the magnitude of an element in this eigenvector, the more strongly the element contributes to the slow convergence of this mode of the design process. Thus, the ranking of the eigenvectors gives useful information for identifying the structure of the total work vector. This interpretation is supported by examining the cumulative work, which is obtained by simulating the design process for 52 weeks, as shown in Figure 8.[16] We see that the cumulative work associated with the local 'instrument panel' (i.e., L_6) is more than the work done on other local tasks. This is primarily due to the large work associated with the system 'instrument panel' (see the cumulative work of S_6) and the long information delay (T = 6) between local and system task execution. This phenomenon can be seen by examining the specific traces for individual local components as shown in Figure 9(a). As can be seen, the instrument panel has the largest number of open design issues at every point of time. Also, the oscillatory changes in design status induced by new information contained in scan transmittals are apparent. Finally, we observe that even in the complete absence of external changes, the appearance design process is not completed on time. Design rework and oscillatory

[16] For instance, by comparing the local tasks we see that, in all cases, the largest terms in the total work vector are also the largest terms in the largest eigenvector. In our case, the second largest eigenvalue is much smaller than the largest eigenvalue; thus, the second mode does not contribute significantly to the total work.

behavior in the process result from the decomposed process structure and product architecture, and can never be eliminated from the appearance design process. We conclude that the appearance process must be redesigned to speed up convergence and mitigate churn.

6.4 Mitigation Scenarios

Recall that the development process is stable, under the base operating conditions, but converges slowly. McDaniel (1996) reported that several mitigation strategies were implemented by the engineering and styling teams in order to speed up the rate of design progression needed to meet the required completion date. The analysis developed in this paper provides insight regarding means for achieving stability for a diverging process or speeding up convergence for a slowly converging process. In particular, three types of mitigation strategies can be applied:

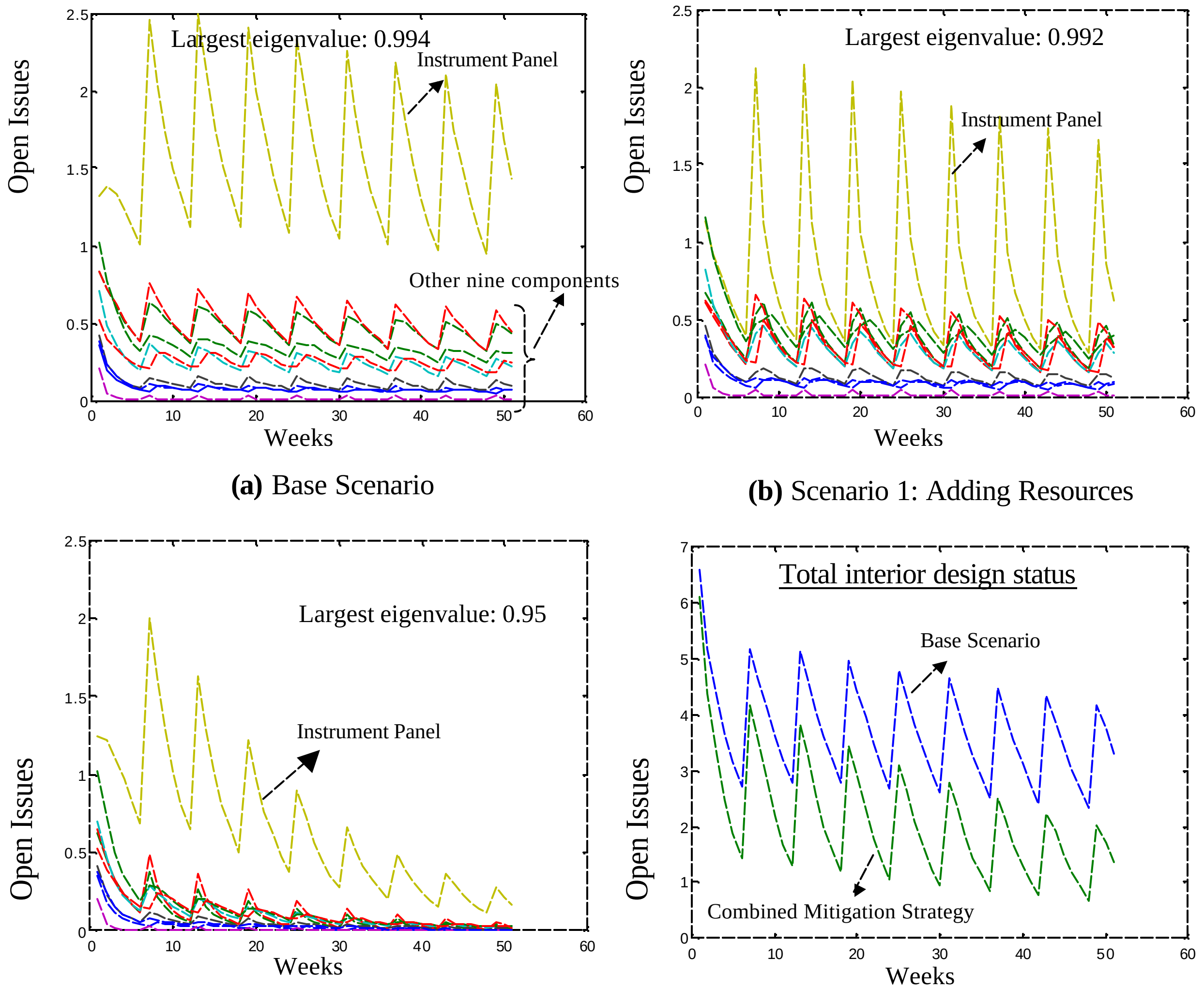
1) Increasing the autonomous design completion rate for each component (i.e., increasing the fraction of work that can be completed in an autonomous manner in every time step);

2) Lessening the pairwise coupling (i.e., dependency strengths) between components;

3) Increasing the frequency with which design information is transmitted from the industrial design to the engineering process (i.e., reducing the information delay T).

The first strategy can be implemented, for instance, by applying resources (work efforts) above the normalized base-case level, which will result in increased progress being made on the independent, autonomous components. The extra resources may be obtained through design technology, personnel training, overtime, skill level, and other determinants of design productivity. The second and third strategies can be accomplished, for instance, by using the knowledge of the inter-component coupling as an aid to making co-location on teaming arrangements (McCord and Eppinger 1993), or by using a variety of formal and informal mechanisms to facilitate the management of design information flows (Braha 2001).

Figures 9(b-c) present the effect of the first two mitigation strategies on the behavior of the base-case model. Scenario 1 represents expending 2.5 normalized resource-weeks and scenario 2 represents modifying the engineering coupling structure by eliminating the weak dependencies. In all cases, the increase in total resource expenditure and reduction in the magnitude of the engineering inter-component dependencies are applied to the more 'complex' local components (i.e., center console (L_2), door trim panel (L_3), and instrument panel (L_6), see Figure 8). Figure 9(d) shows the combined effect of these strategies on the total number of open issues.



(a) Base Scenario

(b) Scenario 1: Adding Resources

(c) Scenario 2: Reduced Coupling

(d) Scenarios 1& 2 Combined

Figure 9: The Effect of Mitigation Strategies on the Behavior of the System

Delays in information flows (introduced by scan transmittal intervals) from the industrial design to the engineering process have a destabilizing effect on system behavior. For example, Figure 10 presents the behavior of the system for various information delays. As can be seen, increasing the information delay results in more extreme churning behavior. Moreover, even though all scenarios are converging, the increased churning behavior leads to slower convergence rates. Indeed, by inspecting the convergence rate (i.e., largest magnitude eigenvalue[17] of the matrix B) of the appearance design process, for various delays between consecutive information releases, we observe that convergence slows *monotonically* for longer delays. To illustrate the economic cost of churn, we inspect the amount of total work in the system over the "convergence" period (i.e., the time required to complete 99% of the initial total work). We see that the work associated with the information delay $T = 6$ is about 10% more than the total work associated with the delay $T = 1$.

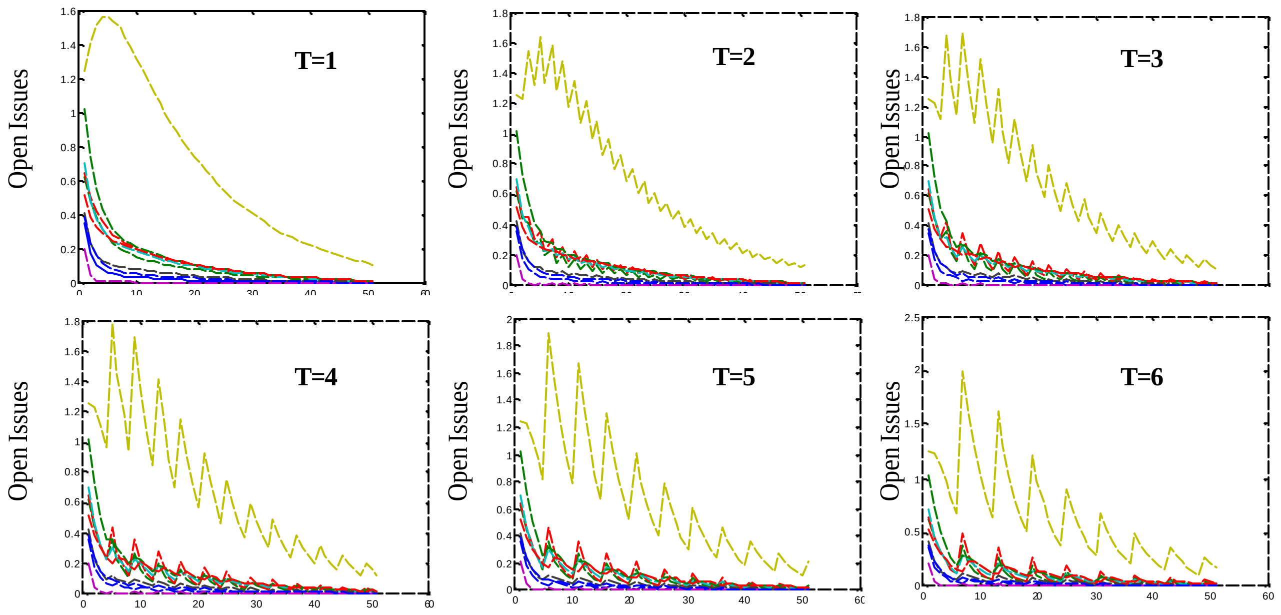


Figure 10. The Effect of Delay on the Churning Behavior

We also notice in Figure 8 that the accumulation of ongoing changes in the industrial design group related to the local 'instrument panel' (see the cumulative work of H_6) is larger than the magnitudes of other elements. Thus, it may be possible to reduce the impact of the accumulated design information by using *differential delays* among components; that is, by increasing the frequency with which design information is transmitted from the industrial design to the local components that *have the most destabilizing effect* on total system performance. For instance, consider the scenario where the industrial design team provides updates to the local engineering tasks L_2, L_3, and L_6 at shorter periodic intervals of $T_1 < 6$ weeks (while maintaining the delay for the others at $T_2 = 6$ weeks). According to the multiple local DSM model of Section 5, the local DSM is now partitioned into two local teams, DSM$_1$= $\{L_2,\ L_3,\ L_6\}$ and DSM$_2$= $\{L_1,$

[17] Recall that the *larger* the eigenvalue the *slower* the system's convergence rate.

L_4, L_5, L_7, L_8, L_9, L_{10} }. By applying the results[18] of Section 5, Figure 11 plots the convergence rate (i.e., largest magnitude eigenvalue of the matrix B) for the base scenario under **1)** five differential information release policies, $T_1 = j$ and $T_2 = 6$ for j =1, 2, …, 5, and **2)** overall information release policy $T = j$ for j =1, 2, …, 5. As can be seen, the differential delay policy *consistently* achieves better "performance" (larger convergence rate) than the corresponding uniform policy; that is, the differential delay policy with $T_1 = j$ and $T_2 = 6$ achieves better "performance" than the uniform information release policy with delay $T = j$ for every j =1, 2, …, 5.[19]

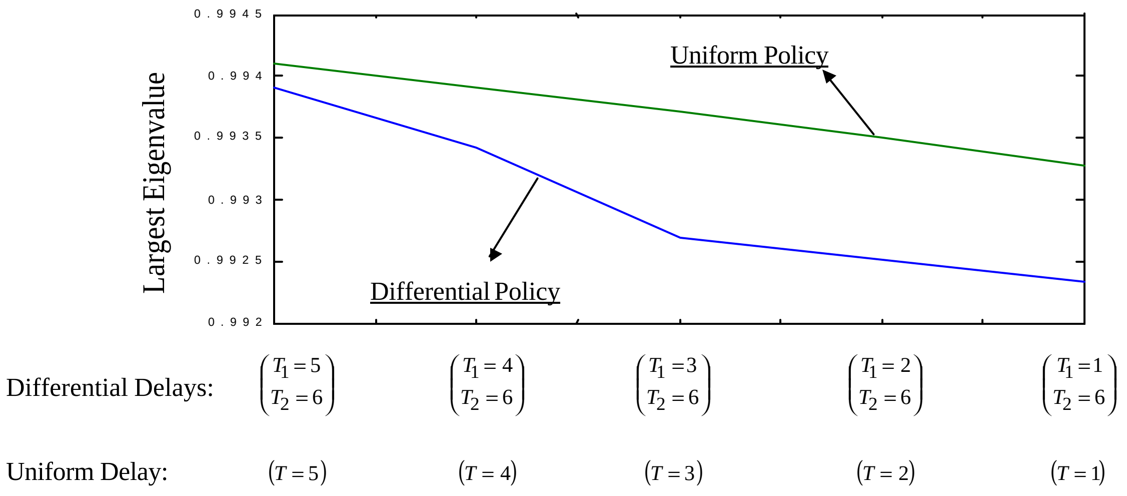


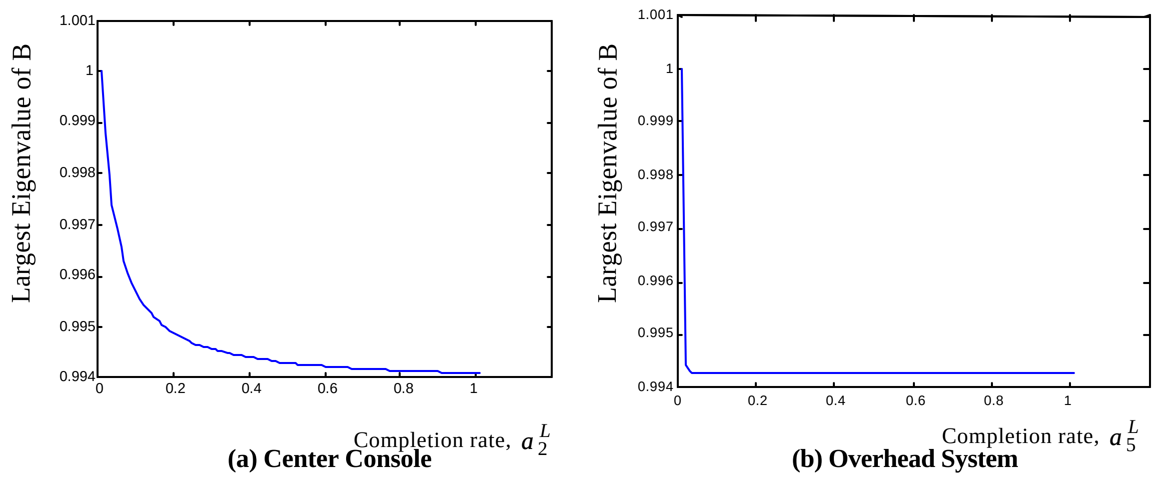
Figure 11. The Effect of Delay Policy on the Largest Eigenvalue

6.5 Sensitivity Analysis

The model developed in this paper enables us to perform sensitivity analysis. For example, let a_2^L be the autonomous local center console completion rate (corresponding to the element in row two, column two in the local DSM). Assume that the other elements in the local DSM are set to their values as specified in Figure 7. Figure 12(a) plots the *largest* magnitude eigenvalue of B against a_2^L. As can be seen, any value of $a_2^L > 0$ will have a stabilizing effect on the system behavior (see Theorem 2). Similar plot for the local overhead system (Figure 12b) suggests that the convergence rate is *completely* in*sensitive* to its autonomous completion rate as long as it is

[18] According to Theorem 4, the fundamental period of the monodromy matrix is 30=lcm(5, 6).
[19] The advantage of reducing the information delay should be weighed against the possibly additional resources and undesirable side effects. Exploration of these tradeoffs is beyond the scope of this paper.

greater than 0.05. Consequently, any increase in total resource expenditure for a bottleneck component (such as the center console) will be effective in improving the system performance.



(a) Center Console **(b) Overhead System**

Figure 12. The Effect of Autonomous Completion Rate on Convergence

7. Discussion and Conclusion

The model described in this paper provides managers with operational insights that explicitly capture the fundamental characteristics of a development process. It allows managers to experiment with several "what-if scenarios" in order to explore and compare the effects of subsequent managerial actions of improvement. However, a basic revelation of the model is that design churn is an unavoidable phenomenon and a fundamental property of a decomposed development process where the product or process architecture dictates delays in feedback information amongst the development groups. Consequently, the most significant insight this model brings to managers is to avoid making myopic decisions based on the observance of churn. The fluctuation in development progress cannot be avoided, but can be managed once managers understand its sources. Our model reveals several main sources of churn:

a. Interdependency of process or product structure is apparent when the development occurs within a monolithic group; however, it is usually hidden, ignored, or forgotten once the process is decomposed into multiple groups. Fully anticipating, understanding, and accommodating this structure, can explain why the tasks seem difficult, frustrating and prone to change.

b. Concurrency of local and system execution may help in expediting the development process; however, careful timing and magnitude of feedback is necessary to provide development groups with enough time, between feedbacks, to understand and react to these feedback flows. If these flows are not carefully planned, they might drive the process unstable by generating more rework than the development teams can handle.

c. Feedback delays are an important factor in developing a clear understanding of the development process and play a major role in determining the system stability. In combination with the interdependency structure, delays are the main reason why development problems (issues) believed to be solved (closed) tend to re-appear (reopen) at later stages of development.

While exposing churn as a fundamental property of a decomposed development process, our model also provides managers with three mitigation strategies to combat design process churn, divergence, or slow convergence. These strategies are:

1. Timing-based strategies: These strategies advocate the minimization of delays for specific tasks that contribute the most to the slow convergence of the development process. Our model provides a quantitative approach to identify these bottleneck tasks. Once identified, strategies for reducing the time delays for these tasks should be implemented. These include the early release of preliminary information and divisive overlapping (Krishnan et al. 1997). Our illustration shows that acceleration of the synchronization frequency for all tasks may not be as effective as accelerating, by the same amount, the synchronization frequency for the bottleneck tasks.
2. Resource-based Strategies: This strategy allows local and system teams to work faster (as captured by the diagonal elements of both W^L and W^S) by incorporating more resources. Our illustration shows that working faster on all the tasks simultaneously may not be as effective as allocating the same amount of resources only to the bottleneck tasks.
3. Rework-based strategies: This strategy suggests that local groups ignore low priority local or system feedback (as captured by the low rework fractions in $W^{L_iL_j}$ or $W^{S_iL_j}$). A similar strategy is to reduce the values of $W^{L_iL_j}$ or $W^{S_iL_j}$ by requiring that local or system teams

not produce much feedback to local groups. Both these strategies benefit from a modular architecture.

All the above strategies are effective in mitigating the three sources of churn (i.e., interdependency, concurrency, and feedback delays) either individually or collectively. We have demonstrated the impact of these strategies using the automotive appearance design process.

Several extensions to our model are possible. First, cost elements associated with the information release and information processing activities may be incorporated within our model. This may result in a convex formulation that allows for the optimal determination of the information delay T (e.g., Thomke and Bell, 2001). Second, except for the local and system autonomous rates of completion, our model does not explicitly account for resource allocation policies. Thus, explicitly incorporating resource allocation as a decision variable may lead to the discovery of better resource allocation policies in the context of decomposed development processes. Finally, the linearity assumption in our model can be relaxed, and non-linear formulations may be developed. For example, our model can be modified by incorporating time-varying rework fractions, which are reduced with time as the development process unfolds.

We have develped a model for a development process based on decomposing it into two groups: local and system. The model incorporates two types of information flows: 1) information flows that reflect internal rework within local and system groups, possibly generating internal rework; and 2) information flows that reflect status updates from local to system tasks and feedback from system to local tasks. These information flows influence both 'fundamental' and extrinsic' churn and determine the shape and rate of convergence of the development process.

References

Adler, P., Mandelbaum, A., Nguyen, V., E. Schwerer. 1995. From Project to Process Management: Empirically-Based Framework for Analyzing Product Development Time. *Management Science* 41(3) 458-483.

Ahmadi, R., R. H. Wang. 1999. Managing Development Risk in Product Design Processes. *Operations Research* 47(2) 235-246.

Bohn, R. 2000. Stop Fighting Fires. 2000. *Harvard Business Review* July-August 83-91.

Braha, D. 2001. *Data Mining for Design and Manufacturing*. Kluwer Academic Publishers: Boston, MA.

Browning, T., Deyst, J., Eppinger, S.D., D.E. Whitney. 2000. Complex System Product Development: Adding Value by Creating Information and Reducing Risk. *Proceedings of the Tenth Annual International Symposium of INCOSE* 581-589.

Browning, T., S.D. Eppinger. 1998. A Model for Development Project Cost and Schedule Planning. M.I.T. Sloan School of Management, Cambridge, MA. Working Paper No. 4050.

Clark, K., T. Fujimoto. 1991. *Product Development Performance: Strategy, Organization, and Management in the World Auto Industry*. Harvard Business School Press, Boston.

Cusumano, M., R. Selby. 1995. *Microsoft Secrets*. Free Press: New York, NY.

DeCarlo, R.A. 1989. *Linear Systems: A State Variable Approach*. New Jersey: Prentice Hall, Englewood Cliffs.

Eppinger, S. D. 2001. Innovation at the Speed of Information. *Harvard Business Review* 79(1) 149-158.

Eppinger, S. D., Whitney, D. E., Smith, R., D. Gebala. 1994. A Model-based Method for Organizing Tasks in Product Development. *Research in Engineering Design* 6(1) 1-13.

Eppinger, S. D. 1997. A Planning Method for Integration of Large-Scale Engineering Systems. *International Conference on Engineering Design*, Tampere, Finland 199-204.

Ford, David and John Sterman, “Overcoming the 90% Syndrome: Iteration Management in Concurrent Development Projects,” Working Paper, Texas A&M, 1999.

Joglekar, N.R., 2001. Data collected at Factory Mutual Insurance Company (Norwood, MA).

Joglekar, N.R., Yassine, A., Eppinger, S.D., D.E.Whitney. 2001. Performance of Coupled Product Development Activities with a Deadline. *Management Science* 47 (12).

Joglekar, N., A. Yassine. 2001. Management of Information Technology Driven Product Development Processes. *New Directions in Supply-Chain Management: Technology, Strategy, and Implementation*. Ram Ganeshan and Tonya Boone (eds.), AMACOM books.

Krishnan, V., Eppinger, S.D., D.E. Whitney. 1997. A Model-Based Framework to Overlap Product Development Activities. *Management Science*, 43 (4) 437-451.

Kuchment, P. 1993. Floquet Theory for Partial Differential Equations. *Operator Theory, Advances and Applications*. Springer Verlag, New York.

Lee, H., Padmanabhan, V., S. Whang. 1997. Information Distortion in a Supply Chain: The Bullwhip Effect. *Management Science* 43(4) 546-558.

Loch, C., T. Christian. 1999. Accelerating the Process of Engineering Change Orders: Capacity and Congestion Effects. *Journal of Product Innovation Management* 16(2) 145-159.

Mar, C. 1999. *Process Improvement Applied to Product Development*. MIT MS thesis.

Marcus, M., H. Minc. 1964. *A Survey of Matrix Theory and Matrix Inequalities*. Allyn and Bacon, Boston, MA.

McCord, K. R., S.D. Eppinger. 1993. Managing the Integration Problem in Concurrent Engineering. Sloan School of Management Working Paper #3594-93-MSA.

McDaniel, C.D. 1996. *A Linear Systems Framework for Analyzing the Automotive Appearance Design Process.* Master's Thesis (Mgmt./EE), MIT, Cambridge, MA.

Mihm, J., Loch, C., A. Huchzermeier. 2001. Modeling the Problem Solving Dynamics in Complex Engineering Projects. INSEAD Working Paper.

Richards, J. 1983. *Analysis of Periodically Time Varying Systems*. New York: Springer-Verlag.

Repenning, N., Gocalves, P., L. Black. 2001. Past the Tipping Point: The Persistence of Firefighting in Product Development. *California Management Review* 43(4) 44-63.

Simon, H. 1996. *The Sciences of the Artificial*. MIT Press: Cambridge, Massachusetts.

Smith, R. P., S.D. Eppinger. 1997. Identifying Controlling Features of Engineering Design Iteration. *Management Science* 43(3) 276-293.

Sobek, D., Ward, A., J. Liker. 1999. Toyota's Principles of Set-Based Concurrent Engineering. *Sloan Management Review* 40(2) 67-83.

Sosa, M.E., Eppinger, S.D., C.M. Rowles. 2000. Designing Modular and Integrative Systems. ASME Conference on Design Theory and Methodology, Baltimore, MD.

Strang, G. 1980. *Linear Algebra and Its Applications*. Harcourt Brace Jovanovich: New York.

Sullivan, K. J., Griswold, W. G., Cai, Y., B. Hallen. 2001. The Structure and Value of Modularity in Software Design. Proceedings of the Joint International Conference on Software Engineering and ACM SIGSOFT Symposium on the Foundations of Software Engineering, Vienna.

Thomke, S., D. Bell. 2001. Sequential Testing in Product Development. *Management Science* 47(2) 308-323.

Wheelwright, S., K. Clark. 1992. *Revolutionizing Product Development*. Free Press: New York.

Appendix A: Specifications of Work Transformation Matrices (W^{L}, W^{S}, W^{LS}, W^{SH}, W^{SL})

The specification of the work transformation matrices is based on the assumption that *only work that is done in the previous period is considered to create rework as a normal course of operation*. Let $\Omega^{\mathrm{L}} = \left(a_{ij}^{\mathrm{L}}\right)$ be the local DSM. The work completion coefficient $a_{ii}^{\mathrm{L}} \equiv a_i^{\mathrm{L}}$ is the local autonomous completion rate for task i at each iteration step. The coupling coefficient a_{ij}^{L} (for $i \neq j$) is the amount of rework created for local task i per unit of work **done** on local task j. Consequently, the elements of the work transformation matrix W^{L} become $\mathrm{w}_{ii}^{\mathrm{L}} = 1 - a_i^{\mathrm{L}}$ and $\mathrm{w}_{ij}^{\mathrm{L}} = a_{ij}^{\mathrm{L}} a_{jj}^{\mathrm{L}}$ (for $i \neq j$). The system DSM Ω^{S} and work transformation matrix W^{S} are defined similarly.

The interaction between the local and system teams is captured by the *inter-component dependency* matrices $\Omega^{\mathrm{LS}} = \left(a_{ij}^{\mathrm{LS}}\right)$ and $\Omega^{\mathrm{SL}} = \left(a_{ij}^{\mathrm{SL}}\right)$. The coupling coefficient a_{ij}^{LS} is the amount of rework created for system task i per unit of work **done** on local task j. Similarly, the coupling coefficient a_{ij}^{SL} is the amount of rework created for local task i per unit of work **done** on system task j. Consequently, the elements of the work transformation matrix W^{LS} are $\mathrm{w}_{ij}^{\mathrm{LS}} = a_{ij}^{\mathrm{LS}} a_{jj}^{\mathrm{L}}$. The matrix W^{SL} is defined as $\mathrm{w}_{ij}^{\mathrm{SL}} = a_{ij}^{\mathrm{SL}} a_{jj}^{\mathrm{L}}$. Finally, the "holding" matrix W^{SH} is defined as $\mathrm{W}^{\mathrm{SH}} = \mathrm{W}^{\mathrm{SL}}$.

Appendix B

Proof of Lemma 1, Theorem 1 and Corollary 1

See Richards (1983).

Proof of Theorem 2.

From Theorem 1, $x(k)$ is a solution of the linear periodic system described by equation (7) if and only if $y(k) = P^{-1}(k)x(k)$ is a solution of the linear autonomous system described by equation (8). The matrix $P(k)$ is nonsingular and periodic. Thus, the stability of the linear periodic system (7) is equivalent to the stability of the associated linear autonomous system (8). Consequently,

1) If the largest magnitude eigenvalue of B (i.e., the largest magnitude Floquet exponent) is less than 1, then every solution $x(k)$ of (7) satisfies $\lim_{k\to\infty} x(k) = 0$;

2) If the largest magnitude eigenvalue of B is less than or equal to 1, then every solution $y(k)$ of (8) remains bounded for $k \geq 0$.

3) (Only if part). Assume that the largest magnitude eigenvalue of B is greater than 1. Then there is a solution $y(k)$ of (8) such that $\lim_{k\to\infty} x(k) = \infty$, and the zero solution is unstable. ν

Corollary 2: Since the eigenvalues of B are the T^{th} roots of the eigenvalues of the monodromy matrix C, corollary 2 immediately follows. ν

Proof of Theorem 3

From Theorem 1, the general solution $x(k)$ of (7) may be written as $x(k) = P(k)y(k)$ where $y(k)$ is the general solution of the linear autonomous system (8). For the linear autonomous system (8), it can be verified that the general solution can be written as $y(k) = B^k S_B g$, where S_B is the eigenvector matrix of B and $g = (g_1, g_2, ..., g_n)' \in R^n$. The powers of B can be found by $B^k = S_B \Lambda_B^k S_B^{-1}$, where Λ_B is a diagonal matrix of the eigenvalues of B. Consequently,

$$y(k) = B^k S_B c = S_B \Lambda_B^k g =$$

$$[x_1, x_2, ..., x_n] \begin{bmatrix} l_1^k & & & 0 \\ & l_2^k & & \\ & & \ddots & \\ 0 & & & l_n^k \end{bmatrix} \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_n \end{bmatrix} = [l_1^k x_1, l_2^k x_2, \cdots, l_n^k x_n] \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_n \end{bmatrix}$$

Where $[x_1, x_2, ..., x_n]$ is the eigenvector matrix for *B*.

Hence the general solution $x(k)$ of (7) may be given by

$$x(k) = P(k)y(k)$$

$$x(k) = P(k)y(k) = [l_1^k P(k)x_1, l_2^k P(k)x_2, \cdots, l_n^k P(k)x_n] \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_n \end{bmatrix} \qquad \text{(B.1)}$$

From equation (B.1) we see that the general solution $x(k)$ of (7) may be given by $x(k) = \Phi(k)g$, i.e., each of the column vectors of $\Phi(k)$ is a nontrivial solution of (7). Let $\hat{x}(k) = l_i^k P(k)x_i$ be such a nontrivial solution. We have

$$\hat{x}(k+T) = l_i^{k+T} P(k+T) x_i = l_i^T \, l_i^k P(k) x = l_i^T \, \hat{x}(k) \tag{B.2}$$

Notice that l_i^k is an eigenvalue of the monodromy matrix C, i.e. l_i^T is a Floquet multiplier of the linear periodic system (B.1). Thus, there exists a solution $\hat{x}(k)$ of the linear periodic system (B.1) such that $\hat{x}(k+T) = l_i^T \hat{x}(k)$, and this is the reason we call l_i^T a multiplier. Now,

(i) If the matrix C has an eigenvalue equal to 1, then $l_i^T = 1$ and from (B.2) there exists a periodic solution of period T. ν

(ii) If the matrix C has an eigenvalue equal to –1, then $l_i^T = -1$ and from (B.2) there exists a periodic solution of period $2T$. ν

(iii) Let the local and system work transformation matrices will be coupled and non-negative. Consequently, the monodromy matrix C will be coupled and non-negative. Thus, in many applications, $C^L > 0$ for some power L (i.e., C is primitive) for L>0. By the Perron-Frobenius theorems for primitive matrices one of its eigenvalues l_C^* is positive real and strictly greater (in absolute value) than all other eigenvalues, and there is a positive eigenvector corresponding to that eigenvalue. Since, $l_B^* = \sqrt[T]{l_C^*}$, according to Eq. (B.1), the largest magnitude eigenvalue of B is also positive real, and there is a positive eigenvector corresponding to that eigenvalue. Therefore, the long term behavior of the system has the form

$$x(k) \sim c_1 (l_B^*)^k P(k) x \tag{B.3}$$

If the largest eigenvalue of C equal to 1, then it follows from (B.3) that the long-term behavior of the system is periodic of period T. ν

Proof of Theorem 4

Since T is the least common multiple of $T_1, T_2, ..., T_m$, it follows that there are integers $a_1, a_2, \ldots, a_m$ such that $T = a_i T_i$ for $1 \le i \le m$. Let $k \ge 0$ be any time point. Assume that at time point k the system team provides updates **only** to the local teams $i_1, i_2, \cdots, i_j$. From the information release policy it follows that there are integers $b_1, b_2, \ldots, b_n$ such that $k = b_{i_\ell} T_{i_\ell}$ for $i_\ell \in \{i_1, i_2, \cdots, i_j\}$ and $k = b_{i_\ell} T_{i_\ell} + e_{i_\ell}$ for $i_\ell \notin \{i_1, i_2, \cdots, i_j\}$ where $0 < e_{i_\ell} < T_{i_\ell}$. Consider time point $k+T$.

For $i_\ell \in \{i_1, i_2, \cdots, i_j\}$, $\quad k + T = b_{i_\ell} T_{i_\ell} + a_{i_\ell} T_{i_\ell} = (b_{i_\ell} + a_{i_\ell}) T_{i_\ell}$

For $i_\ell \notin \{i_1, i_2, \cdots, i_j\}$, $\quad k + T = b_{i_\ell} T_{i_\ell} + e_{i_\ell} + a_{i_\ell} T_{i_\ell} = (b_{i_\ell} + a_{i_\ell}) T_{i_\ell} + e_{i_\ell}$

Thus, we conclude that at time point $k+T$ the system team will provide updates **only** to the local teams $i_1, i_2, \cdots, i_j$. Consequently, the fundamental period of the linear system (12) is T. ν