The Application of System Dynamics to Concurrent Engineering

David N. Ford, David.Ford@ifi.uib.no

Introduction. The design manager appeared simultaneously exhausted, depressed and frustrated. The project manager had been forced to postpone the deadline of the development project again. The delay wiped out the benefits of the concurrent engineering approach that was intended to bring their product to the market before their competitors. Even though design changes had precipitated the slip, the design manager knew he had not been the real cause.

Not long after design work started his estimated dates to complete and release the design packages had exceeded the deadlines he had negotiated with the managers of the project's other phases. He was tempted to ask for more time, but he remembered the project manager's exhortation for proactive management to keep the project on its aggressive schedule. So he had initiated overtime and started recruiting more designers.

But during the time needed to find and train the new designers, the existing staff spent more time training and less time designing and got “burned out” trying to maintain the scheduled rate of progress. This created a big backlog of uncheck work and unresolved design changes, and extended his expected completion dates even further. To meet the “deliver or else” deadlines to release designs to the prototyping and production planning phases, he had reluctantly released designs without complete testing and hoped for the best. But his reprieve from schedule pressure was only been temporary. As the prototype and production engineers tried to use the designs they discovered errors. His “Changes to be Coordinated and Resolved” file had grown much faster than his staff could address it. The estimated completion dates stretched out even further into the future. When the project deadline neared and the project manager intensified his questioning at the weekly managers meeting, the prototype and production planning managers mentioned waiting for these changes as the cause of the delays in their own phases. It was then that the project manager had been forced to slip the schedule, with a menacing glare at the confused design manager.

This hypothetical but typical dilemma illustrates some of the challenges to successfully implementing concurrent engineering in development projects. The diversity of actions and agents, and the relationships that link them is an important feature of concurrent engineering projects. The story above includes development operations (e.g., design, quality assurance and coordination), resources (design staff), management (deadlines and policies), and the behavior of the participants (shifting blame). The diversity and tight coupling of project components make understanding and improving concurrent engineering impossible by focusing solely on development operations, or resources, or management, or the behavior of participants. Only by modeling and analyzing development operations, resources, and management, and the behavior of participants and how they interact, can the systems nature of concurrent engineering be understood and improved. Designers and managers of concurrent engineering projects need to understand how different components influence each other in order to design changes that will improve performance. The effective modeling and analysis of concurrent engineering requires an approach that can describe a variety of components and relationships.

One of the challenges of modeling concurrent engineering is that the important causal paths that link design and management to performance pass through several of the diverse subsystems of the project. For example, an explanation of why the project described above was delayed needs its way through schedule targets, estimates and pressure, design operations, design management policies about overtime and staffing, design quality and release policies, concurrency between phases, prototype and production quality assurance operations, interphase coordination and change resolution and project schedule flexibility. Effectively diagnosing the drivers of concurrent engineering requires explicitly modeling the locations and characteristics of the causal paths that describe how project components affect each other.

Structural feedback is a dominant characteristic of the causal paths in concurrent engineering projects. Structural feedback exists when the impacts of a change in a component travel through one or more causal paths and return to influence the component. In the story above, the impacts of the schedule pressure caused the design manager to recruit designers who influenced the effective size of the design staff, and thereby the design backlog and amount of schedule pressure. Structural feedback is inherent in concurrent engineering due to its many strong dependencies. This makes concurrent engineering projects dynamically complex, evolving over time as project participants, respond to conditions, and those conditions respond to attempts to control the project. Policies that improve conditions initially (e.g., releasing unchecked designs) can degrade performance as delayed effects play themselves out. Systems engineers need a dynamic perspective to design and analyze processes, and policies that generate different
behaviors over time.

**The System Dynamics Approach to Concurrent Engineering.**

System dynamics (Forrester 1961) applies a unique perspective and modeling methodology to concurrent engineering. Three central system dynamics concepts for application to concurrent engineering are structural feedback, delays, and participant behavior. System dynamics identifies structural feedback at the operational level that can generate observed or desired patterns of behavior over time. This approach contrasts with an assumption that external forces are the primary causes of behavior. The focus on internal causes of behavior helps identify high leverage points that can be used to improve performance, and unintended side effects that, in turn, can defeat management efforts. An example of an unintended side effect due to the design manager recruiting more designers is the increase in training and resulting decrease in effective design staff size.

System dynamics also focuses on the delays that constrain progress and distort information. The development operations and management of concurrent engineering projects generate many delays. With structural feedback, these delays cause the long-term impacts of some policies to be very different from their short-term impacts. For example, the short term effect of releasing inadequately checked designs was to reduce schedule pressure, but the delayed effect was to increase design backlog and increase schedule pressure. A feedback perspective and explicit modeling of delays is particularly valuable in modeling management policies and the behavior of participants. Management policies describe the use of system conditions and decision rules to generate actions designed to control behavior. Participant behavior, such as the fatigue experienced by the design staff, strongly influences the effectiveness of policies. The mental models and limits of cognition of both policies and participant behavior perturbs the process and therefore must be described with nonlinear relationships.

Applying system dynamics to concurrent engineering requires effective communication with managers and developers who provide model information, develop insights, and implement system changes based on modeling and analysis. In many cases, these critical participants in the modeling and analysis are unable to understand or use mathematical models. To address this constraint, system dynamics has developed modeling tools and methods that are effective with non-technical participants, as well as those experienced and comfortable with formal models. A typical system dynamics modeling project uses causal loop diagramming (Richardson and Pugh, 1981) to describe the system operations related to the problem and current management. Causal loop diagrams describe the feedback structure of a system and locate major delays.

Figure 1 shows a causal loop diagram that maps part of the feedback structure of the story at the beginning of this article. Balancing loop B1 describes how the design manager's initial response to schedule pressure of recruiting designers was intended to reduce the backlog and schedule pressure. But the unintended side effect of increased training requirements described by reinforcing loop R1 defeated this policy due to the use of experienced designers for training and the delay in getting new designers productive. This reduced the effective size of the design staff, thereby increasing the backlog and schedule pressure. Balancing feedback loop B2 describes the design manager's second policy to reduce schedule pressure, releasing unchecked designs, and its initial success by reducing the design backlog. Reinforcing loop R2 describes how the delayed downstream discovery of change requirements increased the design backlog, increasing the schedule pressure instead of decreasing it, and thereby precipitating the slipping of the project deadline.

---

**Legend**

- - - > a causal relationship: change in variable at arrow's tail causes change in variable at arrow's head
+ variable at arrow's head moves in the same direction as the variable at the arrow's tail
- variable at arrow's head moves in the opposite direction as the variable at the arrow's tail
// delay in response of variable at arrow's head to change in variable at arrow's tail

**R** - Reinforcing feedback loop: in isolation generates exponential growth or decay

**B** - Balancing feedback loop: in isolation generates controlling or goal-seeking behavior

**Feedback Loops**

B1 - Staff size adjusts with a delay in response to schedule pressure
R1 - Design training increases backlog and schedule pressure, increasing staff and training
B2 - Release of unchecked designs responds to schedule pressure
R2 - Delayed return of designs increases schedule pressure, increasing release of unchecked designs
Identifying feedback loops, such as the ones described above, is necessary, but not sufficient, to improve concurrent engineering project design and management. System dynamics formalizes feedback models into mathematical descriptions of the flows and accumulations of work, people, and information in concurrent engineering development projects. These sets of differential equations allow computer simulation of the behavior generated by the system as described, and hypotheses about what structures drive behavior to be tested.

The variation in the types of system components and their relationships in system dynamics models requires a variety of tests to develop confidence in the model's ability to simulate realistic behavior for the same reasons that the behavior is generated in real projects. Examples of these tests include the direct comparison of the model structure with the structure of the project, tests of model behavior for realistic responses over a wide range of conditions, and comparisons of simulated and actual project behavior. The model is ready for use as an analysis and design tool when it simulates actual project behavior patterns using the same causal paths as actual projects.

Applying System Dynamics to Concurrent Engineering: An Example. One example of applying system dynamics to concurrent engineering is Ford (the author) and Sterman's (1998b) investigation of the relationship between concurrency and project schedule performance. Our model describes how the initial completion of work, quality assurance, changes and coordination move work through six conditions in individual development phases (Figure 2). Individual phases are linked with concurrency relationships, coordination and performance targets. Management policies, such as the use of unchecked work as the basis for additional development, work release package sizes, resource allocation policies and the flexibility of schedules, budgets and quality goals, are explicitly modeled so that their descriptions can be discussed with and verified by practitioners, and so that the impacts of policies can be tested.

We calibrated our model to a medium sized semiconductor development project. Data collection included running workshops to elicit and articulate the expert but tacit knowledge about concurrency relationships within and between development phases held by the development engineers and managers. The developers found these workshops useful because they were able to share and compare mental models in a forum which facilitated learning by investigating their underlying assumptions (Ford and Sterman, 1998a).

By explicitly modeling the concurrency relationships within and between development phases, and the constraints on progress imposed by development operations and the resources applied to them, we could analyze the conditions under which each part of the project drove or constrained progress. We used the model to study how unintended side effects of increasing concurrency can defeat efforts to shorten cycle time with concurrent engineering (Ford and Sterman, 1999). We did this by focusing on the causes of the "90% syndrome," a common form of schedule failure in concurrent development. We found that increasing concurrency and the common propensity of developers and managers to conceal required changes from other development team members aggravates the syndrome and degrades schedule performance through feedback loops that control iteration between project phases.

Based on this analysis, we suggested iteration management strategies that may improve concurrent engineering implementation.

System dynamics models can complement other systems engineering approaches to concurrent engineering. Consider the potential synergy between Ford and Sterman's system dynamics model and Melsa and Smith's (1998) approach to defect control for concurrent software development. Melsa and Smith describe a set of policies for controlling the number and release of defects to subsequent development phases and customers. For example, one policy uses system conditions (e.g., the number of unresolved defects) to allocate resources to development operations. They initially test their approach by applying it to a single development project. Despite their success many questions remain.
before their approach can be fully understood and applied to other projects, which may differ in important ways from their case study. "How sensitive is the project's performance to specific project characteristics such as size and complexity? Which policies and combinations of policies increase performance most under what conditions?" More importantly for developing widely applicable insights from their work, "How can we model and test the recommended or alternative policies in the iterative environment that characterizes concurrent engineering and under a variety of conditions?"

A system dynamics approach can be used to develop just such a tool. For example, Ford and Sterman's model includes all the important aspects of concurrent engineering identified by Melsa and Smith. In addition, the majority of the variables proposed by Melsa and Smith as useful for practicing managers are included in the formal mathematical model. The simulation model facilitates the discovery of previously unknown defects, which can be explicitly modeled with the known defects. The model also allows experimentation and exploration of concurrent engineering project structures and policies that could not be done with actual projects due to costs, feasibility and other constraints. System dynamics based project simulation models can reflect the features and issues addressed by many other systems engineering methods, and provide valuable opportunities for analysis and design not available by other means.

Conclusions. System dynamics is particularly effective for modeling and analyzing how the interactions of structures and policies impact project performance in concurrent engineering projects. This is partially due to its focus on behavior generated by structural feedback, multiple time perspectives and the ability to equally model development operations, resources, management, participant behavior and their interactions at an operational level. The method's flexibility allows the rigorous modeling of interactions among the variety of subsystems which form concurrent engineering projects. This makes system dynamics particularly well suited for systems engineering, which addresses the design and management of the interfaces where those interactions occur.

The flexibility of system dynamics provides both great strength as a modeling methodology, and special challenges in developing confidence in the model's ability to reflect projects accurately, and rigorously analyze and describe how project structures drive behavior. System dynamics has been applied to investigate concurrent engineering projects in shipbuilding, consumer and industrial electronics, semiconductor development and, most recently, the construction industry. The construction industry in particular, where concurrent engineering has been practiced for over two decades in fast-track projects, provides unique opportunities to discover the fundamental drivers of behavior in concurrent engineering projects, and how to manipulate them to improve performance.

System dynamics provides a perspective and modeling method that can be used to investigate the interactions of concurrent engineering development operations, resources, management, and participant behavior at the operational level. Its ability to elucidate and explain how the complex structures of concurrent engineering projects drive performance makes it a valuable systems engineering tool.

Biography:
David Ford is an associate professor in the system dynamics program at the Department of Information Science, University of Bergen, Norway. His research investigates the design and management of development projects.

References
INSIGHT SPECIAL FEATURE

Insights into Process Modeling and Management for IPPD