Ford, D. N., & Sterman, J. D. (1998). Dynamic Modeling of Product Development Processes. *System Dynamics Review*, 14(1), 31-68.

Dynamic modeling of product development processes

David N. Ford[a]* and John D. Sterman[b]

David N. Ford is an associate professor in the system dynamics program at the University of Bergen in Norway. He earned his PhD from the Massachusetts Institute of Technology, where he conducted research on the dynamics of development processes. His current research interests include product development management, coordination and policy development.

John Sterman is the J. Spencer Standish Professor of Management at the Massachusetts Institute of Technology (MIT) Sloan School of Management, and Director of the MIT System Dynamics Group.

Abstract

Successful development projects are critical to success in many industries. To improve project performance managers must understand the dynamic concurrence relationships that constrain the sequencing of tasks as well as the effects of and interactions with resources (such as labor), project scope and targets (such as delivery dates). This article describes a multiple-phase project model which explicitly models process, resources, scope, and targets. The model explicitly portrays iteration, four distinct development activities and available work constraints to describe development processes. The model is calibrated to a semiconductor chip development project. Impacts of the dynamics of development process structures on research and practice are discussed. © 1998 John Wiley & Sons, Ltd.

Syst. Dyn. Rev. **14**, 31–68, (1998)

Introduction

Developing products faster, better and cheaper than competitors has become critical to success in many markets, whether the product is an office building, software package, or computer chip. This has made the performance of product development projects an increasingly important area of competitive advantage. In response to these pressures many industries have shifted from a sequential, functional development paradigm to a concurrent, team-based paradigm. Increasing concurrence and cross-functional development also dramatically increases the dynamic complexity of product development (Smith and Eppinger 1997; Ford 1996; Wetherbe 1995; Osborne 1993). However, the mental models used by developers and managers to evaluate, estimate, and manage projects have not generally improved to include dynamic influences on performance. The resulting lack of understanding (Diehl and Sterman 1995; Sterman 1994; Paich and Sterman 1993; Rechtin 1991) and inadequate decision heuristics (Kleinmuntz 1993) have contributed to the frequently cited poor management of development projects (Construction Industry Institute 1990; Womack, Jones and Roos 1990; Dertouzos, Lester and Solow 1989; Davis and Ledbetter 1987; Pugh 1984; Abdel-Hamid 1984; Brooks 1975).

Many aspects of projects influence performance including the development process, resources, project scope, and targets. A project's development process

[a]Department of Information Sciences, University of Bergen, N-5020 Bergen, Norway.
[b]Sloan School of Management, Massachusetts Institute of Technology, 50 Memorial Drive, E53-351, Cambridge, MA 02142 U.S.A.
*Corresponding Author.

System Dynamics Review Vol. 14, No. 1, (Spring 1998): 31–68
©1998 John Wiley & Sons, Ltd. CCC 0883-7066/98/010031–38 $17.50

Received February 1997
Accepted November 1997

describes the flows of work among development phases and the completion of development tasks within each phase. The characteristics of a development process describe the relative difficulty of development activities, concurrence relations among activities, delays within processes such as change discovery, and iteration within and between phases. The quantity and effectiveness of resources constrain the rate at which different development activities are performed by limiting development capacity. A project's scope helps define completion by describing the amount of work required to complete each phase of development. Targets describe acceptable levels of performance and project priorities. The development process, resources, scope, and targets of a project interact in complex ways to drive project performance.

Traditional project management models based on the Critical Path Method (CPM) and PERT (Moder, Phillips and Davis 1983; Halpin and Woodhead 1980) describe process, resources, targets and scope in a static fashion with activity duration estimates and precedence relationships describing the network of development activities. These descriptions are used to predict the effects of process, resources, targets and scope on performance (primarily schedule). These methods are limited by their use of an indirect project measure (time) and by bundling the characteristics of and relationships among scope, resources, and processes in each activity into a single duration estimate. They also tend to ignore iteration or require that iteration be implicitly incorporated into duration estimates and precedence relationships. For example, if a product definition that requires a change is released to designers who then discover the change after design work has begun, the development process must feed the change back from the design phase to the product definition phase and repeat the product definition (i.e., make the change), thereby increasing total project duration and cost. More sophisticated models based on the CPM/PERT paradigm address some of these limitations (e.g., Golenko-Ginzburg and Gonik 1996; Christensen *et al.* 1996), but cannot fully model development processes. Rodrigues and Bowers (1996) and Lyneis (1996) review and evaluate the CPM/PERT approach to project modeling. To be complete, a project model must model the network of project phases. The phased network structure of projects has been incorporated into several system dynamics project models (e.g., Williams *et al.* 1995; Ford, Hou and Seville 1993; Cooper 1980; Roberts 1974), as well into the model we describe in this paper. However, although modeling a project's phase network is necessary it is not sufficient for capturing development processes.

Other research approaches identify some dynamic consequences of different project structures on project performance. For example, the dynamic consequences of iteration among project phases on cycle time have been addressed

directly with the Design Structure Matrix (Smith and Eppinger 1997; Eppinger *et al.* 1994; Steward 1981). The Design Structure Matrix has been used to map and predict information flows among development phases (Morelli, Eppinger and Gulati 1995), study time/quality tradeoff decisions (Chao 1993) and variability in cycle times (Osborne 1993). Design Structure Matrix research demonstrates the results of iteration between phases but does not represent the underlying processes that drive cycle time. A description of the process structure in the form of the causal relationships that generate project behavior is needed to investigate how project processes drive performance. Other model structures based on project characteristics have been suggested (Rodrigues and Williams 1996) and described conceptually (Pugh-Roberts undated; Cooper 1980). However, the specific features and characteristics that distinguish different development processes have not been described at a formal model level of detail.[1]

A complete causal dynamic project model must also explicitly model and integrate the influences of processes, resources, scope and targets on performance. Three of these four features have been modeled extensively by system dynamics researchers. Several system dynamics models have been used to model project resources and investigate the effects of resource management on project progress (e.g., Richardson and Pugh 1981; Cooper 1980; Roberts 1974; Homer *et al.* 1993). For example, Abdel-Hamid (1984) built a system dynamics model of software development in which progress is driven by the daily work force of software developers (resource quantity) and software developer productivity (resource effectiveness). Considerable attention has also been given to the secondary effects of changes in project scope. Cooper (1980) and Reichelt (1990) related customer-initiated design changes to total scope of work and litigation costs, and Pugh-Roberts Associates (Cooper 1980) among others (Williams *et al.* 1995) have used such models extensively and successfully in support of litigation over cost overruns in aerospace, defense, shipping and construction. The influences of schedule targets on performance have been modeled by Abdel-Hamid (1984), Richardson and Pugh (1981) and Roberts (1974) and the evolution of one form of project targets (customer demands) in response to performance has been studied by Fiddaman, Oliva and Aranda (1993). However, little investigation of how the demands and constraints of different development processes influence project performance has been made.

Excluding phase-specific development process structures from project models implicitly assumes that those development processes have no impact on project performance. Yet the availability of work as described by the precedence relationships within and between phases is an important constraint on project performance (Rosenau and Moran 1993; Clark and Fujimoto

1991; Wheelwright and Clark 1992; Moder *et al.* 1983). This simplifying assumption can also lead to grossly unrealistic performance predictions under extreme conditions. For example, a project model driven solely by resources could allow schedule and quality performance to improve as resources grow until, at the limit of infinite labor, the project is predicted to be completed in an infinitesimally small time.

A more suitable description of development dynamics must include iterative flows of work, distinct development activities and available work constraints both within and among development phases. The existing system dynamics models of projects which include process structures have focused on the roles of two development activities. Cooper (1980; 1993; 1994) first and several researchers subsequently (e.g., Kim 1988; Abdel-Hamid 1984; Richardson and Pugh 1981) modeled two development activities by distinguishing between initial completion and rework. Ford *et al.* (1993) expanded this approach to model three development activities (initial completion, required rework, and optimal rework to improve quality). However, these models do not explicitly include other important development activities identified in the product development literature including quality assurance (Rosenau and Moran 1993; Rosenthal 1992; Wheelwright and Clark 1992) and coordination (Adler 1995, Cooper and Kleinschmidt 1994; Clark and Fujimoto 1991). Abdel-Hamid's model includes quality assurance and models the impact of coordination on productivity at the aggregate whole-project level, but does not consider the interactions caused by the need for different phases to coordinate. System dynamics models of iterative flows have evolved from single flows of accurate work slowed by implicit iteration through separate streams of correct and flawed tasks (Abdel-Hamid 1984; Richardson and Pugh 1981) to more realistic closed-loop flows (Ford 1995; Ford *et al.* 1993; Kim 1988). Homer *et al.* (1993) first described development processes with available work constraints imposed by upstream development phases or by the sequence of tasks within a single phase. The dynamic concurrence constraints in the model described in this paper have their conceptual foundation in the Homer *et al.* model. In this work distinct development activities, iteration and dynamic concurrence are integrated in a single model to formally describe development processes in a generic and flexible form. Our formal structure descriptions allow model evaluation, comparison with other model structures and model improvement which have been lacking.

This article describes a product development project model that explicitly models all four performance drivers—process, structure, resources, targets, and scope—in a project network. We calibrated and tested the model for the case of a medium-scale semiconductor product development project. We focus

on the modeling and role of dynamic concurrence on project performance. The importance of integrating process structure with resources, scope, and targets in dynamic models of projects and future research is discussed.

The product development project model

Our model simulates the performance of a multiple-phase development project. Each phase is represented by a generic structure, which is customized to reflect a specific stage of product development such as preparing construction drawings for building an office complex, writing software code, or testing computer chip prototypes. The generic phase structure has four subsystems which interact to affect project performance. The four subsystems are development processes, resources, scope, and targets. We use the three traditional measures of project performance (time, quality, and cost). They are reflected in the model with cycle time, changes, and cost. Primary phase subsystem interactions are shown in Figure 1, including floating goal structures, resource constraints and the generation of a demand for resources by development processes. The development processes subsystem is the focus of this work. Ford (1995) describes the other subsystems in detail.

When the dependencies within individual phases and between the phases in the example network are described with a Design Structure Matrix (Smith and Eppinger 1997) over 80% (13 of 16) of the cells are occupied. This indicates a highly interdependent process in which iteration is particularly important. The

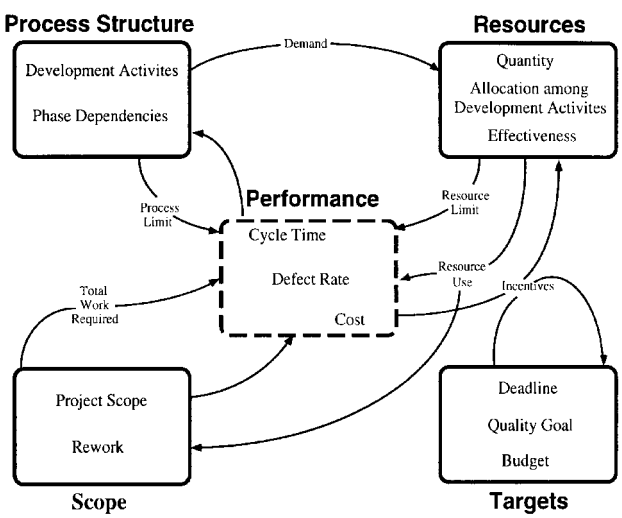

Fig. 1. Phase subsystems

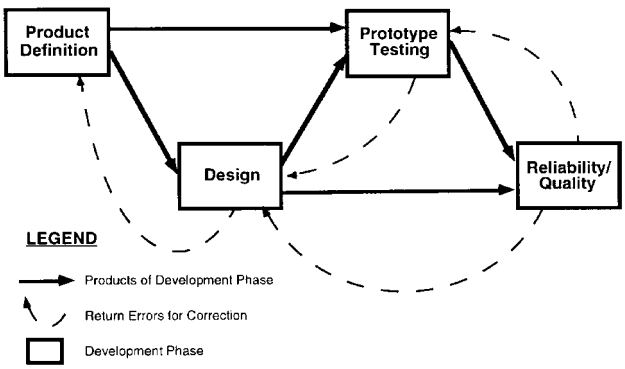

Fig. 2. A project network

links shown in Figure 2 represent several forms of inter-phase interaction, including:

- Work progress in upstream phases constrains progress in their dependent downstream phases. The locations of these constraints are shown by the solid arrows in the project network.
- Changes inherited by downstream phases from upstream phases corrupt downstream work which must then be corrected. When inherited changes are discovered by a downstream phase they are returned for correction or improvement to the phase where the change was generated. These flows are shown by the dashed arrows in the project network.
- The correction of changes requires coordination between the phase that discovered the change and the phase that generated the change.
- Schedule, quality, and cost performance in individual phases influence the conformance of the entire project to the project targets. The responses of project managers to performance affects the targets set for each project phase.

The model allows us to represent projects with any number of phases, which can be linked in an arbitrarily complex network of concurrence relations, including sequential, parallel, partially concurrent, and other relationships. The fundamental units that flow through a project are ''development tasks''. Conceptually a development task is an atomic unit of development work. Examples of development tasks include the selection of a pump, writing a line of computer code and installing a steel beam. The unit of work used to describe a development task may differ among project phases. For example, a product definition phase might use product specifications as the basis for tasks, whereas the design phase of the same project might use lines of computer code. We assume that tasks within a phase are uniform in size and are fungible. This

assumption becomes more accurate as task size decreases. Therefore, relatively small pieces of development work are selected as tasks. Fungible tasks are characteristic of some development project phases (e.g., the delivery and placement of soil for a roadbed). Many other development phases have interdependent but fungible tasks (e.g., software code as in Abdel-Hamid, 1984). Tasks are also assumed to be small enough to either require a change or be correct but not require a partial change. This assumption also becomes more accurate as task size becomes smaller. These assumptions help identify divisions among development phases and development tasks.

We describe development within a project with four activities: completion, quality assurance, iteration, and coordination. Completion is finishing a development task the first time. Quality assurance is the inspection of tasks for changes. Work on tasks to make changes subsequent to their initial completion is referred to as iteration or change. Changes can arise to correct defects, improve quality, respond to changes in upstream activities such as changes in customer requirements, or other environmental disruptions such as changes in suppliers. Coordination is the integration of the product development project among phases. An example of coordination is when designers work with marketers to refine product specifications. Project processes, resources, scope, and targets influence all four development activities. By customizing the characterizations of these features the model can be used to represent many different types of development projects. Process structures simulate the constraints imposed on progress by different development activities, their interactions, and the availability of work within and among phases. Resource structures simulate the effects of work force sizes, the allocation of labor among the four development activities, productivity, experience, and fatigue. Scope structures model the original scope of work for each phase and scope changes in response to schedule, cost, and quality pressures. Target structures simulate the specification and modification of overall project and phase-level targets for cycle time, quality, and cost relative as well as the pressures on developers due to poor performance.

Describing development processes

Development processes in a single project phase

Individual development phases include processes that can constrain project progress. Consider as an example the erection of the structural steel skeleton for a ten-story building. The steel erection requires that the structural members (the columns, beams, and bracing) be installed, inspected for proper installation

and corrected if the installation is found to be defective. For any given technology an average amount of time is required for each of these three development activities to be performed for each structural member, regardless of the quantity or effectiveness of the resources (e.g., laborers and cranes) applied. These average process durations are unique characteristics of the structural steel installation, inspection, and correction activities and descriptive of the development process used in the steel erection phase of the project. An additional constraint imposed by the structure of these three development activities is that they cannot be performed simultaneously on any single structural member since inspection requires that installation be complete and correction of defective work requires that any defects have been discovered through inspection. Additionally, work on some steel members such as beams must wait for the completion of other work such as the installation of the columns that support those beams. Therefore, not all the structural members can be worked on simultaneously. This feature of the steel erection process limits the availability of work based on the amount of work which has been completed and therefore limits the degree of concurrent development possible within a single development phase. Finally, after installation, inspection, and any required corrections, the structural members that have been approved may not be released to subsequent development phases until a sufficient number of members have been approved. For example, approved steel work may not be released for the installation of utilities until all the steel on an entire floor of the building is approved. This can also delay the final completion of the development project. How can these constraints be modeled?

Our model uses three features to describe the development process in a single phase: circular iteration, multiple development activities, and dynamic concurrence. Circular iteration is described with the stock and flow structure (Figure 3). In our model development tasks flow among five states: Tasks not Completed (TnC), Tasks Completed by not Checked (TCnotCk), Tasks to be Changed (TtCh), Tasks Approved (TAppr) and Tasks Released (TRel). Tasks initially reside in the Tasks not Completed stock. The first development activity is Complete Tasks (CT). Completed tasks accumulate in the Completed not Checked stock. If no tasks require changes or those changes are not discovered during quality assurance, the tasks leave the Completed not Checked stock and pass through the Approve Tasks (ApprT) flow into the stock of Tasks Approved (TAppr). Approved tasks are subsequently released through the Release Tasks (RelT) flow to the stock of Tasks Released (TRel). This represents delivering tasks to the managers of downstream phases or to customers. Tasks needing changes are discovered through the Quality Assurance (QA) activity. These tasks move through the Discover Changes (DiscCh) flow from the

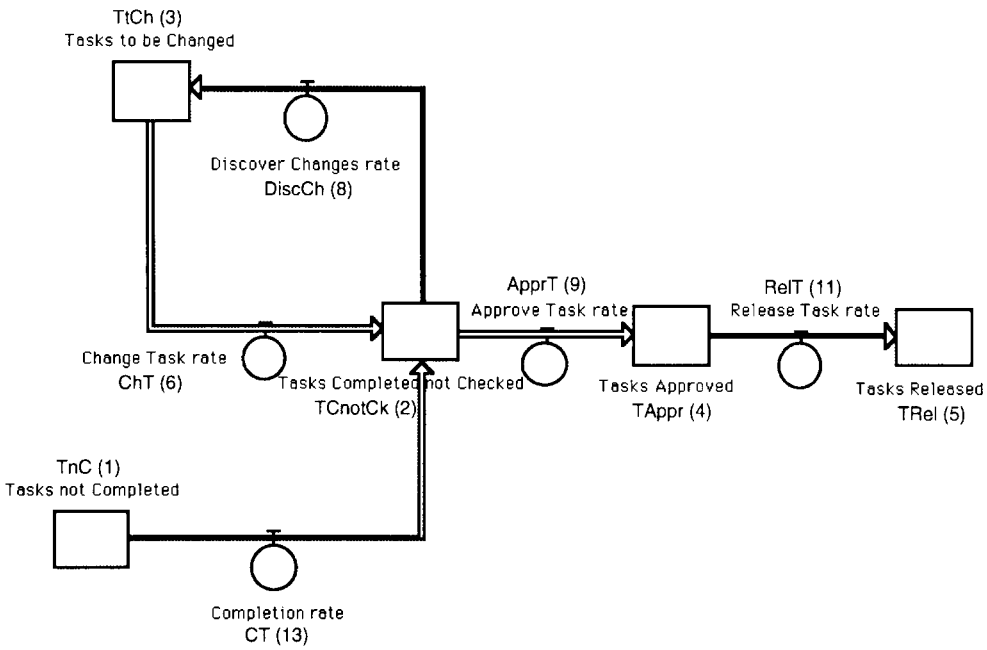

Fig. 3. Development process model stocks and flows of a single phase

Completed not Checked stock to a stock of Tasks to be Changed. These tasks are corrected or improved through the Change Tasks (ChT) activity and returned to the Completed not Checked stock. Changes can be generated during both completion and correcting or improving tasks. The structure of these stocks and flows is shown in Figure 3 with the variable name abbreviations and their equation numbers.

An analogy of paper passing from box to box on the project manager's desk can clarify the movement of tasks through a development phase. Each task is written on a sheet of paper. The scope of the phase is the number of sheets of paper. Before development begins all the sheets of paper are in the project manager's "In" box (Tasks not Completed). As the tasks are completed the pieces of paper move from the "In" box to the project manager's "Work to be Checked" box, which contains the contents of the Completed not Checked stock. Quality assurance moves sheets of paper from the "Work to be Checked" box into the project manager's "Corrections Required" box (Tasks to be Changed) or to the "Okay" box (Tasks Approved). Correcting errors moves paper from the "Corrections Required" box back to the "Work to be Checked" box. When the project manager considers the number of sheets of paper in the "Okay" box to be sufficient, he shifts them into the "Done" file (Tasks Released), analogously releasing the approved work to downstream phases or customers.

Five differential equations describe the flows of development tasks through any single phase j in a project of n phases where $j \in \{1, 2 \ldots, n\}$. We omit the subscript j for clarity.

$$(d/dt)(\mathrm{TnC}) = -\mathrm{CT} \tag{1}$$

$$(d/dt)(\mathrm{TCnotCk}) = \mathrm{CT} + \mathrm{ChT} - \mathrm{DiscCh} - \mathrm{ApprT} \tag{2}$$

$$(d/dt)(\mathrm{TtCh}) = \mathrm{DiscCh} - \mathrm{ChT} \tag{3}$$

$$(d/dt)(\mathrm{TAppr}) = \mathrm{ApprT} - \mathrm{RelT} \tag{4}$$

$$(d/dt)(\mathrm{TRel}) = \mathrm{RelT} \tag{5}$$

The primary feedback loops in the model's process structure for a single phase are shown in Figure 4. The negative loop depicts the reduction in the number of tasks available for completion as work is completed. The completion rate is based on the Tasks Available for Completion, the Average Complete Task Duration and the Resource Constraint. An increase in the completion rate increases the number of Tasks Completed, which decreases the number of Tasks Available for Completion. If it is assumed that an average time is taken for the completion of each task and that all available tasks can be worked on simultaneously, a decrease in the number of Tasks Available for Completion reduces the completion rate. This loop introduces the feature we use to describe each of the development activities in a specific development phase, the Average Process Activity Duration. The Average Process Activity Duration is the average time required to complete a development activity on a task if all required information, materials and resources are available and no changes are

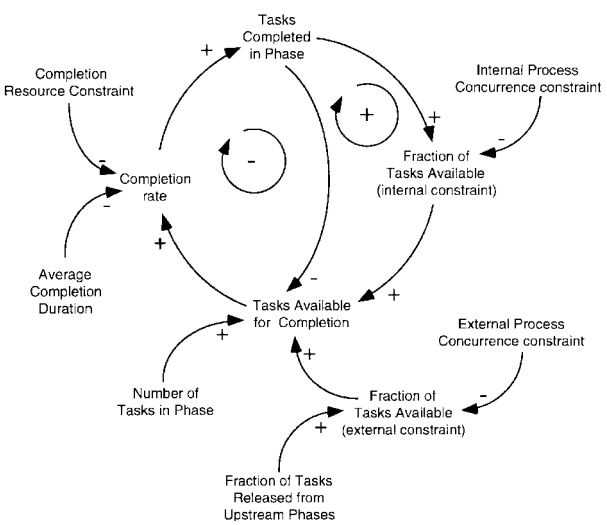

Fig. 4. Feedback loops in a single development phase for the complete task development activity

generated. It describes the purest time constraint the process imposes on progress by answering the question "How fast (on average) would a development activity on a task be completed if everything needed was available?" For example, "How fast would a structural steel member be installed if all the best equipment and installers were available and the installers knew how to install the member?" In Figure 4 the Average Process Activity Duration is applied to completion. The model uses structures similar to the negative loop shown in Figure 4 to describe the role of the Average Quality Assurance Duration in discovering changes, the Average Change Duration to correct or improve tasks, and the Average Coordination Duration to resolve inter-phase concerns.

We formally model the process structure for a single phase with Equations (1) through (5) (Figure 3) and the following 13 equations (Figure 5).

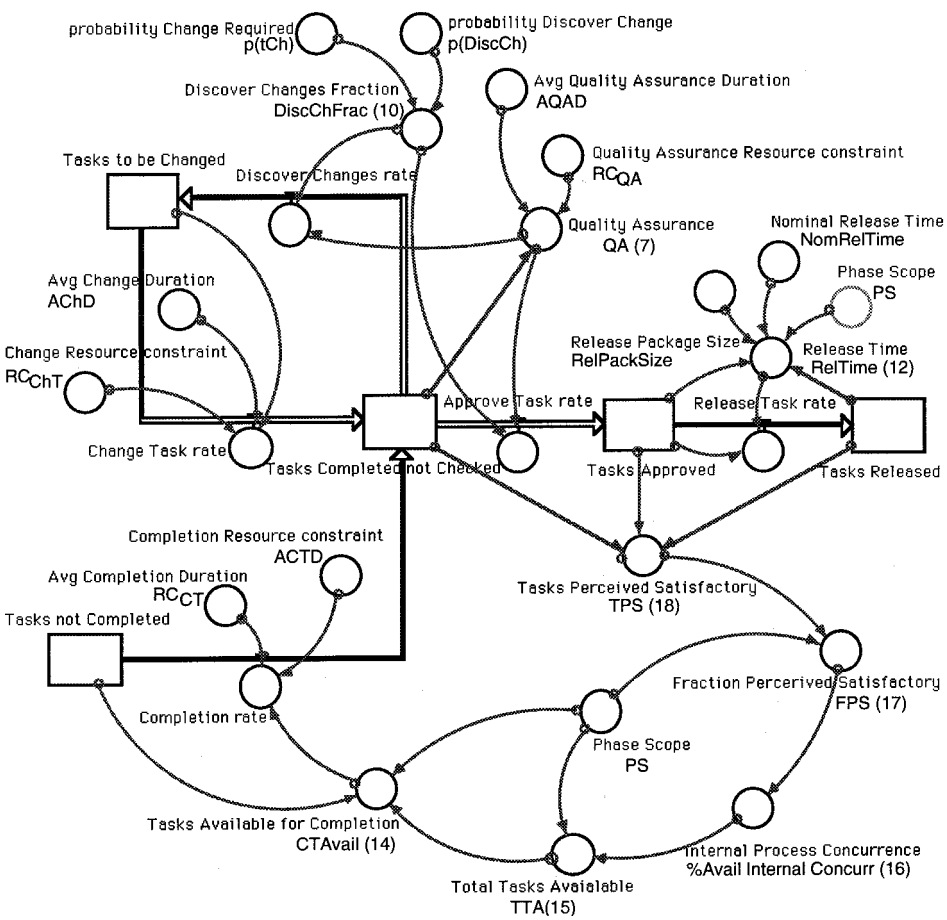

Fig. 5. The development process structure for a single project phase

Three development activities drive the flows of tasks in each project phase: completion, quality assurance, and change. Each activity requires both that sufficient resources are available to undertake the work (that is, enough skilled workers) and that enough of the information and material needed to complete the activity is available (that is, that there is a stock of tasks which can be addressed as a result of completion of prior activities). Therefore, the progress rate for each of the three development activities is the minimum of the rate allowed by resources and the rate allowed by the process of performing the specific development activity. The equations for the four flows in a single phase that depend directly on development activities are based on the Resource Constraint, the Average Process Activity Durations of the three development activities and the work available for each activity. The work available for the change activity is the number of tasks known to require changes. Therefore the average change rate is the lesser of the Change Resource Constraint (RC_{ChT}) and the number of tasks waiting to be changed divided by the average time that the process requires to change a task, the Average Change Duration (AChD):

$$ChT = Min(RC_{ChT}, TtCh/AChD) \quad (6)$$

In a similar manner the work available for the quality assurance activity is the number of tasks Completed but not Checked. Therefore the average quality assurance rate (QA) is the lesser of the Quality Assurance Resource Constraint (RC_{QA}) and the number of tasks waiting for quality assurance (TCnotCk) divided by the average time that the process requires to perform quality assurance on a task (AQAD):

$$QA = Min(RC_{QA}, TCnotCk/AQAD) \quad (7)$$

The two flows based on the quality assurance activity include explicit modeling of the generation and discovery of changes. We assume that quality assurance efforts are not perfect and some changes are missed. Therefore, some tasks that must be changed are mistakenly considered to be finished and therefore are approved and subsequently released to downstream phases with the tasks that do not require changes. We assume that no tasks that do not require changes are mistakenly believed to need correction or improvement. We model change rates with the fraction of tasks that are discovered to require changes. (DiscChFrac). This fraction (measured in %) is the product of the probability a given task requires a change (p(TCh)) and the accuracy of quality assurance, which we measure with the probability of discovering the need for a change if it exists (p(DiscCh)). Changes are modeled with a parallel co-flow structure

similar to the development task structure. The equations for the change co-flow structure are shown in the Appendix. The number of tasks found to need changes is the product of the quality assurance rate and the Discover Change Fraction. The Approve Tasks rate (ApprT) is the number of tasks checked by quality assurance but not found to need changes. Therefore, the formulations for the Discover Changes rate (DiscCh), Approve Tasks rate (ApprT) and Discover Change Fraction (DischChFrac) are:

$$\text{DiscCh} = \text{QA} * \text{DiscChFrac} \tag{8}$$

$$\text{ApprT} = \text{QA} * (1 - \text{DiscChFrac}) \tag{9}$$

$$\text{DiscChFrac} = p(\text{DiscCh}) * p(\text{TCh}) \tag{10}$$

As shown on the right side of Figure 5, approved tasks are released based on a Release Package Size (RelPackSize), which describes the minimum fraction of the unreleased tasks (the Project Scope less the Tasks Released) required in the Tasks Approved stock to begin releasing work. The tasks that have been approved are released when the number of Tasks Approved equals or exceeds this threshold. Tasks are released over a Release Time (RelTime), which approaches infinity when the Release Package Size criteria has not been met and is the Nominal Release Time (NomRelTime) period otherwise. The Nominal Release Time is an average time required to release approved tasks and can vary with the task processes and policies of specific projects or phases. We model the instantaneous release of tasks by setting the Nominal Release Time to the simulation time step. Therefore the equations for the release of work are:

$$\text{RelT} = \text{TAppr}/\text{RelTime} \tag{11}$$

$$\text{RelTime} = \text{IF } (\text{TAppr} \geqslant \text{RelPackSize} * (\text{PS} - \text{TRel})) \text{ THEN (NomRelTime) ELSE } (\infty) \tag{12}$$

The Completion rate (CT) is formulated similarly to the change and quality assurance rates. The completion rate is based on the Resource Constraint (RC_{CT}), the Average Complete Task Duration (ACTD) and the work available for Completion (CTAvail):

$$\text{CT} = \text{Min}(\text{RC}_{\text{CT}}, \text{CTAvail}/\text{ACTD}) \tag{13}$$

The number of tasks available for completion depends on the Internal Process Concurrence relationship, another characteristic feature of the development process. A phase can represent multiple activities, not all of which can necessarily be performed independently. The positive loop shown in Figure 4 models the increase in the number of tasks that will become available for

completion as work within the phase is completed. For example, in the construction of a ten-story building the structural work on the upper floors is not available until the lower floors that support them are completed. The Internal Process Concurrence relationship answers the question ''How much work can now be completed based upon how the work has progressed thus far?'' We describe the specification and estimation of Internal Process Concurrence constraints below. The number of tasks available for initial completion can also be constrained by upstream phases. Testing, for example, cannot commence until a prototype is completed. We model these inter-phase constraints with External Process Concurrence relationship (described later).

If we move backwards through the causal path in the lower portion of Figure 5 (counterclockwise), the tasks available for completion (CTAvail) are those that can be completed less those that have already been completed. Therefore the number of tasks available for completion is the difference between the Total Tasks Available (TTA) and those tasks that have been completed (the Phase Scope less the Tasks not Completed). Finding a large number of changes while the completion rate is low can reduce the fraction of tasks perceived to be satisfactory. This can in turn reduce the total tasks available to a level below the number of tasks completed, causing an infeasible negative number of tasks available for initial completion. In actual projects this reflects a condition in which tasks requiring changes must be worked on before more tasks can be completed. The maximum function incorporates this constraint into the equation for the tasks available for completion:

$$\text{CTAvail} = \text{Max}(0,\ \text{TTA} - (\text{PS} - \text{TnC})) \tag{14}$$

The Total Tasks Available is the product of the Fraction of Tasks Available, as defined by the constraint described by the Internal Process Concurrence relationship (%Avail Internal Concurr) and the number of tasks in the phase (the Phase Scope, PS). The Fraction of Tasks Available due to the Internal Process Concurrence constraint (%Avail Internal Concurr) is a function of the fraction of the phase scope that is perceived to have been completed satisfactorily, Fraction Perceived Satisfactory (FPS). The Fraction of tasks Perceived Satisfactory is the ratio of the Tasks Perceived Satisfactory thus far (TPS) and the Phase Scope (PS). The number of Tasks Perceived to be completed Satisfactorily is either the sum of the Tasks Completed but not Checked (TCnotCk), the Tasks Approved (TAppr) and the Tasks Released (TRel) or the number of approved or released tasks, depending on whether developers use unchecked work as the basis for additional development. The tasks to be changed are not included because tasks known to require changes do not make

additional work available. Whether to consider the tasks Completed but not Checked satisfactory and therefore include them in the Tasks Perceived Satisfactory can be viewed as a project management policy which can be investigated with our model. We include unchecked tasks based on the results of our field studies. Therefore, the final equations of our single-phase process model are:

$$\text{TTA} = \text{PS} * \%\text{Avail Internal Concurr} \tag{15}$$

$$\%\text{Avail Internal Concurr} = f_{\text{IPC}}(\text{FPS}) \tag{16}$$

$$\text{FPS} = \text{TPS}/\text{PS} \tag{17}$$

$$\text{TPS} = \text{TCnotCk} + \text{TAppr} + \text{TRel} \tag{18}$$

Describing internal process concurrence

A project phase's Internal Process Concurrence relationship describes the interdependency of the tasks within the phase. In describing the construction of the structure for an office building, the Internal Process Concurrence relationship could capture the physical constraint that the floors must be completed sequentially one at a time from the ground up because lower floors support those above. These constraints can act as a bottleneck in the availability of work. Most previously published system dynamics models of projects have assumed that all uncompleted tasks are available for completion or have incorporated these constraints into other project features (e.g., Abdel-Hamid 1984; Richardson and Pugh 1981; Roberts 1974). An assumption that all incomplete tasks are available implies that all tasks are independent and could be performed in parallel. Product development research (e.g., Rosenthal 1992; Wheelwright and Clark 1992; Clark and Fujimoto 1991) and the example above show that processes can and frequently do constrain the availability of work.

Internal Process Concurrence relationships capture the degree of sequentiality or concurrence of the tasks aggregated together within a phase, including possible changes in the degree of concurrence as the work progresses. As shown in Figure 6, all Internal Process Concurrence relationships for feasible projects must lie strictly above the 45° line, otherwise work could not be completed until it was already completed. Within the feasible region a variety of functional forms are possible including non-linear concurrence relationships. In general more concurrent processes are described by curves near the upper and left axes of the Internal Process Concurrence graph and less concurrent processes are described by curves near the 45° line. Figure 6 shows the Internal

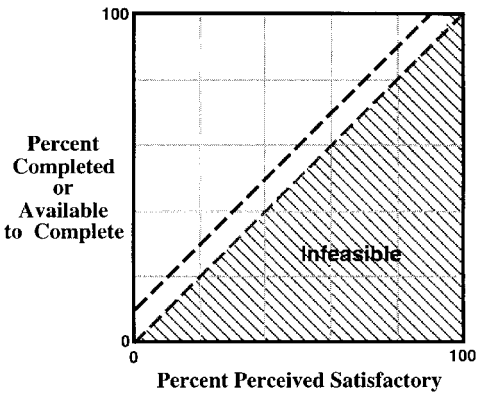

Fig. 6. A linear internal process concurrence relationship

Process Concurrence relationship for the construction of the structure for a ten-story office building in which the floors are erected sequentially from the ground up. At the beginning of the project only the first floor (10%) is available to be completed. The completion of portions of the first floor make available portions of the second floor. When the first floor is finished 10% is completed and the entire second floor is available, making 20% completed or available to be completed. This linear progression continues until the completion of the ninth floor releases the final floor for completion (100% completed or available to be completed when the phase is 90% complete).

In contrast to the office building example, many such relationships are not linear. The Internal Process Concurrence relationship shown in Figure 7 is based upon the design of software code used to create the layout of a new computer chip and was derived from fieldwork with the design team (Ford 1995). The code to be produced is aggregated into several blocks. A few of these blocks of code must be designed and written before other blocks can be

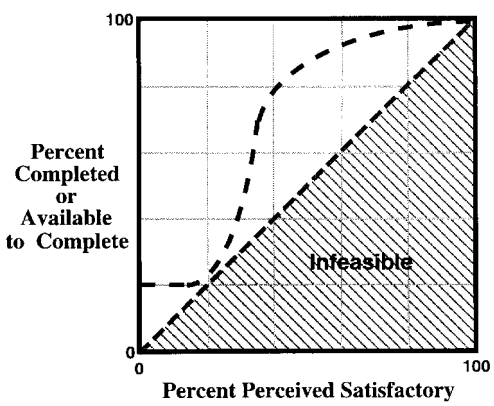

Fig. 7. A non-linear internal process concurrence relationship

started. Therefore, only these important blocks (estimated to be 20% of the code) can be produced at the beginning of the design phase. More code does not become available until these blocks approach completion. The increased potential concurrence of subsequent code is reflected in the steep rise of the function between 20 and 40% Perceived Satisfactory. When most of the blocks of code have been written, the work of integrating the blocks into a single operational program begins. Integration is fundamentally less parallel, producing the flat tail on the right side of the graphical function.

Development processes among multiple project phases

We capture the links among project phases in two ways: External Process Concurrence relationships and coordination. External Process Concurrence relationships are used to describe available work constraints between development phases in a manner analogous to the internal available work constraints described by Internal Process Concurrence. An External Process Concurrence relationship describes the amount of work that can be done in a downstream phase based on the percentage of work released by an upstream phase. For example, the testing of a prototype cannot begin until the prototype is built and the amount of detailed design work that can be completed is constrained by the amount of product definition work that has been released to the detailed design phase.

The purpose of External Process Concurrence relationships is similar to the purpose of the precedence relationships used in the Critical Path and PERT methods: to describe the dependencies of development phases on each other. However External Process Concurrence relationships as used here can describe these relationships in greater detail and richness than the precedence relations used in many Critical Path and PERT methods for several reasons:

- External Process Concurrence relationships describe the dependency between two phases along the entire duration of the phases instead of only at the start and finish of the phases as in the Critical Path and PERT methods.
- External Process Concurrence relationships can be non-linear, whereas precedence relationships used in the Critical Path and PERT methods cannot. That is, the External Process Concurrence relationship can represent changes in the degree of possible concurrence between two phases as work on the upstream phase progresses.
- External Process Concurrence relationships describe a dynamic relationship between development phases by allowing the output (Percent Tasks

Available for Completion) to vary over the life of the project depending on the current conditions of the project. For example, if design drawings are returned from construction (the downstream phase) to design (the upstream phase) for iteration, then the work available to construction is reduced, possibly requiring construction to cease until the drawings are changed and re-released. In contrast the precedence relationships used in many Critical Path and PERT methods are static.

External Process Concurrence relationships can be applied from any upstream development phase on which a phase depends. In our multiple-phase model the Total Tasks Available for Completion (TTA) is based on the minimum (i.e., tightest) of the constraints set by the phase's Internal Process Concurrence relationship (%Avail Internal Concurr) and the External Process Concurrence relationships (%Avail External Concurr) which link the phase to upstream development phases. The revised equation for the total tasks available is:

$$\text{TTA} = \text{PS} * \text{Min}(\%\text{Avail Internal Concurr}, \%\text{Avail External Concurr}) \quad (15a)$$

The Fraction of Tasks Available due to the External Process Concurrence relationship (%Avail External Concurr) is the minimum of the constraints between the phase and each upstream phase in the project network (%Avail External Concurr_j). The fraction available due to each constraint is a function of the Fraction of tasks Released from the Upstream phase in the project network (FRelUp_i). The function is defined by the External Process Concurrence relationship ($\text{EPC}_{i,j}$) between the focal phase (denoted with subscript j where $j \in \{1, 2 \ldots, n\}$) and each upstream phase in the activity network (denoted with subscript i). The Fraction of tasks Released from the Upstream phase is the ratio of the number of tasks released by the upstream phase (TRel_i) to the scope of the upstream phase (PS_i):

$$\%\text{Avail External Concurr}_j = \text{Min}(f_{\text{EPC}_{i,j}}(\text{FRelUp}_i)) \text{ for } i \neq j \quad (19)$$

$$\text{FRelUp}_i = \text{TRel}_i / \text{PS}_i \quad (20)$$

Note that the External Process Concurrence relationship $f_{\text{EPC}_{i,j}} = 1$ for upstream phases i that do not constrain downstream focal phases j as shown in Figure 8(a). In general, External Process Concurrence relationships describe more concurrent processes with curves near the upper and left axes of the External Process Concurrence graph and less concurrent processes with curves near the lower and right axes. Figure 8 illustrates several possible External Process Concurrence relationships. Four of the examples (8(a), 8(b), 8(c) and 8(d))

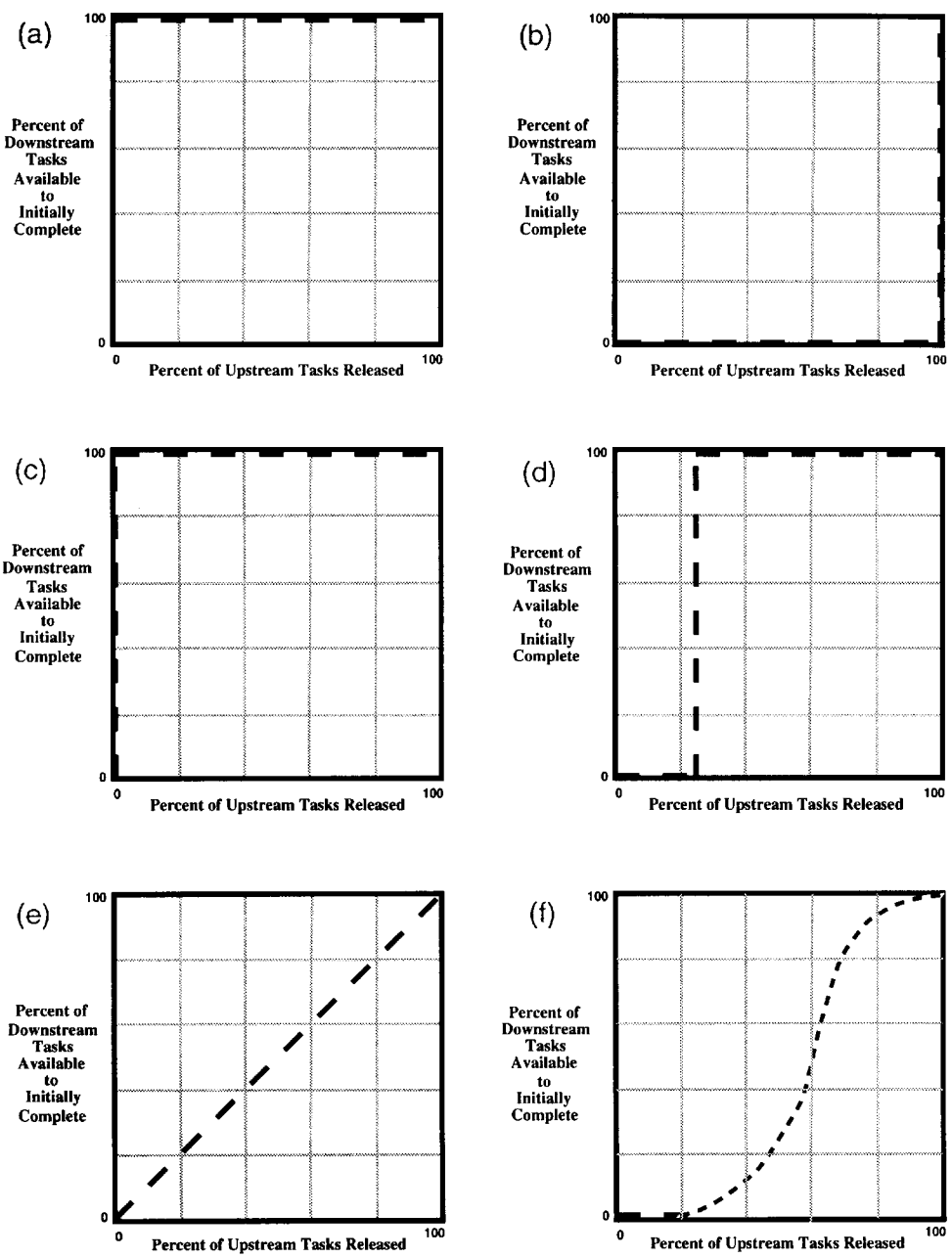

Fig. 8. Examples of External Process Concurrence relationships. (a) No inter-phase relationship; (b) sequential inter-phase relationship; (c) parallel inter-phase relationship; (d) delayed-start inter-phase relationship; (e) "lockstep" inter-phase relationship; (f) delay with partially concurrent inter-phase relationship

demonstrate how External Process Concurrence relationships can be used to describe inter-phase relationships commonly used by the Critical Path method and PERT. For example, infinite-order delays between phases based on development progress can be described by shifting the point along the abscissa at which the first downstream tasks become available (Figure 8(d)). Two of the examples (Figures 8(e) and 8(f)) show inter-phase relationships which can be described with External Process Concurrence but cannot be described with the Critical Path method or PERT without modeling project activities at an unrealistically high level of detail. Different levels of concurrence among phases can be described with External Process Concurrence relationships by altering the shape of the curve in the graphical function. For example, Figure 8(e) describes an inter-phase relationship in which the downstream phase can proceed at the same rate as the upstream phase but not faster. The development process allows them to proceed in ''lockstep''. Figure 8(f) describes an inter-phase relationship in which the downstream phase must wait until the upstream phase has released 20% of its tasks and then can proceed at varying degrees of potential concurrence.

Coordination

The second development process that directly links project phases is coordination. Coordination describes the inter-phase effects of releasing and inheriting changes. For example, engineers may need to meet with product architects to explain why certain product specifications cannot be met within the time and budget available, and then revise the specifications. Our explicit modeling of coordination allows us to distinguish between two important types of changes encountered in product development, changes caused by factors internal to a development phase and changes caused by using work from upstream phases which requires changes itself as the basis for development work.

Tasks requiring inter-phase coordination, such as meeting to explain or resolve problems, accumulate in a stock of Tasks to be Coordinated before they are coordinated and moved to the Tasks to be Changed stock (Figure 9). Two flows fill the Tasks to be Coordinated stock. First, tasks that are discovered to need changes due to the inheritance of changes from upstream phases flow from the Completed not Checked stock into the stock of Tasks to be Coordinated. We call this flow Discover Inter-Phase Changes (DiscInterPhaseCh) to distinguish it from the flow of internally generated changes (equation 8), which we rename Discover Intra-Phase Changes (DiscIntraPhaseCh) for clarity. Second, the tasks requiring changes that have been released, discovered by a downstream phase and returned for coordination and correction or

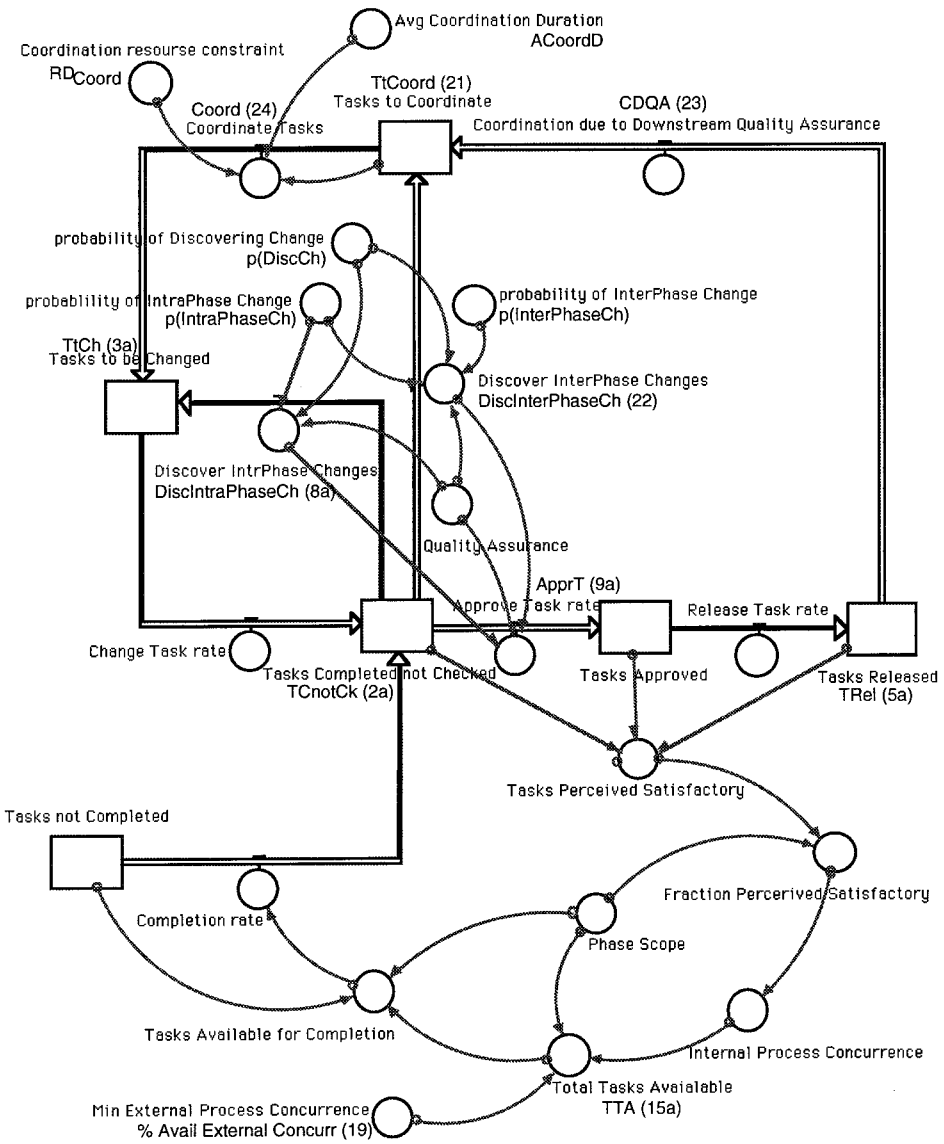

Fig. 9. Multiple-phase process concurrence and coordination structure

improvement are removed from the upstream phase's stock of Tasks Released and enter the stock of Tasks to be Coordinated (TtCoord) through the flow of Coordination due to Downstream Quality Assurance (CDQA). Tasks leave the Tasks to be Coordinated stock and enter the Tasks to be Changed stock through the coordination activity (Coord). Therefore the equation for the stock of Tasks to be Coordinated (TtCoord) and revised equations for the Tasks

Completed but not Checked, Tasks to be Changed, Tasks Released stocks and the Approve Tasks that reflect coordination are:

$$(d/dt)(\text{TtCoord}) = \text{DiscInterPhaseCh} + \text{CDQA} - \text{Coord} \quad (21)$$

$$(d/dt)(\text{TCnotCk}) = \text{CT} + \text{ChT} - \text{DiscIntraPhaseCh} - \text{DiscInterPhaseCh} - \text{ApprT} \quad (2a)$$

$$(d/dt)(\text{TtCh}) = \text{DiscIntraPhaseCh} + \text{Coord} - \text{ChT} \quad (3a)$$

$$(d/dt)(\text{TRel}) = \text{RelT} - \text{CDQA} \quad (5a)$$

$$\text{ApprT} = \text{QA} - \text{DiscIntraPhaseCh} - \text{DiscInterPhaseCh} \quad (9a)$$

To model the discovery of changes caused by inheriting tasks that require changes from upstream phases (DiscInterPhaseCh), we distinguish between flows of tasks with internally and externally (upstream) generated changes. This is because the flow of tasks with internal changes, which we call Discover IntraPhase Changes (DiscIntraPhaseCh), goes directly from the Completed not Checked stock to the Tasks to be Changed stock, but the flow of tasks with changes due to inherited changes (DiscInterPhaseCh) flows from the Completed not Checked stock to the Tasks to be Coordinated stock before going to the Tasks to be Changed stock. We model internal changes with a co-flow structure similar to the development tasks structure described above. We model changes due to inherited upstream changes with a second co-flow structure, which is also based on the development tasks structure. The equations for both co-flow structures and their basis in the development tasks structure are shown in the Appendix. See Ford (1995) for model structures of project features that affect change generation rates.

An assumption is required to allocate the tasks requiring changes due to both internal and external changes to the flow of Discover IntraPhase Changes or Discover InterPhase Changes. We assume that developers will correct changes generated within their own phase before bringing those tasks to coordination so as to avoid revealing to colleagues that they have made errors (a behavior observed in our field studies). Therefore, the overlapping discoveries can reasonably be assumed to go directly to the Tasks to be Changed stock. Other assumptions are easily accommodated. Based on this reasoning and the densities of the two kinds of changes in the stock of tasks Completed but not Checked, we redefine the flows of tasks requiring changes from the Completed not Checked stock as:

$$\text{DiscIntraPhaseCh} = \text{QA} * p(\text{DiscCh}) * p(\text{IntraPhaseCh}) \quad (8a)$$

$$\text{DiscInterPhaseCh} = \text{QA} * p(\text{DiscCh}) * (p(\text{InterPhaseCh}) - (p(\text{IntraPhaseCh}) * p(\text{InterPhaseCh}))) \quad (22)$$

The flow of Coordination due to Downstream Quality Assurance (CDQA) arises because not all needed changes within a phase are caught by quality assurance. Often tasks requiring changes are released and subsequently discovered by downstream phases. This flow is the sum of the IntraPhase Changes and InterPhase Changes released by the phase and discovered by all its downstream phases. These flows are modeled in the change co-flow structures based on the probability of the focal phase (denoted j) releasing a change to a downstream phase (denoted $i \in \{1, 2 \ldots, n\}$), the probability of the receiving phase discovering the inherited change ($p(\text{DischCh}_i)$) and the rate at which the receiving phase is checking tasks for changes through quality assurance (QA_i). The probability of the focal phase releasing a task requiring change is the ratio of the number of changes released (IntraChRel_j or InterChRel_j) to the number of tasks released (TRel_j). Simultaneous discovery of the same inherited change by multiple downstream phases is assumed insignificant. This assumption is valid for relatively small released change densities and a sufficiently short time step in the simulations:

$$\text{CDQA} = \Sigma[((\text{IntraChRel}_j/\text{TRel}_j) + (\text{InterChRel}_j/\text{TRel}_j)) * p(\text{DiscCh}_i) * \text{QA}_i * \text{PS}_j/\text{PS}_i] \text{ for } i \neq j \quad (23)$$

Note that $p(\text{DiscCh}_i) \neq 0$ only for phases that utilize the output of the focal phase. The Tasks to be Coordinated stock is drained through the coordination of tasks. The Coordination Rate (Coord) is based on the resource constraint for coordination (RC_{Coord}) and the rate each development process allows tasks to be coordinated. Using a formulation similar to that for initial completion, quality assurance, and change, this process rate is the work requiring coordination (the Tasks to be Coordinated, TtCoord) divided by the Average Coordination Duration (ACoordD):

$$\text{Coord} = \text{Min}(\text{RC}_{\text{Coord}}, \text{TtCoord}/\text{ACoordD}) \quad (24)$$

For simplicity we have assumed that coordination across phases can be accomplished asynchronously with tools such as memoranda, sharing of CADD/CAM files or electronic mail. An extension to the model would explicitly require various degrees of simultaneity in the resolution of inter-phase coordination. In the extreme case where all coordination requires full simultaneous participation of employees from all affected phases, the coordination rate would be constrained by the minimum of resources devoted to coordination supplied by the involved phases.

Coordination has effects on development projects beyond the resolution of inherited changes described above. A beneficial side effect of coordination is

developers' learning about developments in dependent phases. Improved understanding decreases the generation of changes and increases the likelihood of developers finding their own and inherited changes and thereby increasing their productivity. Since these effects influence underlying rates of change generation, change discovery, and productivity, we model them as non-linear functions of the intensity of coordination. A still more detailed model of coordination would model proactive coordination as well as the reactive function described above.

Model testing

The full model includes structures describing resources (labor), targets such as overall and phase-specific deadlines, and scope, as well as the process structure described above. These structures have been generally based upon previous system dynamics models (Fiddaman *et al.* 1993; Homer 1985; Abdel-Hamid 1984; Richardson and Pugh 1981; Roberts 1974). Resources are allocated to initial completion, quality assurance, iteration, and coordination based upon the relative pressures for each activity as perceived by developers. Pressures for resources are driven by the demand for each activity, as modeled by the process structures above, and are influenced by delays in perception and decision making along with schedule, quality, and cost targets. Complete model equations and documentation are available from the authors. A somewhat simpler version is fully documented in Ford (1995).

Validation of the process portion of the model structure was considered particularly important since these structures are absent in the system dynamics and project management literature. Direct structure tests (Barlas 1989; Forrester and Senge 1980) included discussions with developers and managers in the product development organization at a multinational semiconductor firm to confirm that our model structure closely reflected their process. These tests and interview data support the previously discussed inadequacy of development project mental models and helped validate our model structure. Of particular significance here is that several of the negative loops systematically missed by the managers and developers are described by our model's process structure and are not related to resources, targets, or scope. These loops include the constraint on available work imposed at the beginning of a phase by its Internal Process Concurrence relationship and the negative loop in which completing work reduces the amount of work available to be done. Extreme conditions tests helped validate the integration of process, resource, target, and scope structures.

Behavior-reproduction tests (Forrester and Senge 1980) were also used to validate the model by comparing simulations with field data for a mid-size chip development program, which we call the Python project. Historical data and reference modes were developed from records of the Python project. Specification items were selected as the basis for development tasks in the Product Definition phase. Numbers of specification items and changes to those items in sequential versions of the product specifications were identified and counted in primary records of the project to generate time-series data for the progression of the Product Definition phase. The Design phase consisted primarily of writing the computer code used to lay out physical chip components in the available space. The firm graciously provided the full source code to us for model calibration. The number of lines of code is considered by the company to be sensitive information. However, inspection of the code revealed a uniform code density, so we used vertical inches of code on a printout page as the basis for Design tasks. The code was written in several blocks. Each block included a record of the original completion data and notes describing subsequent revisions. These dates and notes were used to generate time-series data for the initial completion and iteration performed on each block of code. These were then aggregated to generate time series for the Design phase. Similar approaches were used for the other development phases.

Workshops were conducted with developers and managers on the Python project to estimate the relevant Internal and External Process Concurrence relationships. The protocol for the elicitation of the concurrence relationships is described in Ford (1995) and Ford and Sterman (1997). These workshops not only proved to be useful for eliciting the data needed to calibrate the model, but also helped the members of different development groups understand each other's mental models better. As an example, Figure 10 shows expert estimates of the External Process Concurrence relationship between the product definition and design phases of the Python project. The product definition phase develops product specifications describing the Python chip's market, target performance and architecture. The designers use these specifications as the basis for writing the software code used to lay out the chip's individual components. Each estimate in Figure 10 describes the mental model of a participant in the product definition or design phase concerning the question "How much design work can be completed based on how much product definition work has been completed and released?".

It is interesting to note that the two "upstream" people (the marketing representative and the product architect) believe the "downstream" people (the design manager and designer) can and presumably should begin their work quite early, when few product specifications have been released, while those

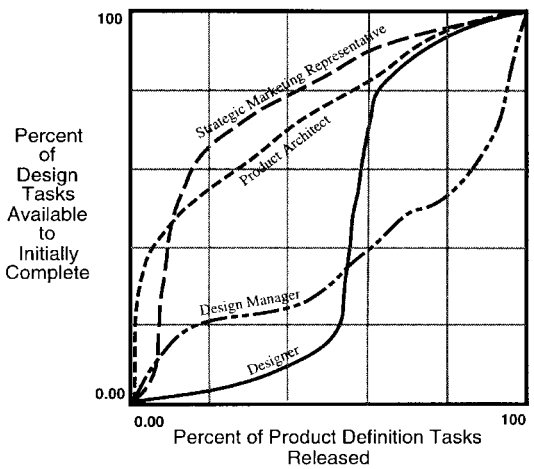

Fig. 10. Four estimates of external process concurrence relationship between the product definition and design phases

downstream believe their work can only progress when a majority of the specifications have been released. This difference in mental models had led to conflict in previous development projects. Different process concurrence relationship estimates were used as the basis for sensitivity testing of the model. Basing sensitivity tests on actual expert estimates can increase confidence in the sensitivity tests over tests based solely on modeler estimates of those relationships.

Data and model simulations for the four phases shown in the network in Figure 2 are shown in Figures 11 through 14.

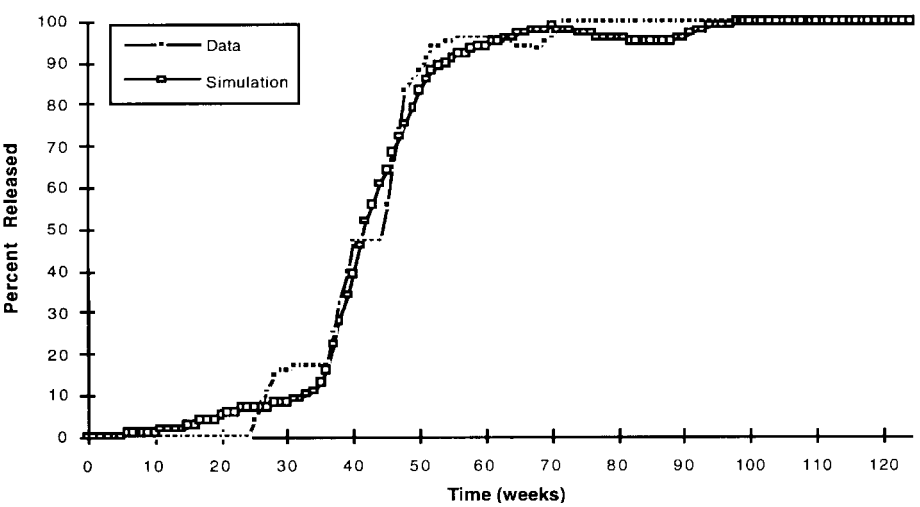

Fig. 11. Reference mode and simulation of product definition phase

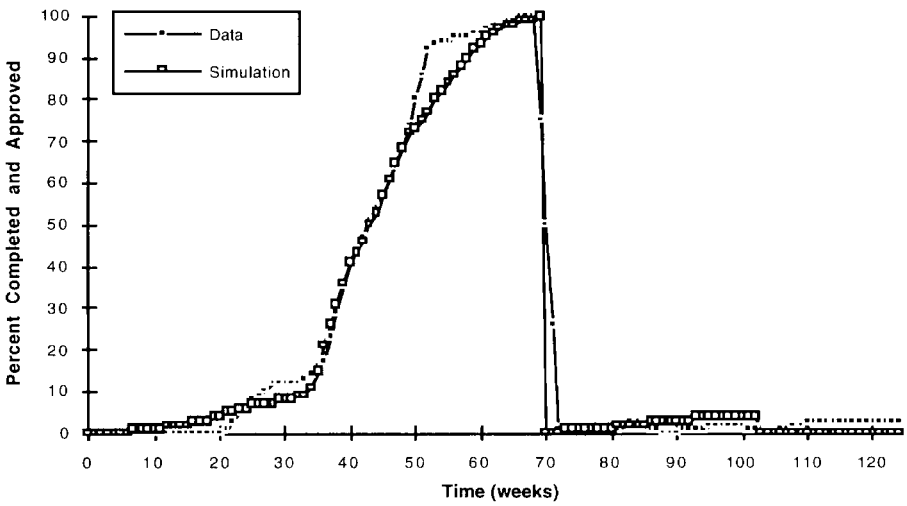

Fig. 12. Reference mode and simulation of design phase

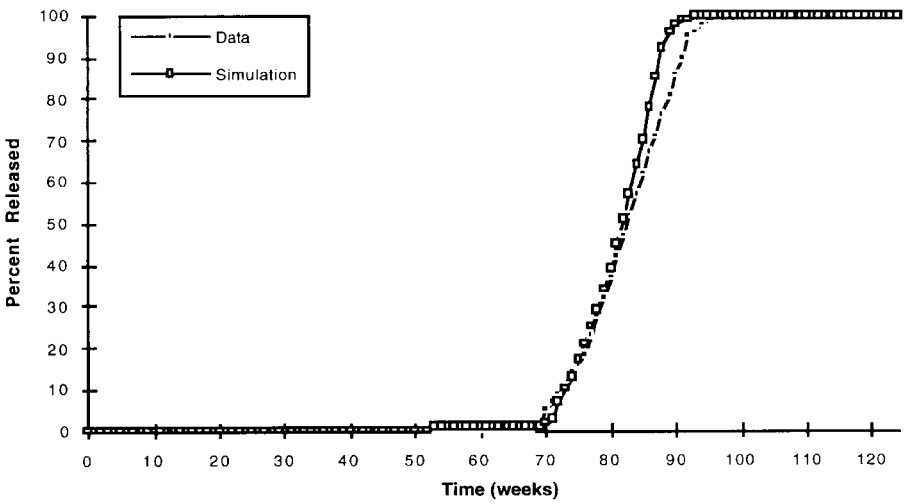

Fig. 13. Reference mode and simulation of test prototypes phase

The behavior patterns of our four project phase simulations closely replicate the patterns of the reference modes. Error statistics (Table 1) were calculated for reference mode and simulated behavior patterns only over times when phases were active. This prevented including data in error statistics that matched exactly as a result of inactivity and would inappropriately reduce calculated error measures. Errors for the four phases are reasonable (average $R^2 = 99\%$ and average MAE as percent of mean = 13%). Partitioning the mean squared error (MSE) using the Theil inequality statistics (Sterman 1984) reveals

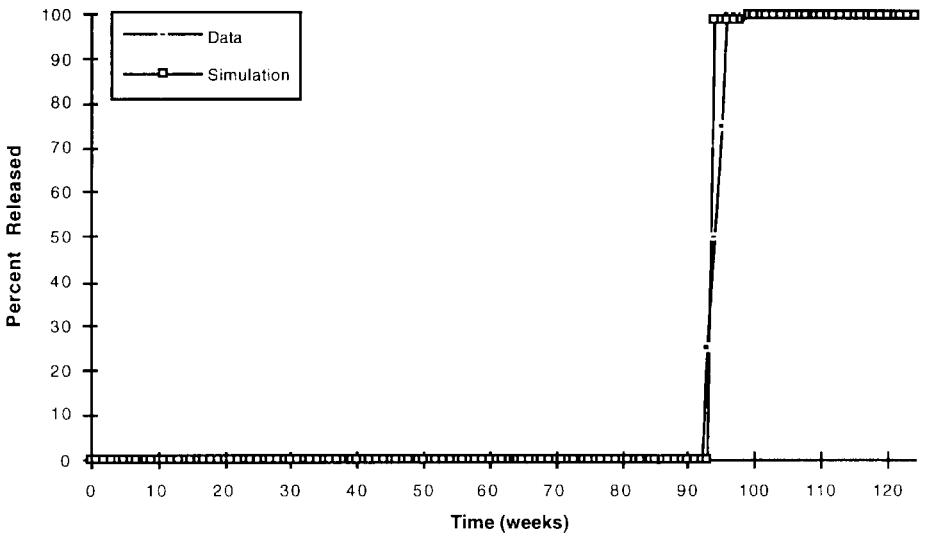

Fig. 14. Reference mode and simulation of reliability/quality control phase

low MSE dominated by unequal covariance in the phases with the large actual data sets and errors. This indicates that the errors are caused primarily by differences in the exact timing between the simulated and actual values, such as the difference in the timing of the drop in the design phase near week 70, or by noise. The fit between simulated and actual data could be improved beyond that shown by specifying process concurrence relationships with higher resolution. However, this would require specification beyond the data resolution justified by the knowledge elicitation process and would result in calibrating the model to process and measurement noise. Process noise can be caused by the natural variations in work effort and productivity around the averages represented in the model. Measurement noise can be caused by the way the original data was coded (e.g., errors in dates of completion of design code) or data reduction errors.

Table 1. Error statistics for the four phases

Phase	Figure No.	R^2 (%)	RMS error (%)	MAE as % of mean	*n*	Thiel inequality statistics Bias (%)	Variation (%)	Covariance (%)
Product definition	11	99	4.2	6	92	7	8	85
Design	12	98	5.7	13	117	10	6	84
Test prototypes	13	99	5.5	12	43	31	40	29
Reliability/ quality control	14	100	17.7	21	6	54	46	0

Impacts of development process dynamics on projects

We use our model to demonstrate that in systems with high iteration, such as product development, modeling process concurrence relationships endogenously and separately from other processes is critical to understanding the dynamics of projects. To illustrate differences in the influence of development activities on performance we simulated project durations with Average Process Activity Durations for a single development activity in all project phases at levels of 25, 50, 100, 200, 400, and 800 percent of the base value used in the model test described above. These durations can represent process differences across industries or firms or differences within a firm due to varying levels of process complexity and through variation in the use of process improvement tools such as information technology. The resulting project schedule performance for each of the four development activities is shown in Figure 15.

Project durations change significantly more in response to changes in the completion duration than in response to changes in the other development activities, as indicated by the steeper slope of the line representing different completion durations. In the case of long durations, the difference exceeds 50%. Combinations of development activity durations, such as short completion and long quality-assurance durations, can create different progress constraints. However, the high sensitivity of project duration to the completion duration relative to the other development activities in this example demonstrates the need for modeling development activities separately.

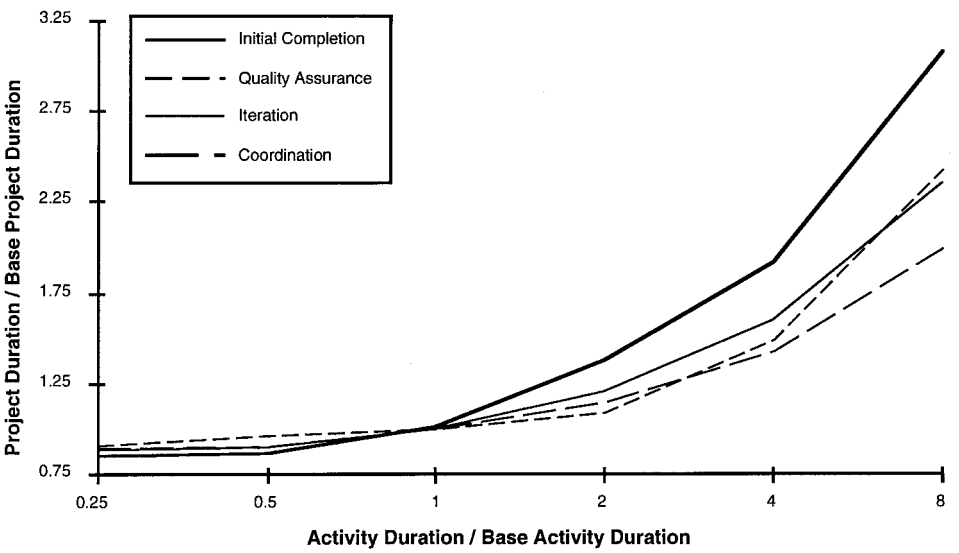

Fig. 15. Impacts of average process activity durations on project duration

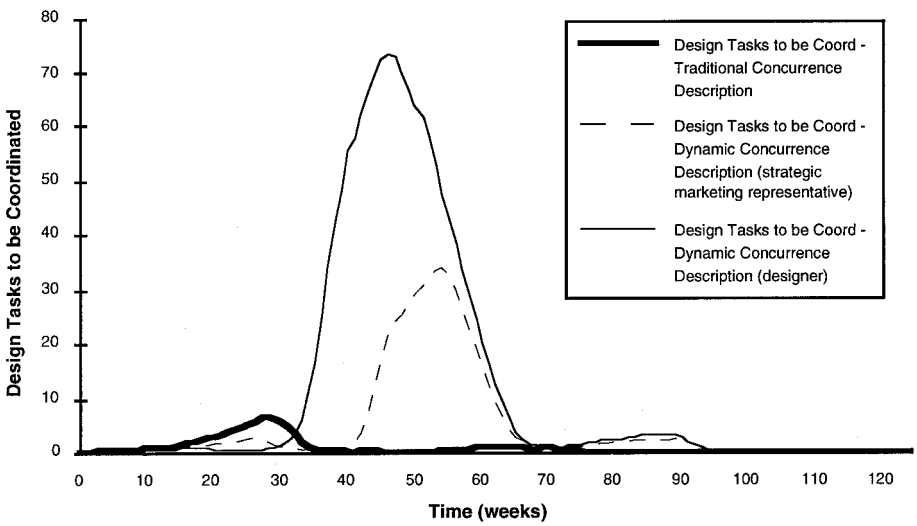

Fig. 16. Design tasks requiring coordination under three External Process Concurrence relationships

To illustrate the importance of modeling process concurrence relationships explicitly and dynamically, we simulated a project three times using a different external process concurrence relationship between the Product Definition and Design phases in each simulation. The three relationships are:

1. the dynamic relationship described by the designer (Figure 10);
2. the dynamic relationship described by the strategic marketing representative (Figure 10);
3. a purely parallel and static relationship such as would be modeled with the Critical Path Method (Figure 8(c)).

Figure 16 shows the number of design tasks to be coordinated using each of the three descriptions. Both explicit and traditional dynamic external process concurrence relationships capture the significant coordination impact of concurrence but the traditional concurrence description recognizes very little coordination. This demonstrates the importance of modeling development concurrence relationships explicitly and endogenously to capture the dynamics of projects.

Conclusions

We have described a dynamic structure for modeling integrated development processes separately from project resources, scope, and targets and have demonstrated the importance of distinguishing process dynamics from other project dynamics. Capturing how development processes affect project

performance by explicitly modeling those processes provides significantly improved descriptions of development team mental models, project constraints, and the drivers of project performance. The significant effects which these process structures have on performance demonstrate the need for integrated project models that include processes, resources, scope, and targets.

Our work contributes to the modeling of project management by adding development processes to the three domains identified by Rodrigues and Bowers (1996) as currently being addressed with system dynamics (monitoring and control, rework, and human resources). We have improved the modeling of development projects by explicitly and formally describing and testing structures for process concurrence, four distinct development activities, inter- and intra-phase changes, explicit feedback flows of changes between phases, and change co-flow structures in an integrated generic project model. The model can be used to investigate further the impacts of process, resources, scope, and targets on project performance. We have also identified process concurrence relationships as an important aspect of practitioner mental models and a topic for future work.

Project performance problems can be expected to increase as competitive forces and new development paradigms increase project complexity. Our work points to an important gap between the primary methods currently used to describe, model, communicate, and manage projects in practice and the complexity of the structures that actually drive the behavior of those projects. Current project management literature directed toward practitioners is based on open-loop, single-link linear relationships which are often reduced to lists of rules-of-thumb (e.g., Thomsett 1990). Field data collected during our research indicates that development practice also suffers from these limitations (Ford 1995). Current project modeling tools are incapable of capturing the dynamics produced by the development processes described here. Barring reduced project complexity, performance cannot significantly improve without the development of tools for the description, modeling, and management of dynamic development processes. Our model helps fill this gap by providing a framework for designing and testing development process structures and management policies. By explicitly modeling development processes and relating process, resources, scope, and target features to performance, our model plays a similar role for dynamic project features as the Critical Path and PERT methods provide for static project features.

Future research can test the broader application of our model by applying it to other development projects and processes. Such tests will provide a basis for comparing concurrence relationships across project characteristics (e.g., big versus small projects or simple versus complex projects) and project types

(e.g., different industries or different objectives). The results of such work can lead to the abstraction of more general dynamic lessons for managers and improved project performance. The resulting models can be used to integrate further the influences of process, resources, targets, and scope on quality, cost, and schedule performance. Improved descriptions of development processes in models can also improve the use of those models as learning tools to improve development team mental models.

Acknowledgements

The authors thank the Organizational Learning Center and the System Dynamics Group at the Sloan School of Management and the Python organization for financial support. Special thanks to the members of the Python project for their interest, commitment and time.

Appendix: the change co-flow structures

The change co-flow structures simulate the movement of changes arising within a development phase (internal) or due to inherited change work (external) with a stock and flow structure directly parallel to the development task sector described above. The change co-flow structures will be described by following process equations from the development tasks sector (in italics here) with the parallel change co-flow equation. Co-flow parameter abbreviations without subscripts apply to both internal and external change co-flows. Differences in equations describing internal changes (subscripted with i) and external changes (subscripted with e) follow the general co-flow equations, as do co-flow parameter abbreviations.

Stocks

$$(d/dt)(Tn\mathrm{C}) = -CT \tag{1}$$

$$(d/dt)(\mathrm{PotCh}) = -\mathrm{GenChComp} \tag{25}$$

PotCh = Potential Changes

GenChComp = Generate Changes during Completion activity

$$(d/dt)(TCnotCk) = CT + ChT - DiscIntraPhaseCh - DiscInterPhaseCh - ApprT \quad (2a)$$

$$(d/dt)(\text{UnDiscCh}) = \text{GenChComp} + \text{GenChCh} - \text{DiscInternalCh} - \text{DiscExternalCh} - \text{ApprCh} \quad (26)$$

UnDiscCh = Undiscovered Changes
GenChComp = Generate Changes during Completion activity
GenChCh = Generate Changes during Change activity
DiscInternalCh = Discover Internal Changes
DiscExternalCh = Discover External Changes
ApprCh = Approve Changes

$$(d/dt)(TtCh) = DiscIntraPhaseCh + Coord - ChT \quad (3a)$$

$$(d/dt)(\text{KnownCh}) = \text{DiscInternalCh} + \text{CoordCh} - \text{GenChCh} - \text{CorrCh} \quad (27)$$

KnownCh = Known Changes
DiscInternalCh = Discover Internal Changes
CoordCh = Coordinate Changes
GenChCh = Generate Changes during Change activity
CorrCh = Correct Changes

$$(d/dt)(TAppr) = ApprT - RelT \quad (4)$$

$$(d/dt)(\text{ChAppr}) = \text{ApprCh} - \text{RelCh} \quad (28)$$

ChAppr = Changes Approved
RelCh = Release Changes

$$(d/dt)(TRel) = RelT - CDQA \quad (5a)$$

$$(d/dt)(\text{ChRel}) = \text{RelCh} - \text{RecRelCh} \quad (29)$$

ChRel = Changes Released
RelCh = Release Changes
RecRelCh = Receive Released and Discovered Changes

$$(d/dt)(TtCoord) = DiscInterPhaseCh + CDQA - Coord \quad (21)$$

$$(d/dt)(\text{ChtCoord}) = \text{DiscExternalCh} + \text{RecRelCh} - \text{CoordCh} \quad (30)$$

ChtCoord = Changes to be Coordinated

Flows and change densities

$$ChT = Min(RC_{ChT}, TtCh/AChD) \quad (6)$$

$$\text{GenChCh} = (\text{KnownCh/TtCh}) * p(\text{GenCh}_{\text{int}}) * \text{ChT} \quad (31)$$

$$\text{CorrCh} = (\text{KnownCh/TtCh}) * (1 - p(\text{GenCh}_{\text{int}})) * \text{ChT} \quad (32)$$

GenChCh = Generate Changes during Change activity

$p(\text{GenCh}_{\text{int}})$ = probability of generating an internal change

CorrCh = Correct Changes

Changing tasks has two effects on changes: the generation of new internal changes (Eq. 31) and the correction of discovered changes (Eq. 32). We consider changes generated during the change activity to be externally caused when the change activity is performed on an externally generated change. We therefore model them in the external change co-flow. However, since the changes are being done within a phase that has a particular set of characteristics and propensity to generate changes, the probability of generating a change during change activity in the external change co-flow is the same as the probability of generating an internal change. The complement of the tasks which require changes during the change activity is corrected without generating new changes.

$$DiscIntraPhaseCh = QA * p(DiscChTask) * p(IntraPhaseCh) \quad (8a)$$

$$\text{DiscInternalCh} = \text{DiscIntraPhaseCh} \quad (33)$$

$$p(\text{IntraPhaseCh}_{\text{i}}) = (\text{UnDiscCh}_{\text{i}}/\text{TCnotCk}) \quad (34.1)$$

$$p(\text{IntraPhaseCh}_{\text{e}}) = 0 \quad (34.2)$$

DiscInternalCh = Discover Internal Changes

UnDiscCh = Undiscovered Changes

$$DiscInterPhaseCh = QA * p(DiscCh) * (p(InterPhaseCh) - (p(IntraPhaseCh) * p(InterPhaseCh))) \quad (22)$$

$$\text{DiscExternalCh} = \text{DiscInterPhaseCh} \quad (35)$$

$$p(\text{InterPhaseCh}_{\text{i}}) = 0 \quad (36.1)$$

$$p(\text{InterPhaseCh}_{\text{e}}) = (\text{UnDiscCh}_{\text{e}}/\text{TCnotCk}) \quad (36.2)$$

DiscExternalCh = Discover External Changes

UnDiscCh = Undiscovered Changes

$$ApprT = QA - DiscIntraPhaseCh - DiscInterPhaseCh \quad (9a)$$

$$\text{ApprCh}_i = (\text{QA} * p(\text{IntraPhaseCh})) - \text{DiscInternalCh} \quad (37.1)$$

$$\text{ApprCh}_e = (\text{QA} * p(\text{InterPhaseCh})) - \text{DiscExternalCh} \quad (37.2)$$

ApprCh = Approve Changes

DiscInternalCh = Discover Internal Changes

DiscExternalCh = Discover External Changes

$$RelT = TAppr/RelTime \quad (11)$$

$$\text{RelCh} = \text{RelT} * (\text{ChAppr/TAppr}) \quad (38)$$

RelCh = Release Changes

TAppr = Tasks Approved

$$CT = Min(RC_{CT},\ CTAvail/ACTD) \quad (13)$$

$$\text{GenChComp}_i = \text{CT} * p(\text{GenCh}_{int}) \quad (39.1)$$

$$\text{GenChComp}_e = \text{CT} * (p(\text{GenCh}_{ext}) - (p(\text{GenCh}_{ext}) * p(\text{GenCh}_{int}))) \quad (39.2)$$

GenChComp = Generate Changes during Completion

p(GenCh) = probability of generating a change

We assume that those tasks caused to need change by both internal and external causes (represented by the intersection of the probabilities of the generation of a change due to internal or external causes) are processed as being internally changed. This is consistent with reasoning presented in the body of this paper and the formulation of the separation of discovered changes due to internal and external causes (Eqs. 8a and 22).

$$CDQA = \Sigma[((IntraChRel_j/TRel_j) + (InterChRel_j/TRel_j)) * p(DiscCh_i) * QA_i * PS_j/PS_i] \text{ for } i \neq j \quad (23)$$

$$\text{RecRelCh} = (\text{ChRel/TRel}) * p(\text{DiscChDown}) * \text{QADown} \quad (40)$$

RecRelCh = Receive Released and Discovered Changes

p(DiscChDown) = probability of downstream phase discovering change

QADown = Quality Assurance rate in downstream phase

$$Coord = Min(RCCoord,\ TtCoord/ACoordD) \quad (24)$$

$$\text{CoordCh}_i = 0 \quad (41.1)$$

$$\text{CoordCh}_e = \text{Coord} \quad (41.2)$$

Notes

1. Some structures with objectives similar to those described in this paper are apparently included in the PMMS (Project Management Modeling System) model, which is described conceptually in Lyneis (1996) and Pugh-Roberts (undated), but the PMMS model is not available for evaluation or comparison.

References

Abdel-Hamid, T. K. 1984. The Dynamics of Software Development Project Management: An Integrative System Dynamics Perspective. Doctoral thesis, MIT, Cambridge, MA.

Adler, P. S. 1995. Interdepartmental interdependence and coordination: The case of the design/manufacturing interface. *Organization Science* 6(2): 147–167.

Barlas, Y. 1989. Multiple test for validation of system dynamics type of simulation models. *European Journal of Operational Research* 42, 59–87.

Brooks, F. P. 1975. *The Mythical Man-month*. Reading, MA: Addison-Wesley.

Chao, L. 1993. Improving Quality and Time to Market in the VLSI Product Life Cycle. Unpublished master's thesis, Sloan School of Management, MIT, Cambridge, MA.

Christensen, L. C., T. R. Christian, Y. Jin, J. Kung and R. E. Levitt. 1996. Modeling and simulation in enterprise integration—a framework and an application in the offshore oil industry. *Concurrent Engineering: Research and Applications* 4(3), 247–259.

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

Construction Industry Institute 1990. The *Quality Performance Management System: A Blueprint for Implementation*: Report 10-3, Construction Industry Institute, University of Texas, Austin, TX.

Cooper, K. G. 1980. Naval ship production: A claim settled and a framework built. *Interfaces* 10(6), 20–36.

———. 1993. The change cycle: Benchmarks for the project manager. *Project Management Journal* 24(1), 17–22.

———. 1994. The $2,000 hour: How managers influence project performance through the change cycle. *Project Management Journal* 25(1), 11–24.

Cooper, R. G. and E. Kleinschmidt. 1994. Determinants of timeliness in product development. *The Journal of Product Innovation Management* 11(5), 381–396.

Davis, K. and W. B. Ledbetter. 1987. *Measuring Design and Construction Quality Cost*. Austin, TX: Construction Industry Institute, University of Texas.

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

Diehl, E. and J. Sterman. 1995. Effects of feedback complexity on dynamic decision making. *Organizational Behavior and Human Decision Processes* 62(2), 198–215.

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

Fiddaman, T., R. R. Oliva and R. Aranda. 1993. Modeling the impact of quality initiatives over the software product life cycle. *Proceedings of the 1993 International System Dynamics Conference*, Cancun, Mexico.

Ford, D. N. 1995. The Dynamics of Project Management: An Investigation of the Impacts of Project Process and Coordination on Performance. Doctoral thesis, Massachusetts Institute of Technology, Cambridge, MA.

———. 1996. Impacts of product development process structure on cycle time and project manageability. *Proceedings of the Third International Product Development Conference*, European Institute for Advanced Studies in Management, Fountainbleau, France, 15–16 April.

Ford, D., A. Hou and D. Seville. 1993. *An Exploration of Systems Product Development at Gadget Inc.* Technical Report D-4460, System Dynamics Group, Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA.

Ford, D. N. and J. D. Sterman. 1997. Expert Knowledge Elicitation to Improve Mental and Formal Models. Working Paper #3953-97-MA. Sloan School of Management. Massachusetts Institute of Technology, Cambridge, MA.

Forrester, J. W. and P. M. Senge. 1980. Tests for building confidence in system dynamics models. *TIMS Studies in the Management Sciences* 14, 209–228.

Golenko-Ginzburg, D. and A. Gonik. 1996. On-line control model for cost simulation network projects. *Journal of the Operations Research Society* 47, 266–283.

Halpin, D. W. and R. W. Woodhead. 1980. *Construction Management*. New York: John Wiley & Sons.

Homer, J. 1985. Worker burnout: a dynamic model with implications for prevention and control. *System Dynamics Review* 1(1), 42–62.

Homer, J., J. Sterman, B. Greenwood and M. Perkola. 1993. Delivery time reduction in pump and paper mill construction projects: A dynamic analysis of alternatives. *Proceedings of the 1993 International System Dynamics Conference*. Monterey Institute of Technology, Cancun, Mexico.

Kim, D. H. 1988. Sun Microsystems, Sun3 Product Development/Release Model. Technical Report D-4113. System Dynamics Group, Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA.

Kleinmuntz, D. N. 1993. Information processing and misperceptions of the implications of feedback in dynamic decision making. *System Dynamics Review* 9(3), 223–237.

Lyneis, J. M. 1996. *Critical Path Approaches to Project Management—Applicability for Determining Estimates to Complete, Project Duration, and Delay and Disruption*. Cambridge, MA: Pugh-Roberts Associates.

Moder, J. J., C. R. Phillips and E. W. Davis. 1983. *Project Management with CPM, PERT and Precedence Diagramming*. New York: Van Nostrand Reinhold.

Morelli, M. D., S. D. Eppinger and R. K. Gulati. 1995. Predicting technical communications in product development organizations. *IEEE Transactions on Engineering Management* 42(3), 215–222.

Osborne, S. M. 1993. Product Development Cycle Time Characterization through Modeling of Process Change. Unpublished master's thesis, Sloan School of Management, MIT, Cambridge, MA.

Paich, M. and J. Sterman. 1993. Boom, bust, and failures to learn in experimental markets. *Management Science* 39(12), 1439–1458.

Pugh, E. W. 1984. *Memories that Shaped an Industry, Decisions Leading to IBM Systems/360*. Cambridge, MA: MIT Press.

Pugh-Roberts Associates undated. *Program Management Modeling System, PMMS*. Cambridge, MA: Pugh-Roberts Associates.

Rechtin, E. 1991. *Systems Architecting, Creating and Building Complex Systems*. Englewood Cliffs, NJ: Prentice Hall.

Reichelt, K. S. 1990. *Halter Marine: A Case Study in the Dangers of Litigation*. Technical Report D-4179, System Dynamics Group, MIT Sloan School of Management, Cambridge, MA.

Richardson, G. P. and A. L. Pugh III. 1981. *Introduction to System Dynamics Modeling with Dynamo*. Cambridge, MA: MIT Press.

Roberts, E. B. 1974. A simple model of R&D project dynamics. In E. B. Roberts, (ed.), *Managerial Applications of System Dynamics*, pp. 293–314. Cambridge, MA: Productivity Press.

Rodrigues, A. and J. Bowers. 1996. System dynamics in project management: a comparative analysis with traditional methods. *System Dynamics Review* 12(2), 121–139.

Rodrigues, A. and T. Williams. 1996. System dynamics in software project management: towards the development of a formal integrated network. *European Journal of Information Systems* 6, 51–56.

Rosenau, M. D. and J. Moran. 1993. *Managing the Development of New Products, Achieving Speed and Quality Simultaneously Through Multifunctional Teamwork*. New York: Van Nostrand Reinhold.

Rosenthal, S. R. 1992. *Effective Product Design and Development*. Homewood, IL: Business One Irwin.

Smith, R. P. and S. D. Eppinger. 1997. A predictive model of sequential iteration in engineering design. *Management Science* 43(8), 1104–1120.

Sterman, J. 1984. Appropriate summary statistics for evaluating the historical fit of system dynamics models. *Dynamica* 10(2), 51–66.

———. 1994. Learning in and about complex systems. *System Dynamics Review*. 10(2–3), 291–330.

Steward, D. V. 1981. The design structure system: A method for managing the design of complex systems. *IEEE Transactions on Engineering Management*. EM-28(3), 71–74.

Thomsett, M. C. 1990. *The Little Black Book of Project Management*. New York: American Management Association.

Wetherbe, J. C. 1995. Principles of cycle time reduction: You can have your cake and eat it too. *Cycle Time Research* 1(1), 1–24.

Wheelwright, S. C. and K. B. Clark. 1992. *Revolutionizing Product Development, Quantum Leaps in Speed, Efficiency, and Quality*. New York: The Free Press.

Williams, T., C. Eden, F. Ackerman and A. Tait. 1995. The effects of design changes and delays on project costs. *Journal of Operational Research Society* 46, 809–818.

Womack, J. P., D. T. Jones and D. Roos. 1990. *The Machine that Changed the World*. New York: Rawson Associates.